Inconsistency Management for Description Logics

Kevin Lee

Supervisors: Prof. Norman Foo and Prof. Tommie Meyer

A thesis in fulfillment of the requirements for the degree of Doctor of Philosophy

School of Computer Science and Engineering

Faculty of Engineering

September 2012

PLEASE TYPE

THE UNIVERSITY OF NEW SOUTH WALES
Thesis/Dissertation Sheet

Surname or Family name: **LEE**

First name: **KEVIN** Other name/s: **SIN YEE**

Abbreviation for degree as given in the University calendar: **PhD**

School: **School of Computer Science and Engineering (CSE)** Faculty: **Faculty of Engineering**

Title: **Inconsistency Management for Description Logics**

Abstract 350 words maximum: (PLEASE TYPE)

Description logics belong to a family of knowledge representation formalisms that are widely used for representing ontologies. However, ontologies are subject to changes and are susceptible to logical errors as they evolve. Ontology reasoners are able to identify these errors, but they provide very limited support for resolving them. In particular, the existing tools do not provide adequate support to prevent logical errors from being introduced into an ontology. In this research, we investigate three different operations that are directly related to the management of logical inconsistencies in ontologies, namely: ontology contraction, integration and debugging. Ontology contraction concerns the removal of information from a set of description logic sentences, where the resulting set of sentences is consistent. Ontology integration is the problem of combining multiple sets of description logic sentences in a consistent manner. Ontology debugging deals with the removal of description logic sentences to restore the consistency of an ontology. In this regard, contraction and integration can be considered as prevention of logical errors, and debugging as cure.

We present a construction of contraction for description logics based on the well-known partial meet contraction for belief bases from the area of belief change. We show that this construction produces more refined solutions, and we show that this construction is governed by a refined set of contraction postulates. Moreover, we recast a class of propositional knowledge integration strategies known as adjustments. We show that these strategies cannot be directly used in the description logic setting due to limitations in the expressive power of description logics. We then provide two new adjustment strategies, which are appropriate for description logics, and we show that these strategies produce more refined solutions. Furthermore, we study a tableau-based algorithm that identifies the maximally satisfiable subsets (and minimally unsatisfiable subsets) of an ontology. We show that classical blocking do not guarantee completeness in the presence of cyclic definitions, and we provide revised blocking conditions and prove that they preserve both soundness and completeness. Finally, we introduce a diagrammatic approach for debugging ontologies based on Reduced Ordered Binary Decision Diagrams (ROBDDs).

Declaration relating to disposition of project thesis/dissertation

I hereby grant to the University of New South Wales or its agents the right to archive and to make available my thesis or dissertation in whole or in part in the University libraries in all forms of media, now or here after known, subject to the provisions of the Copyright Act 1968. I retain all property rights, such as patent rights. I also retain the right to use in future works (such as articles or books) all or part of this thesis or dissertation.

I also authorise University Microfilms to use the 350 word abstract of my thesis in Dissertation Abstracts International (this is applicable to doctoral theses only).

..... Signature ..... Witness 30/09/2012 Date

The University recognises that there may be exceptional circumstances requiring restrictions on copying or conditions on use. Requests for restriction for a period of up to 2 years must be made in writing. Requests for a longer period of restriction may be considered in exceptional circumstances and require the approval of the Dean of Graduate Research.

FOR OFFICE USE ONLY Date of completion of requirements for Award:

THIS SHEET IS TO BE GLUED TO THE INSIDE FRONT COVER OF THE THESIS

ORIGINALITY STATEMENT

'I hereby declare that this submission is my own work and to the best of my knowledge it contains no materials previously published or written by another person, or substantial proportions of material which have been accepted for the award of any other degree or diploma at UNSW or any other educational institution, except where due acknowledgement is made in the thesis. Any contribution made to the research by others, with whom I have worked at UNSW or elsewhere, is explicitly acknowledged in the thesis. I also declare that the intellectual content of this thesis is the product of my own work, except to the extent that assistance from others in the project's design and conception or in style, presentation and linguistic expression is acknowledged.'

Signed .....

Date 30/09/2012

COPYRIGHT STATEMENT

'I hereby grant the University of New South Wales or its agents the right to archive and to make available my thesis or dissertation in whole or part in the University libraries in all forms of media, now or here after known, subject to the provisions of the Copyright Act 1968. I retain all proprietary rights, such as patent rights. I also retain the right to use in future works (such as articles or books) all or part of this thesis or dissertation.
I also authorise University Microfilms to use the 350 word abstract of my thesis in Dissertation Abstract International (this is applicable to doctoral theses only).
I have either used no substantial portions of copyright material in my thesis or I have obtained permission to use copyright material; where permission has not been granted I have applied/will apply for a partial restriction of the digital copy of my thesis or dissertation.'

Signed .....

Date 30/09/2012

AUTHENTICITY STATEMENT

'I certify that the Library deposit digital copy is a direct equivalent of the final officially approved version of my thesis. No emendation of content has occurred and if there are any minor variations in formatting, they are the result of the conversion to digital format.'

Signed .....

Date 30/09/2012

To

Mum and Dad

Abstract

Description logics belong to a family of knowledge representation formalisms that are widely used for representing ontologies. However, ontologies are subject to changes and are susceptible to logical errors as they evolve. Ontology reasoners are able to identify these errors, but they provide very limited support for resolving them. In particular, the existing tools do not provide adequate support to prevent logical errors from being introduced into an ontology. In this research, we investigate three different operations that are directly related to the management of logical inconsistencies in ontologies, namely: ontology contraction, integration and debugging. Ontology contraction concerns the removal of information from a set of description logic sentences, where the resulting set of sentences is consistent. Ontology integration is the problem of combining multiple sets of description logic sentences in a consistent manner. Ontology debugging deals with the removal of description logic sentences to restore the consistency of an ontology. In this regard, contraction and integration can be considered as prevention of logical errors, and debugging as cure.

We present a construction of contraction for description logics based on the well-known partial meet contraction for belief bases from the area of belief change. We show that this construction produces more refined solutions, and we show that this construction is governed by a refined set of contraction postulates. Moreover, we recast a class of propositional knowledge integration strategies known as adjustments. We show that these strategies cannot be directly used in the description logic setting due to limitations in the expressive power of description logics. We then provide two new adjustment

strategies which are appropriate for description logics, and we show that these strategies produce more refined solutions. Furthermore, we study a tableau-based algorithm that identifies the maximally satisfiable subsets (and minimally unsatisfiable subsets) of an ontology. We show that classical blocking do not guarantee completeness in the presence of cyclic definitions, and we provide revised blocking conditions and prove that they preserve both soundness and completeness. Finally, we introduce a diagrammatic approach for debugging ontologies based on Reduced Ordered Binary Decision Diagrams (ROBDDs).

Contents

List of Figures

Chapter 1

Introduction

The proliferation of the World Wide Web (WWW) in the past decade has made a dramatic impact to the way people live and interact with the world. Information has become more accessible and more diverse. Search engines provide a convenient and powerful means to filter important information. Yet, the often overwhelming amount of search results make it prohibitive for users to identify and locate useful information. In addition, information available on the web is still primarily in text that requires human effort to parse, interpret and make sense from it. This has led to investigations on technologies that could improve the web, in particular how information can be shared and acquired across the web more easily, and how information can be exchanged between agents with minimal human involvement and interactions. A promising line of research towards this direction is the so-called Semantic Web. The dream of the Semantic Web as envisioned by Tim-Berners Lee is:

> "The Web was designed as an information space, with the goal that it should be useful not only for human-human communication, but also that machines would be able to participate and help." [BL98]

> "The Semantic Web is not a separate Web but an extension of the current one, in which information is given well-defined meaning, better enabling computers and people to work in cooperation." [BLHL01]

An important goal, therefore, as pointed out by Tim-Berners Lee is to devise a means to represent information in an unambiguous manner, that enables the exchange of information among machines across the web. For this to be realised, there needs to be a formal representation that not only provides sufficient expressive power, but also supports powerful and efficient reasoning. Research in Knowledge Representation and Reasoning (KRR), especially formal representations of ontologies provides the foundation for this vision. Nevertheless, there are serious challenges that remain to be solved as well as some concerns with the practical use of ontologies in the real world. The challenge that is central to our investigation is the management of logical errors in ontologies.

In this dissertation, we address the issue of inconsistency management in formally represented ontologies. Our investigation is primarily from the perspective of formal logics, where ontologies are represented as description logic knowledge bases. We consider ontologies undergoing changes and we look at the impact of these changes on the logical integrity of the ontologies.

1.1 What is an Ontology?

The term 'ontology' originates from the area of philosophy where it was taken as the study of being or existence. There is, however, a huge discrepancy in how this term is being defined and used in Computer Science. We list below the definitions provided by Tom Gruber [Gru93, Gru95]:

> "An *ontology* is an explicit specification of a conceptualisation."

> "A specification of a representational vocabulary for a shared domain of discourse – definitions of classes, relations, functions, and other objects – is called an ontology."

While these definitions are generally vague and that there is no consensus on the precise meaning of this term, it is often perceived that an ontology is a representation that establishes a view of the world. Its purpose is to standardise and formalise concepts within a domain so that they can be unambiguously represented. For example,

some ontology languages consists of the disjunction construct or the "OR" construct. This construct is used in our daily lives but its intended semantics is often unclear. It could be used to mean "*only* one of the statements is true" or "*at least* one of the statements is true". In this sense, ontological languages are not different from Knowledge Representation (KR) languages where the heart of KR research is to devise appropriate representations to serve some particular purposes in some domain of discourse. Therefore, the more important question here is *not* what ontologies are, but rather, what their functions are and how they are practically used in the real world. One convenient way to assess the usefulness of an ontology representation is to consider its expressive power and its computational efficiency in providing some of the reasoning services. In this regard, description logics are most suited for representing ontologies because they have been carefully considered in the literature both in terms of their expressive power and computational efficiency.

1.2 Description Logic and Logical Inconsistencies

Description logics (DLs) has been widely accepted as a suitable class of logical formalisms to represent ontologies. An ontology represented in description logics is a DL knowledge base or simply a set of DL sentences. This set of DL sentences can be further divided into components: a set of TBox axioms and a set of ABox assertions. The former contains intensional knowledge and the latter contains extensional knowledge. Similar to propositional logic and first-order logic, description logics are monotonic logics, meaning that their consequence relation exhibits monotonic behaviours. That is, adding a sentence α to a set of sentences A will *never* lead to a smaller set of consequences than that of A. Similarly, removing a sentence α from a set of sentences A will *never* lead to a larger set of consequences than that of A.

In some fragments of description logics, it is possible to introduce conflicting information into a DL knowledge base. For example, the classical description logic $\mathcal{ALC}$ allows one to express complement of a concept description. This means that a DL knowledge base in $\mathcal{ALC}$ may, for example, contain a statement like "Tweety is a bird" and since

complement of a concept description can be expressed, one may also have the statement "Tweety is *not* a bird" in the same DL knowledge base. Hence, this DL knowledge base contains at least one conflicting piece of information. This is undesirable in many classical logics, including classical description logics, where a single inconsistency could "corrupt" the whole knowledge base and results in a theory that contains every sentence in the language. In other words, once an inconsistency is introduced into the knowledge base, one could conclude everything from it, thus making it no longer useful for inferencing. Therefore, it is important to maintain an ontology in a consistent state.

However, ontologies are expected to be constantly undergoing changes and these changes are prone to introduction of logical errors. For example, sentences could be added to an ontology to expand the scope of an ontology, or the part of the world that one wants to model has changed and modifications are needed to update the ontology, or one might want to update an ontology in order to provide a more finer-grain view for their intended application. All of these would require changing an ontology. These changes are often made manually by knowledge engineers who are familiar with the specific domain that they are trying to model, but they may not necessarily have a complete understanding of ontology languages or how inconsistencies are introduced.

The existing reasoners provide a convenient way to determine the consistency (or satisfiability) of an ontology, but they provide limited support to prevent inconsistencies from happening or to remove inconsistencies once they are introduced into an ontology. To understand this problem better, consider the following scenario:

A Sample Scenario

> *Bob is a knowledge engineer who is an expert in the military domain, but knows little about ontology languages. He was asked to build an ontology to be used in a situation awareness system. Bob makes use of an existing ontology, which provides him the basic skeleton for the ontology he wants to build, but the existing ontology does not capture many of the important concepts that he needs for the situation awareness system. Bob's intention is to extend the ontology with concepts that he needs, so he makes use of an*

ontology editor Protege to try and extend the ontology.

Although Bob is an expert in the military domain, it is likely that he will be introducing logical errors while he tries to extend the existing ontology with additional concept and role axioms. The fact that he is using an existing ontology means that Bob may not necessarily fully understand the implications of all the information in the ontology. It is also likely that the original designer of the ontology has a different view (conceptualisation) about the military domain from that of Bob. This makes extending an ontology difficult and also more prone to logical errors.

Debugging an Ontology with a Reasoner

An effective, yet cumbersome, way to ensure the consistency of an ontology is by checking its consistency each time a sentence is changed. This is similar to the strategy that a computer programmer adopts when he runs a debugger each time he makes changes to a program. Obviously, if new bugs were introduced after making the changes, then these changes are necessarily responsible for the bugs. Similarly for ontology debugging, if adding new sentences causes the ontology to become inconsistent, then these sentences alone or together with other sentences in the ontology are necessarily the culprit of the inconsistencies. However, there are a number of disadvantages to this approach. Firstly, it is a time-consuming process and running the reasoner on a large ontology (e.g., SNOMED) could take a considerable amount of time. Secondly, it does not give any information about the exact cause of the problem and it does not provide a way to resolve this problem. This would be particularly useful in large ontologies.

In particular, one of the emphases of Semantic Web research is the reuse aspect of ontologies. Ontologies are often built by domain experts over a long period of time (e.g. SNOMED and GALEN).

These ontologies contain a large number of concepts and cannot be easily reproduced. Therefore, knowledge engineers are encouraged to make use of these existing ontologies as much as possible for their applications. However, it is also recognised that these ontologies have limitations and cannot always suit all applications, which means changes are required to tailor ontologies to suit specific needs.

1.3 An Ontology Example: Cluedo

The aim of this section is to demonstrate how an ontology can be created, what can be achieved with ontology reasoning and what the limitations of ontologies are. We achieve these goals by modelling a board game called Cluedo as an ontology represented in description logic $\mathcal{ALCQO}$. The representation we came up makes use of all the constructors in $\mathcal{ALCQO}$. However, this is by no means the only way to represent the game. Some constructors are deliberately used to demonstrate the expressive power of DL.

As a brief introduction, Cluedo is a popular board game invented in the late 40s. The goal of the game is to identify three cards from a stack of twenty-one cards, which are hidden away in an envelope at the start of the game. The cards are divided into three categories: six weapon cards, nine room cards and six character cards. The typical setting of the game involves 6 players. The stack of cards are shuffled and three cards are distributed to each player with the remaining three (one room card, one suspect card and one weapon card) hidden away in an envelope. Players are allowed to see their own cards but not the others. At each round of the game, each player takes turns to ask about the cards his/her adjacent player (clock-wise) holds. Specifically, the player is only allowed to name three cards and ask if the adjacent player has at least one of them. If the adjacent player has one of three cards then he is obliged to show one of them to the requested player as a proof. Otherwise, he has to say that he is not in possession of any of the cards. The game continues until a player could work out what the hidden cards are.

This game can be expressed as an ontology in KRSS format[1] as shown in Program 1. We use KRSS format to represent the Cluedo ontology for two reasons: (1) KRSS is a valid input format for many major reasoners (e.g., FaCT++, CEL and RACER), and this is exactly how one would construct and execute an ontology; (2) It can be easily written and read by human, as opposed to OWL that is designed to be parsed by machines.

The line `(implies player (=3 has_card card))` states that each player has exactly three cards, and the line `(equiv card (or suspect_card weapon_card room_card))`

[1] http://dl.kr.org/krss-spec.ps

states that each card is either a room card, a suspect card, or a weapon card. The line `(implies envelope (=1 has_card room_card))` says that an envelope has exactly one room card, and similarly, there is a suspect card and a weapon card in the envelope. Next, we describe instances of each of the three card types. For example, `(instance ballroom room_card)` states that ballroom is a room card.

Once the game is properly represented as an ontology (as we have done), one could feed it into a KRSS reasoner (e.g. RacerPro [HMW03]) and query about the ontology. For example, one could ask for the list of room cards from the ontology using an appropriate querying language (e.g. nRQL). We could also extend the ontology dynamically while we play the game. For example, one could insert the following assertions while the game is being played:

```
(instance sam player)
(instance sam (some has_card (one-of hall dagger white)))
```

The first line states that Sam (an actual player) is a player and the second line states that Sam has one of the three cards: hall, dagger or white. Notice that the second line is information that we have obtained while the game is in progress and we are dynamically asserting it into the ontology. One could then query the ontology with the following:

```
(retrieve (?x) (some (inv has_card) (one-of sam)))
```

The above query attempts to infer from the ontology exactly, which cards are being held by Sam. It shows how one could use reasoning to obtain information from an ontology. The example also exposed some limitations of DL systems in representing a dynamic system. In particular, DL ontologies are not designed to handle changes over time, it is merely a static representation.

1.4 Web Ontology Language (OWL)

The Web Ontology Language (OWL) is a family of knowledge representation languages endorsed by the W3C for representing ontologies. It is based on the Resource Description Framework (RDF) syntax, which means an OWL ontology is represented as a set of RDF triples. However, there is a clearly defined semantics to allow constructs to be interpreted

```
(implies player (=3 has_card card))

(implies envelope (=1 has_card room_card))
(implies envelope (=1 has_card suspect_card))
(implies envelope (=1 has_card weapon_card))

(equiv card (or suspect_card weapon_card room_card))

;; room cards
(instance ballroom room_card)
(instance billiard room_card)
(instance conservatory room_card)
(instance dining_room room_card)
(instance hall room_card)
(instance kitchen room_card)
(instance library room_card)
(instance lounge room_card)
(instance study room_card)

;; suspect cards
(instance black suspect_card)
(instance green suspect_card)
(instance mustard suspect_card)
(instance peacock suspect_card)
(instance plum suspect_card)
(instance scarlett suspect_card)
(instance white suspect_card)

;; weapon cards
(instance candlestick weapon_card)
(instance dagger weapon_card)
(instance pipe weapon_card)
(instance revolver weapon_card)
(instance rope weapon_card)
(instance spanner weapon_card)
```

Program 1: Representing Cluedo in KRSS

logically. The original OWL specification [BvHH+04] proposed three variants of the language with varying expressive power and computational properties. They are OWL Lite, OWL DL and OWL Full.

An OWL ontology contains class axioms: `owl:subClassOf`, `owl:equivalentClass` and `owl:disjointWith`. These axioms in turn make use of different types of class descriptions. However, not all axiom types or class descriptions are supported by all three variants of OWL. More specifically, OWL Lite supports only a small number of these constructs. OWL DL and OWL Full support most of these constructs but there are still differences between them.

One clear distinction is that OWL Full does not explicitly distinguish between classes and individuals in an ontology. This will not only impact the way an ontology is represented but also such representations are likely to incur additional computational costs.

It should be noted that OWL does not have the Unique Name Assumption (UNA), meaning that multiple syntactically different names or identifiers can be used to refer to the same instance and inference mechanisms are provided in OWL reasoners to infer hidden relationships between individuals (e.g. inferring that two individual names are referring to the same instance).

There are six types of class descriptions in OWL, they are `owl:Class`, `owl:Restriction`, `owl:oneOf`, `owl:intersectionOf`, `owl:unionOf` and `owl:complmentOf`.

OWL Lite is the least expressive of the three sub-languages. It is designed with simple constructs in mind to enable highly efficient computations. OWL Lite allows for restricted forms of class descriptions. In terms of concept descriptions, OWL Lite does not support `owl:oneOf`, `owl:unionOf` and `owl:complementOf`. However, it does support a restricted form of `owl:intersectionOf` that only allows to build intersections of class identifiers and property restrictions. In terms of axioms, OWL Lite does not permit the use of `owl:disjointWith`. It allows a restricted form `owl:subClassOf` and `equivalentClass`, where the subject of the constructor is required to be a class identifier, and the object is either a class identifier or a property restriction. The type of restrictions imposed on the syntax of OWL Lite has the implication that OWL Lite ontologies are always consistent.

The following table provides a summary of the constructs that are supported by the different sublanguages of OWL.

Constructor	**OWL Lite**	**OWL DL**	**OWL Full**
owl:Class	•	•	•
owl:Restriction	•	•	•
owl:oneOf	×	•	•
owl:intersectionOf	○[2]	•[2]	•
owl:unionOf	×	•	•
owl:complementOf	×	•	•
owl:allValuesFrom	○[3]	•	•
owl:someValuesFrom	○[3]	•	•
owl:hasValue	×	•	•
owl:maxCardinality	○[4]	•	•
owl:minCardinality	○[4]	•	•
owl:cardinality	○[4]	•	•
owl:subClassOf	○[5]	•	•
owl:equivalentClass	○[5]	•	•
owl:disjointWith	×	•	•
owl:ObjectProperty	•	•	•
owl:DatatypeProperty	•	•	•
owl:sameAs	•	•	•
owl:differentFrom	•	•	•
owl:AllDifferent	•	•	•
owl:InverseFunctionalProperty	•	•	•
owl:SymmetricProperty	•	•	•
rdfs:subPropertyOf	○	○	•
rdfs:domain	○[6]	•	•
rdfs:range	○[6]	•	•

• - supported, ○ - limited support, × - not supported

The OWL-DL fragment of the OWL standard has received much interest in the Semantic Web community. It corresponds to an interesting class of description logics that is known to perform decidable (but intractable) reasoning, and therefore provides a theoretical foundation for promising practical implementations, such as RacerPro [HM01b, HM01a], FaCT [Hor98], FaCT++ [TH06] and Pellet [SPG+07].

Extensions of OWL

There have been two extensions of the Web Ontology Language (OWL) since the original version of OWL 1.0.

Both of them are motivated by practical uses in the Semantic Web community and in particular, they were carefully chosen to suit the needs for certain applications. One is an extension of OWL DL known as OWL 1.1. OWL 1.1 supports the description logic $\mathcal{SROIQ}$ that is an extension of the well known $\mathcal{SHOIQ}$ [HS07] with the following features: disjoint roles, reflexive and irreflexive roles, negated role assertions, complex role inclusion axioms, universal role, local reflexivity of roles.

These features are known to be useful in practise (e.g. medical ontologies) and do not incur additional computational costs. In particular, they do not introduce additional non-determinisms to the reasoning algorithm. Details of the description logic $\mathcal{SROIQ}$ are discussed in 2.3.3. The development of OWL 2.0 [MPSP+08] has led to three additional sublanguages: OWL 2 EL, OWL 2 QL and OWL 2 RL. However, these languages are beyond the scope of this dissertation.

[1] Only intersections of class identifiers or property restrictions.
[2] Only class identifier is allowed as object of this construct.
[3] Only used with values 0 or 1.
[4] Subject of the construct must be a class identifier and object must be a concept identifier or a property restriction.
[5] Value of the construct must be a class identifier.

1.5 Non-standard Reasoning Services

1.5.1 Ontology Alignment

Ontology Alignment is one of the major challenges in ontology modification. The setting is as follows: Two or more agents have different ontologies relating to the same domain. Their purpose is to be able to communicate. What is needed is a method of translation from one ontology to the other.

This is a difficult yet important challenge as solving this problem essentially enables effective communications between machines. The main difficulty of this alignment problem is that the terminologies (ontologies) of the two agents can be substantially different, even though they are describing the same domain. There are several sources of heterogeneity that could come into play and they can vaguely be categorise into syntactic and semantic differences. The most obvious ones, and perhaps the easiest to solve, are the terminologies with syntactic differences but with same or similar meanings. For this class of alignment problems, we could turn to classical Natural Language Processing literature for solutions. Most of the existing approaches rely on normalisation of concept names in terminologies so that syntactically similar concept names can be identified and aligned. This approach, however, does not resolve the more critical problem of semantic differences that often present in real-world ontologies. In particular, real-world ontologies are often constructed directly by human or through conversion from some existing structured texts. This means that these ontologies are constructed for different purposes and the granularity or description of (even the same) concepts can be different. For example, people's perceptions of a dog can vary due to their background and knowledge. A typical person might describe it as an animal with four legs but a biologist would have a completely different notion, even though they are referring to the same class of objects. Therefore, the first question one should ask before committing to an alignment is whether it makes sense to align two ontologies, or more specifically, how does one measure semantic differences between ontology concepts.

There has been some work [YK03] in making use of Channel Theory to perform ontology alignment, in which a method for automatically determining such ontology

translation procedures is defined, under the assumption that all agents share a set of basic primitives on which all other terms are based. In description logics this would amount to the alignment of concepts defined in the TBox. Therefore, if we are to perform ontology modification, one would probably give higher priorities to the TBox than the ABox.

1.5.2 Ontology Merging

It is possible to identify different types of ontology merging. Merging TBoxes amounts to making sure that different agents use the same terms in identical ways. The goal here is quite unlike that of classical belief merging [KPP98]. In belief merging, information from different sources are pooled together and the requirement is to find a consistent set of beliefs representing the merged information. In the case of merging TBoxes, it seems that either one of the two things could happen. If agents find that they use the same terms in ways that differ only slightly, it would be required from all involved agents to amend their definitions slightly. This is indeed reminiscent of classical belief merging. The other possibility is for them to realise that although they are using the same term(s), they are really describing sufficiently different concepts. In such a case it would be necessary for both agents to invent new terms to describe the concept of the other one. Perhaps a combination of alignment and merging would be appropriate.

Observe that TBox merging ought not to affect the ABoxes, although it might affect the conclusions that can be drawn, from assertions in the ABoxes. Suppose that a concept in the TBoxes of agents 1 and 2 has been modified slightly; that is, merging has taken place. For example, suppose agent 1's concept of fish in its TBox is of an animal with fins that lives in water, and that its ABox contains the information that Willy is an animal, has fins, lives underwater, and suckles its young. It will then be able to conclude that Willy is a fish. But if, after merging, agent 1's concept of a fish is changed to an animal with fins, living in water, not suckling its young, Willy would no longer be classified as a fish.

A formalisation of TBox merging looks quite different from that of classical merging. For example, one of the basic properties of belief merging is that if all the pieces of

information to be merged put together yield a consistent set of sentences, this is what the merged outcome should be. But consider a situation in which agent 1 defines a dog as an animal with four legs that barks, and agent 2 defines a dog as a *large* animal with four legs, that barks. Although these two definitions are consistent, it is unintuitive to take the merged outcome to be these two definitions put together (yielding a definition of a dog as a large animal with four legs that barks). Agent 1 might well reason that agent 2's definition is based on the fact that it has not yet encountered small dogs.

The other type of merging is ABox merging. This is merging in the classical style, although one has to make the assumption that TBoxes have already been aligned and merged before ABox merging takes place.

1.5.3 Ontology Revision

One view of ABox revision is that it is essentially classical revision, but with the provision that everything in the TBox is fixed. Formally, this can be described as classical AGM revision, for example, with sentences in the TBox treated similarly to logically valid sentences. However, there exists another view of ABox revision. Suppose that Agent 1's definition of a bird is of an animal with feathers. Furthermore, suppose that Agents 1's TBox contains the fact that all birds fly. Now suppose further that I have observed that Agent 1 has observed that Tweety is an animal with feathers, and Agent 1 has been told that Tweety is a bird. Now Agent 1 observed that Tweety cannot fly. This thus creates an inconsistency with information in TBox. One solution to this, of course, is to remove the information that Tweety is a bird.

An even more natural solution would seem to be to modify the ABox assertion so that it is no longer inconsistent with the TBox. For example, asserting that Tweety is an animal instead of a bird. At the moment it is still unclear as to when to perform the first kind of ABox revision, and when it would be more appropriate to perform the second kind.

1.6 Ontology Tools

Ontology Editors

There are a number of open-source ontology editors, including Protege [NFM00], Swoop [KPS+06], OilEd [BHGS01] and Hozo [KKIM02], as well as commercial ones such as RacerPorter [HMW03].

The main functions of these ontology editors are to facilitate the construction of ontologies in a user-friendly environment.

In particular, since OWL is not designed to be read or written directly but serve as a representation to store ontologies, the ontology editors parse this representation and presents useful information about the ontologies. For example, they show detailed information about classes, properties and instances.

In addition, most ontology editors support reasoning over ontologies. This is usually achieved by either connecting to a remote server where an ontology reasoner resides or through access to a locally running ontology reasoner. For the former, there is a protocol DIG that is used by a number of ontology reasoners to transfer and query ontologies over a TCP/IP connection. However, the DIG protocol in its current state has many problems. Specifically, it does not capture some of the constructors in OWL adequately. An example is the lack of datatype support in DIG 1.0 and 1.1. There are a number of ontology reasoners that have adopted DIG, including CEL, FaCT++ [TH06], Pellet [SPG+07] and RacerPro [HM01b]. For the latter, reasoners are simply running locally in the machine where the ontology editor resides. In many cases, this is a more efficient solution since data is no longer required to be sent over a TCP/IP connection, thus avoiding transfer overhead. Ontology editors can simply query a reasoner through a reasoning interface. For example, JRacer can be used to access RacerPro in Java. Reasoners can also be accessed through general reasoning interfaces, such as the OWL API and Jena.

In addition to standard reasoning services, some ontology editors also provide so-called non-standard reasoning services, such as axiom pinpointing. These additional reasoning services intend to provide additional support to enhance ontology construction,

maintenance and reuse. The ontology editor SWOOP supports axiom pinpointing.

OWL Reasoners

There are a number of highly optimised OWL reasoners, including RacerPro [HM01b], FaCT++ [TH06] and Pellet [SPG+07]. Most of these are based on the tableau-based algorithm. There is also a probabilistic description logic reasoner called Pronto [Kli08]. Bock et al. presented a benchmarking of the popular OWL reasoners [BHJV08].

Reasoner	**Reasoning Type**	**Expressivity**	**OWL**
RacerPro [HM01b]	Tableau	$\mathcal{SHIQ}(\mathcal{D})$	OWL DL
FaCT++ [TH06]	Tableau	$\mathcal{SROIQ}(\mathcal{D})$	OWL DL
Pellet [SPG+07]	Tableau	$\mathcal{SROIQ}(\mathcal{D})$	OWL DL
KAON2 [Mot08]	Resolution	$\mathcal{SHIQ}(\mathcal{D})$	OWL DL
HermiT [SMH08]	Hyper-tableau	$\mathcal{SHOIQ}^{+}$	OWL DL
CEL	Structural	$\mathcal{EL}^{++}$	OWL EL

1.7 Research Contributions

Our main research contribution is the investigation of three important operations concerning the management of logical inconsistencies in description logics (DL). The three operations are ontology contraction, ontology integration and ontology debugging, they are presented in Chapter 3, Chapter 4 and Chapter 5 respectively.

In Chapter 3, we consider the problem of ontology contraction where a set of DL sentences A is contracted by a single sentence α. We study the AGM method of partial meet contraction in the DL setting, and show by example that it can be directly applicable to DL, but may lead to counter-intuitive results. We address issues in applying AGM methods in the context of DL, which we discussed in [LM04]. We then present the notion of *exception*, which we introduced in [MLB05]. We then present the notion of remainder set for DL as a refinement of the classical notion of remainder set. We use our notion of remainder set for DL to produce a partial meet contraction operator for DL and argue that it retains more information than that of classical partial meet

contraction. We then show that our partial meet contraction operator for DL does not satisfy the classical AGM contraction postulates, and we present a refinement of these postulates that our partial meet contraction operator satisfies. Moreover, we extend the notion of multiple contraction into the DL setting, which involves contraction of a set of sentences A by another set of sentences B instead of a single sentence. We present the notions of package and choice contractions for DL and show that they satisfy our refined sets of postulates.

In Chapter 4, we address the issue of ontology integration, where a stratified set of sentences is to be integrated into a single set of sentences. We consider a class of propositional knowledge integration techniques called *adjustment* and show that adjustment is *not* directly applicable to DL. We show that there are limitations to the expressive power of DL and in particular disjunctions of DL sentences cannot be freely expressed, which we published in [MLB05]. We then introduce the notion of disjunctive knowledge base [MLB05], which provides a way to express the result of integrating a stratified DL knowledge base. We present a new version of adjustment called Conjunctive Maxi-Adjustment (CMA), which we introduced in [MLB05], as a variant of a propositional adjustment strategy known as (whole) Disjunctive Maxi-Adjustment (DMA). We present an adaptation of the CMA strategy into the DL context called CMA-DL and show that it is applicable to DLs. Yet this approach does not adequately exploit the structure of DL sentences. We address this issue by introducing a refined version of CMA-DL called RCMA-DL, which makes use of the notion of exception we introduced in [MLB05]. We show that RCMA-DL presents results that retains more information than that of CMA-DL. Moreover, we define the semantics of CMA-DL and RCMA-DL based on the notion of lexicographic ordering on DL interpretations and we establish the connection of lexicographic ordering on DL interpretations with lexicographic entailment of both CMA-DL and RCMA-DL.

In Chapter 5, we study a method for restoring consistency of an inconsistent ontology. Specifically, we look at a tableau-based algorithm, which was first introduced by Baader et al. in [BH95] and later studied by [Sch05b] and by us [MLBP06]. We show by example that the labelled consistency algorithm with classical subset blocking does not

guarantee completeness in the presence of cyclic definitions, as we argued in [LMPB06]. We then introduce a refined version of subset blocking that takes the labels of sentences into account and show that this refined version guarantees completeness. We extend the notion of label blocking into more expressive description logics and present both a refined version of equivalence blocking and a refined version of pair-wise blocking. Furthermore, we argue that using propositional formulas as a way to retain a trace in the labelled consistency algorithm has many advantages. In particular, we show that propositional formulas can be compiled into Reduced Ordered Binary Decision Diagrams (ROBDDs), which can be used as a diagrammatic tool to repair an ontology.

1.8 Structure of the Dissertation

This dissertation is organised as follows: Chapter 1 specified the problem of inconsistency management in description logics. We provide background to the area of Semantic Web and ontologies, and their connection with description logics. We also introduced some existing semantic technologies. Chapter 2 reviews material in description logics. We introduce the syntax and semantics of description logics. We describe the various types of reasoning tasks offered by description logics and how these are realised through reasoning algorithms. Chapter 3 presents ontology change as operations acting upon ontologies. We focus on the semantic characterisation of change in description logics. Chapter 4 is pertinent to the problem of ontology integration where multiple ontologies are to be integrated to form a single coherent ontology. We establish the connection of this problem with a similar problem in propositional logic and show how the existing solutions in propositional logic can be employed into the description logic context. Chapter 5 extends the work from the previous chapters to provide a tableau-based algorithm for repairing inconsistent ontology. Chapter 6 closes this dissertation with a summary of the results and contributions.

Chapter 2

Description Logics

Description Logic (DL) plays a foundational role in the development of the Semantic Web. Perhaps the most significant contribution of DL is that it has brought forth a spectrum of formal languages with serious considerations of their expressive power and computational properties, making concrete what ontologies are and precisely what is achievable with them. Research in DL has also given rise to a wide range of semantic technologies, including ontology standards, frameworks, APIs and tools. In particular, it has facilitated the development of the Web Ontology Language (OWL) [BvHH+04] and its successors OWL 1.1 [MPSH07] and OWL 2.0 [MPSP+08, GHM+08], establishing a W3C recommendation for specifying ontologies and allowing them to be shared and distributed across the web. It has also led to the implementation of ontology editors that support the construction and maintenance of ontologies, including Protege, RacerPorter [HMW03], Swoop, Hozo [KKIM02] and OilEd. It has resulted in many highly optimised DL reasoners that are able to support standard reasoning services such as consistency and satisfiability checking and subsumption tree generation, as well as non-standard reasoning services such as ontology debugging, alignment, mapping and integration. Many of these technologies have been deployed in various domains and real-world application areas.

In this chapter, we provide an overview of description logics. This chapter is organised in three sections. The first section gives an introduction to the family of description logics. In particular, we make clear the connection between the Semantic Web and

description logics, and explain why description logics are appropriate for representing ontologies. We focus our study on the classical description logic $\mathcal{ALC}$ [SSS91] which is the foundation to many well-known description logics (e.g. $\mathcal{SHOIN}$ [HS99], $\mathcal{SHIQ}$ and $\mathcal{SROIQ}$) that are used in practise. The second section is devoted to a discussion of reasoning algorithms in description logics. These reasoning algorithms enable description logics to provide the various types of reasoning services. We look at the most popular reasoning algorithm in DL known as tableau algorithm, which is deployed in many of the state-of-the-art reasoners. These reasoners are known to produce promising computational performance [Tob01, HST00]. The third section provides a summary of other description logics, including the class of DLs known as expressive description logics, such as $\mathcal{SHIQ}$, $\mathcal{SHOIN}$ and $\mathcal{SROIQ}$. This class of logics is highly expressive but are still known to be in the decidable fragments of first-order logic. In addition, we also look at description logics that are less expressive than $\mathcal{ALC}$ [SSS91] such as $\mathcal{EL}$ and $\mathcal{EL}^+$. These logics are known to allow tractable reasoning [BLS06] and perform exceptionally well in certain application areas.

2.1 Description Logic Knowledge Bases

Description Logics are a class of knowledge representation languages with formal syntax and semantics. They are decidable fragments of first-order logic carefully chosen to have adequate expressive power but also possess desirable computational properties, such as soundness and completeness of reasoning.

A typical DL knowledge base $K = \langle \mathcal{T}, \mathcal{A}, \mathcal{R} \rangle$ comprises three finite and mutually disjoint sets: a set of concept axioms (or terminologies) $\mathcal{T}$, a set of assertions $\mathcal{A}$ and a set of role properties [1] $\mathcal{R}$, where $\mathcal{T}$, $\mathcal{A}$ and $\mathcal{R}$ are also known as TBox, ABox and RBox respectively. Concept axioms describe relationships between concepts. Some common concept axioms are inclusion axioms and equality axioms. Assertions describe individuals in the knowledge base and how they relate to concepts and roles. Common assertions

[1] Role properties were called role assertions in [HKS06]. However, role assertions are traditionally used in the literature to describe DL sentences of the form $R(s,t)$. These were called individual assertions in [HKS06].

are role assertions and concept assertions. It should be noted that description logic knowledge bases do not traditionally contain an RBox, although role properties such as transitivity of roles are supported in DLs such as $\mathcal{SHIQ}$ and $\mathcal{SHOIN}$. Description logics knowledge bases in $\mathcal{RIQ}$ and $\mathcal{SROIQ}$ are known to contain RBoxes. These logics support additional role properties such as reflexivity, irreflexivity and disjointness. For convenience, we simply assume that the RBox of a knowledge base is an empty set for cases where role properties are not supported in the logic.

2.1.1 The Syntax

Each description language is defined by a set of grammar rules which governs the types of axioms, assertions, concept descriptions and role descriptions that can be used in the language. The precise syntax can be expressed as a context-free grammar in Backus-Naur Form (BNF). For example, the classical description logic $\mathcal{ALC}$ [SSS91] allows terminologies in the following syntactic forms:

$$\begin{aligned} \langle concept\_axiom \rangle &::= \langle inclusion\_axiom \rangle \mid \langle equality\_axiom \rangle \\ \langle inclusion\_axiom \rangle &::= \langle concept \rangle \text{“ } \sqsubseteq \text{ ”} \langle concept \rangle \\ \langle equality\_axiom \rangle &::= \langle concept \rangle \text{“ } \doteq \text{ ”} \langle concept \rangle \end{aligned}$$

The rules above define a concept axiom to be either an inclusion axiom (also called subsumption axiom) or an equality axiom. The left-hand side and right-hand side of the subsumption symbol ($\sqsubseteq$) and the equality symbol ($\doteq$) are concept descriptions. In $\mathcal{ALC}$ [SSS91], the syntax of the concept and role descriptions are inductively defined as follows:

$$
\begin{aligned}
\langle concept \rangle ::= \quad & \text{“(”} \langle concept \rangle \text{“} \sqcap \text{”} \langle concept \rangle \text{“)”} \quad | \\
& \text{“(”} \langle concept \rangle \text{“} \sqcup \text{”} \langle concept \rangle \text{“)”} \quad | \\
& \text{“}\exists\text{”} \langle role \rangle \text{“.”} \langle concept \rangle \quad | \\
& \text{“}\forall\text{”} \langle role \rangle \text{“.”} \langle concept \rangle \quad | \\
& \text{“}\neg\text{”} \langle concept \rangle \quad | \\
& \langle concept\_name \rangle \quad | \quad \text{“}\top\text{”} \quad | \quad \text{“}\bot\text{”} \\
\langle role \rangle ::= \quad & \langle role\_name \rangle
\end{aligned}
$$

The non-terminals $\langle concept\_name \rangle$ and $\langle role\_name \rangle$ are assumed to be defined by some terminal expressions. If the left-hand side of an equality axiom is a concept name then it is also called a concept definition. It is important to identify concept definitions because they often constitute a large portion of real-world ontologies and they have the nice property of being $unfoldable$. As we shall see in later sections, unfoldable terminologies are highly optimised for reasoning using a technique called absorption that avoids unnecessary non-deterministic branching. A TBox $\mathcal{T}$ is *unfoldable* if and only if the left-hand side of every terminology $\tau \in \mathcal{T}$ contains a concept name A, that there are no other τs with A on the left-hand side, and that the right-hand side of τ contains no direct or indirect references to A (i.e. there are no cycles). The syntactic forms of assertions are similarly defined as follows. An assertion can be either a concept assertion or a role assertion. Again, the non-terminal $\langle individual\_name \rangle$ is assumed to be defined by a terminal expression.

$$
\begin{aligned}
\langle assertion \rangle ::= \quad & \langle concept\_assertion \rangle \quad | \quad \langle role\_assertion \rangle \\
\langle concept\_assertion \rangle ::= \quad & \langle concept \rangle \text{“(”} \langle individual\_name \rangle \text{“)”} \\
\langle role\_assertion \rangle ::= \quad & \langle role \rangle \text{“(”} \langle individual\_name \rangle \text{“,”} \langle individual\_name \rangle \text{“)”}
\end{aligned}
$$

The syntactic rules above completely define the grammar of the $\mathcal{ALC}$ [SSS91] language. It should be noted that there are no syntactic rules for RBox elements, because the $\mathcal{ALC}$ language does not support role axioms. This (as mentioned before), we simply take the RBox as an empty set. Below is an example of a simple knowledge base in $\mathcal{ALC}$.

Example 2.1

$$\mathcal{T}\begin{cases} Penguin \sqsubseteq Bird \\ Bird \doteq Animal \sqcap Fly \end{cases}$$

$$\mathcal{A}\begin{cases} Penguin(tweety) \\ hasChild(tweety, chirpy) \end{cases}$$

Consider a DL knowledge base $K = \langle \mathcal{T}, \mathcal{A} \rangle$, where $\mathcal{T}$ and $\mathcal{A}$ are the sentences stated above. This ontology makes references to four concepts, named `Penguin`, `Bird`, `Animal` and `Fly`; one role `hasChild`; and two individuals `tweety` and `chirpy`. There are two concept axioms: the first axiom states that every penguin is a bird and the second axiom *defines* a bird as being an animal that flies. Notice that we have used a subsumption for the first axiom but an equivalence for the second. The reason is that we want to describe a bird as the necessary condition to be a penguin but not the other way around. That is, a bird is *not* necessarily a penguin. In the second axiom, an equivalence is used instead because we are describing a necessary and sufficient condition. That is, it is not only that every bird is an animal that flies, but also every animal that flies is a bird. There are also two assertions in the knowledge base: a concept assertion and a role assertion. The former describes an individual tweety as being a penguin and the latter describes the relationship between two individuals, that tweety has a child named chirpy.

Thus far we have only stated syntactic rules to write description languages. This provides us a way to check the syntactic validity of statements in the language, but it does not tell us how to interpret them. For example, we have mentioned earlier that $\sqsubseteq$ and $\doteq$ are two different relations. One expresses a necessary condition, while the other expresses both a necessary and sufficient condition. However, we have not yet defined any rule that enforces this. In fact, $\sqsubseteq$ and $\doteq$ are merely symbols in a (description) language, and one could interpret them in exactly the same way. As we shall see in the next section, description languages have formally defined semantics. Symbols in a description language, such as concept names and operators, are interpreted in a systematic way and through this they are given a precise meaning.

2.1.2 The Semantics

The formal semantics of description languages is provided through means of interpretations. An interpretation $\mathcal{I}$ is a 2-tuple $\langle \Delta^{\mathcal{I}}, .^{\mathcal{I}} \rangle$ where $\Delta^{\mathcal{I}}$ is a non-empty (possibly infinite) set that denotes the domain of $\mathcal{I}$, and $.^{\mathcal{I}}$ is a function that maps every individual s to an element $s^{\mathcal{I}} \in \Delta^{\mathcal{I}}$, every concept C to a subset $C^{\mathcal{I}}$ of $\Delta^{\mathcal{I}}$, and every role R to a subset $R^{\mathcal{I}}$ of $\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$. [2] In addition, there are two constants $\top$ and $\bot$. The top concept $\top$ maps to the set $\top^{\mathcal{I}} = \Delta^{\mathcal{I}}$, and the bottom concept $\bot$ maps to the set $\bot^{\mathcal{I}} = \emptyset$.

The definition of the interpretation function $.^{\mathcal{I}}$ is extended to concept and role descriptions. It can be seen as a set of constraints imposed onto the interpretation. For $\mathcal{ALC}$, the interpretation $\mathcal{I}$ satisfies the following conditions:

$$(\neg C)^{\mathcal{I}} = \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}} \tag{2.1}$$

$$(C \sqcap D)^{\mathcal{I}} = C^{\mathcal{I}} \cap D^{\mathcal{I}} \tag{2.2}$$

$$(C \sqcup D)^{\mathcal{I}} = C^{\mathcal{I}} \cup D^{\mathcal{I}} \tag{2.3}$$

$$(\exists R.C)^{\mathcal{I}} = \{x \mid \exists y(\langle x, y \rangle \in R^{\mathcal{I}} \wedge y \in C^{\mathcal{I}})\} \tag{2.4}$$

$$(\forall R.C)^{\mathcal{I}} = \{x \mid \forall y(\langle x, y \rangle \in R^{\mathcal{I}} \rightarrow y \in C^{\mathcal{I}})\} \tag{2.5}$$

In $\mathcal{ALC}$ and all description languages that extend it, it is required that Conditions (2.1)-(2.5) are satisfied. For description logics that support additional constructs such as inverse roles and transitive roles, there are additional conditions that an interpretation must satisfy. Details of more expressive description logics are in Section 2.3.

Definition 2.2 (DL Model)

An interpretation $\mathcal{I}$ is a model of a knowledge base $K = \langle \mathcal{T}, \mathcal{A} \rangle$ if and only if $\mathcal{I}$ is a model of $\mathcal{T}$ and $\mathcal{A}$. An interpretation $\mathcal{I}$ is a model of a TBox $\mathcal{T}$ if and only if it satisfies Conditions 2.6 and 2.7. An interpretation $\mathcal{I}$ is a model of an ABox $\mathcal{A}$ if and only if it

[2] $X \times Y$ is the Cartesian product of the sets X and Y, defined as $\{\langle x, y \rangle \mid x \in X \text{ and } y \in Y\}$

satisfies Conditions 2.8 and 2.9.

$$(C \sqsubseteq D) \in \mathcal{T} \Longrightarrow C^{\mathcal{I}} \subseteq D^{\mathcal{I}} \tag{2.6}$$

$$(C \doteq D) \in \mathcal{T} \Longrightarrow C^{\mathcal{I}} = D^{\mathcal{I}} \tag{2.7}$$

$$C(x) \in \mathcal{A} \Longrightarrow x^{\mathcal{I}} \in C^{\mathcal{I}} \tag{2.8}$$

$$R(x, y) \in \mathcal{A} \Longrightarrow \langle x, y \rangle^{\mathcal{I}} \in R^{\mathcal{I}} \tag{2.9}$$

Note that C, D are concept descriptions, R is a role name and x, y are individual names.

Logical Integrity

The notion of model from Definition 2.2 can be extended to the notion of logical integrity. It should be noted that coherence in Definition 2.4 applies to a TBox whereas consistency in Definition 2.5 applies to a DL knowledge base (i.e. the combination of a TBox and an ABox). The three notions in Definition 2.3-2.5 form the basis for all the other reasoning tasks available in DL. In fact, most existing reasoners are focused towards the optimisation of satisfiability checking. All other reasoning tasks are essentially reduced to it.

Definition 2.3 (Concept Satisfiability)
A concept C is satisfiable *with respect to a TBox $\mathcal{T}$ if and only if there is a model $\mathcal{I}$ of $\mathcal{T}$ such that $C^{\mathcal{I}} \neq \emptyset$.*

Definition 2.4 (Coherence)
A TBox $\mathcal{T}$ is coherent *if and only if A is satisfiable with respect to $\mathcal{T}$ for all concept names A appearing in $\mathcal{T}$.*

Definition 2.5 (Consistency)
A DL knowledge base K is consistent *if and only there exists a model $\mathcal{I}$ of K.*

Example 2.6
Consider Example 2.1 again and suppose we have the interpretation $\mathcal{I}$ as shown below. Recall that an interpretation consists of a set $\Delta^{\mathcal{I}}$ and a mapping function $.^{\mathcal{I}}$. In this

example, we have $\Delta^{\mathcal{I}} = \{tweety, chirpy, bob\}$ and the following defines the mapping function $\cdot^{\mathcal{I}}$:

$$\begin{aligned}
\top^{\mathcal{I}} &= \{tweety, chirpy, bob\} \\
Bird^{\mathcal{I}} &= \{tweety, chirpy\} \\
Animal^{\mathcal{I}} &= \{tweety, chirpy\} \\
Fly^{\mathcal{I}} &= \{tweety, chirpy\} \\
Penguin^{\mathcal{I}} &= \{tweety\} \\
hasChild^{\mathcal{I}} &= \{\langle tweety, chirpy\rangle\} \\
tweety^{\mathcal{I}} &= tweety \\
chirpy^{\mathcal{I}} &= chirpy \\
(Animal \sqcap Fly)^{\mathcal{I}} &= \{tweety, chirpy\} \\
&\dots
\end{aligned}$$

$\mathcal{I}$ is an interpretation for $\mathcal{ALC}$ because it satisfies Conditions (2.1)-(2.5). It is also a model of the knowledge base defined in Definition 2.2 because it satisfies Conditions (2.6)-(2.9). Notice that $\mathcal{I}$ is a model even though the individual bob in $\Delta^{\mathcal{I}}$ does not appear anywhere in the knowledge base.

Unique Name Assumption (UNA)

The Unique name Assumption (UNA) enforces that symbols in a language that are different are necessarily referring to different objects in the domain. Description logics do not employ the UNA, concept (also role and individual) names can be syntactically different but still be referring to the same objects in the domain. For example, we could construct an interpretation $\mathcal{I}$ such that $\Delta^{\mathcal{I}} = \{alice, bob\}$, and have $bob^{\mathcal{I}} = bob$ and $bobby^{\mathcal{I}} = bob$. In this case, the names Bob and Bobby are referring to the same single object (the person Bob) in the domain, where Bob could be the actual name of the person and Bobby is a nickname. This kind of scenarios often occur in real-world applications [BBB+09]. Systems that do not adopt the UNA can be seen as being more

general than systems that adopt the UNA. This is because without the UNA, one could assert sentences to explicitly state that two individuals are (necessarily) equivalent or that two individuals are *not* equivalent. The Web Ontology Language (OWL) also does not enforce the UNA instead it allows the use of `owl:sameAs` and `owl:differentFrom` to make explicit whether two individuals are referring to the same objects in the domain or not. If none of these are specified and no additional information can be inferred then it is assumed to be unknown. Some reasoners, such as RacerPro, allows the user to specify whether to employ the UNA or not in the reasoning. Employing the UNA will usually incur additional computational costs.

Open and Closed World Semantics

An important distinction between a (relational) database and a DL knowledge base is that the former is based on the Closed World Assumption (CWA) while the latter is based on the Open World Assumption (OWA). To illustrate the differences between the two notions, consider the sentence "Bob is a student". In the case of a relational database, this sentence can be represented as a row in a database table and for a DL knowledge base, it can be represented as a concept assertion `Student(bob)`. We can query both the database and the knowledge base, and both would return the correct output (i.e., "Bob is a student"). In this scenario, there is no difference between the two systems. The difference lies in cases where information is in absence. Suppose the row that encodes the above sentence is not present in a relational database. That is the sentence is *not* asserted true explicitly.

Since relational databases adopts the CWA, a sentence that is not asserted true implies that its negation is true or simply that the sentence is false. Therefore, it concludes that Bob is *not* a student. This is different from a DL knowledge base that adopts the OWA, where absence of information is regarded as unknown, without passing judgement on its truth value. The fact that `Student(bob)` is not asserted true means it could either be the case that Bob is not a student *or* that Bob is in fact a student. There is simply insufficient information in the knowledge base to make a conclusion. In some sense, the OWA enables reasoning in a knowledge base because it allows information to be inferred

from different fragments of the knowledge base.

Another way to view CWA and OWA is that the CWA can be seen as a version of two-valued logics where propositions are assigned one of the two truth values: true or false. On the other hand, the OWA can be regarded as a three-valued logic where sentences are assigned with one of the three truth values: true, false or unknown. There are advantages and disadvantages to adopting either of these approaches but the choice of choosing one is dependent on the application scenario. Roughly speaking, systems that adopt the OWA are generally more suitable in scenarios where complete information is not available. For example, the problem of Situation Awareness (SA) in the military domain as demonstrated in [BBB+09] where sensor information is not necessarily complete due to failures in sensing equipment, interference or equipment limitations.

2.2 Reasoning in Description Logics

The ability to make inferences is a prominent feature in DL systems. It allows implicit knowledge to be inferred based on the explicitly asserted knowledge. In Section 2.1.2, we have defined some of the standard reasoning tasks that can be performed in description logics, including satisfiability and consistency checking. In the subsections ahead, we will explain how these reasoning tasks are realised using reasoning algorithms. We will also give an overview of other reasoning services that are provided by DLs including instance and subsumption checking, and the generation of subsumption hierarchy.

2.2.1 Reasoning Services

The various types of reasoning services are summarised in the table below.

Reasoning Task	**Definition**
Satisfiability	$C^{\mathcal{I}} \neq \emptyset$ for some model $\mathcal{I}$s of $\mathcal{T}$
Subsumption	$C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ for all model $\mathcal{I}$s of $\mathcal{T}$
Equivalence	$C^{\mathcal{I}} = D^{\mathcal{I}}$ for all model $\mathcal{I}$ of $\mathcal{T}$s
Disjointness	$C^{\mathcal{I}} \neq D^{\mathcal{I}}$ for all model $\mathcal{I}$s of $\mathcal{T}$
Instance checking	$s^{\mathcal{I}} \in C^{\mathcal{I}}$ for all model of $\mathcal{I}$ of $\mathcal{A}$

2.2.2 Constructing a Subsumption Hierarchy

The construction of a subsumption hierarchy is one of the most important reasoning tasks in description logics. A subsumption hierarchy, also known as a classification tree, is a partially ordered set over the set of concept names appearing in an ontology.

The construction of a subsumption hierarchy requires determining subsumption relationships between all ordered pairs of concept names. For example, suppose C, D, E are concept names that appear in the axioms of an ontology, then a subsumption hierarchy will tell us the subsumption relationships between these concepts (e.g., $C \sqsubseteq D$ and $D \sqsubseteq E$). Note that, subsumption is a transitive relation so it is necessary that $C \sqsubseteq E$ in the previous example. Therefore, an important question is how we could minimise the number of subsumption checks (a standard query in most DL reasoners) in order to induce the whole subsumption hierarchy, and this question relates closely to the classical problem of computing a partially ordered set. We will describe below in some detail how to compute a partially ordered set, but we will first define the properties of a subsumption relation.

Definition 2.7

The subsumption relation ($\sqsubseteq$) is a partial order over a set of concept names. Being a partial order, it satisfies the following properties:

(Reflexivity) $C \sqsubseteq C$, *for all* C.

(Antisymmetry) *if* $C \sqsubseteq D$ *and* $D \sqsubseteq C$ *then* $C \doteq D$ *for all* C, D.

(Transitivity) *if* $C \sqsubseteq D$ *and* $D \sqsubseteq E$ *then* $C \sqsubseteq E$ *for all* C, D, E.

Computing a partially ordered set

The problem of computing a partially ordered set (poset) was first explored by Faigle et al. [FT88], where they have devised two algorithms for sorting posets and have shown a query complexity of $O(wn\ log\frac{n}{w})$, where n and w corresponds to the number of elements and the width of the poset respectively. These algorithms assume that there is an oracle that answers queries and provides information about the ordering of the elements.

They also assume that the computational cost for querying an oracle is expensive and that all queries are of equal computational cost. Therefore the aim is to minimise the number of queries. However, this problem is not completely identical to the problem of constructing a subsumption hierarchy. It is similar in the sense that one could consider each subsumption check as a query to an oracle. Minimising the number of subsumption checks to some extent minimises the overall computation cost. However, the cost of each subsumption check is not necessarily the same and in many cases they vary. Therefore minimising the number of subsumption checks does not necessarily minimises the overall computational cost.

For any ordered pair of concepts (C, D), we could determine their subsumption relationship by performing a subsumption check. For an ontology $\mathcal{O}$, subsumption checking is a function $Sub_{\mathcal{O}} : \mathcal{L} \times \mathcal{L} \rightarrow B$ where $\mathcal{L}$ is the set of all concept names appearing in $\mathcal{O}$ and B is the boolean domain $\{true, false\}$.

	$\boldsymbol{C \sqsubseteq D}$	$\boldsymbol{C \not\sqsubseteq D}$
$\boldsymbol{D \sqsubseteq C}$	$C \equiv D$	$D \sqsubset C$
$\boldsymbol{D \not\sqsubseteq C}$	$C \sqsubset D$	$D \nsim C$

Figure 2.1: Subsumption relationships of two concepts C, D

By performing subsumption checks on a pair of concepts C, D, one could derive the following four properties: (1) If $C \sqsubseteq D$ and $D \sqsubseteq C$ then C and D are equivalent, (2) if $C \sqsubseteq D$ and $D \not\sqsubseteq C$ then $C \sqsubset D$, (3) if $C \not\sqsubseteq D$ and $D \sqsubseteq C$ then $D \sqsubset C$, (4) if $C \not\sqsubseteq D$ and $D \not\sqsubseteq C$ then C and D are incomparable (i.e. $C \nsim D$). This is summarised in Fig. 2.1

For less expressive description languages that do not express negations, such as the $\mathcal{EL}$ family, it is possible to apply structural subsumption algorithms to induce the subsumption hierarchy [BLS06]. However, in classical description languages such as $\mathcal{ALC}$ and other more expressive description languages, subsumption checking is normally reduced to satisfiability checking, as we have indicated in Section 2.2.1.

2.2.3 Reasoning Algorithms

In this section, we study a reasoning algorithm that has been used to realise the reasoning services described in Section 2.2.1. More specifically, we focus our attention to the problem of satisfiability checking for description logic $\mathcal{ALC}$ and its extensions. Among all reasoning algorithms, tableau-based algorithm is the most extensively studied in the literature and there exists highly optimised implementations. There are other reasoning algorithms that have been explored in the literature, including SAT-based algorithms [KH08].

Tableau Algorithms

The tableau algorithm can be seen as a search algorithm. It comprises of two parts: a search procedure that drives the construction of a search tree, and a set of expansion rules that determines what to do at each node in the search tree. The search procedure is not prescribed to any specific search strategy. However, it is typically implemented as a variant of depth-first search, because it is more memory conservative and easier to keep track of the search states than breadth-first search. Algorithm 1 shows a typical implementation of the search procedure.

It should be noted that Algorithm 1 is not fixed to a specific search strategy, but instead the choice of the queue used in the algorithm determines it. For example, a last-in-last-out queue results in a depth-first search, while a first-in-first-out queue results in a breadth-first-search. The behaviour of `Add` and `Remove` are determined by the type of queue used.

The `Expand` function is the core part of the algorithm. It encompasses a set of expansion rules as shown in Figure 2.2. For simplicity, we treat `Expand` as a blackbox and assumes that it takes exactly one ABox $\mathcal{A}$ as input and outputs either one ABox $\mathcal{A}'$, or two ABoxes $\mathcal{A}'$ and $\mathcal{A}''$.

The tableau algorithm takes as input a TBox *tbox* and a concept C. It outputs a boolean value: true if C is satisfiable with respect to *tbox* and false otherwise.

The procedure is initialised as follows: the queue *queue* as an empty list, the boolean variable $isSat$ as false, the initial ABox $\mathcal{A}_{init}$ as a set with only one concept assertion

```
input  : a TBox tbox, a concept C
output: true if C is satisfiable, otherwise false
initialise an empty queue queue;
isSat ← false;                                          /* set to false */
A_init ← {C(s)};                                        /* initialise ABox */
Add(queue, A_init);                                     /* initialise queue */
while true do                                           /* loops indefinitely */
    A_next ← Remove(queue);
    if A_next is not consistent then                    /* if not consistent */
        continue;
    end
    if A_next is expandable then                        /* if expandable */
        aboxes ← Expand(tbox,A_next);
        foreach A_new in aboxes do                      /* add new ABoxes to queue */
            Add(queue,A_new);
        end
    else
        isSat ← true;                                   /* set to true */
        break;
    end
end
return isSat;
```

Algorithm 1: A typical search algorithm for satisfiability checking

$\{C(s)\}$ where C is the concept to check for satisfiability and s is an anonymous individual. The initial ABox $\mathcal{A}_{init}$ is then added to *queue*. The main part of the procedure is a while-loop. In each iteration, an ABox $\mathcal{A}_{next}$ is removed from *queue* via `Remove`. After this, $\mathcal{A}_{next}$ is checked for consistency. If $\mathcal{A}_{next}$ is not consistent, it will go to the top of the while-loop. Otherwise, $\mathcal{A}_{next}$ is consistent and it will proceed to checking whether $\mathcal{A}_{next}$ is expandable. An ABox is expandable if it triggers at least one of the expansion rules in Figure 2.2. If $\mathcal{A}_{next}$ is expandable, then `Expand` is called to generate a set of ABoxes *aboxes*. Each of these new ABoxes $\mathcal{A}_{new}$ is an expansion of the preceding ABox $\mathcal{A}_{next}$, meaning that for each $\mathcal{A}_{new} \in aboxes$, $\mathcal{A}_{next} \subseteq \mathcal{A}_{new}$. All new aboxes are then added to the *queue*. If $\mathcal{A}_{next}$ is not expandable, then none of the expansion rules is applicable to $\mathcal{A}_{next}$. In this case, we have constructed a complete and clash-free ABox for C with respect to *tbox*, hence we simply set *isSat* to true and break out of the while-loop.

As we have mentioned earlier, the core part of the tableau algorithm is the `Expand`

function. It consists of a set of expansion rules that controls how the search tree is expanded. The expansion rules for $\mathcal{ALC}$ are shown in Fig. 2.2. The notation we use here is one adopted in [BCM$^+$03]. The intuitive idea is that we start off with an ABox and the expansion rules are treated as completion rules driven by the growing ABox and the static TBox, where each application of the expansion rule "completes" the ABox a step further and produces more complete ABoxes. An alternative notation from [HS07] was also commonly adopted in the literature.

$\sqcap$-rule	**if** $\mathcal{A}$ contains $(C_1 \sqcap C_2)(x)$, but it does not contain both $C_1(x)$ and $C_2(x)$. **then** $\mathcal{A}' = \mathcal{A} \cup \{C_1(x), C_2(x)\}$
$\sqcup$-rule	**if** $\mathcal{A}$ contains $(C_1 \sqcup C_2)(x)$, but neither $C_1(x)$ nor $C_2(x)$. **then** $\mathcal{A}' = \mathcal{A} \cup \{C_1(x)\}$, $\mathcal{A}'' = \mathcal{A} \cup \{ C_2(x)\}$
$\exists$-rule	**if** $\mathcal{A}$ contains $(\exists R.C)(x)$, but there is no individual name z such that $C(z)$ and $R(x, z)$ are in $\mathcal{A}$. **then** $\mathcal{A}' = \mathcal{A} \cup \{ C(y), R(x, y)\}$ where y is an individual name not occurring in $\mathcal{A}$.
$\forall$-rule	**if** $\mathcal{A}$ contains $(\forall R.C)(x)$ and $R(x, y)$, but it does not contain $C(y)$. **then** $\mathcal{A}' = \mathcal{A} \cup \{C(y)\}$.

Figure 2.2: Expansion Rules for $\mathcal{ALC}$

Algorithm 1 above will guarantee termination because each expansion rule can only be applied to the same fact once. As shown in the expansion rules, the precondition of each expansion rule checks the presence of certain ABox or TBox sentences. The TBox is static and no additional axioms are added in the course of the algorithm, so it will not cause any problem to termination. On the other hand, the ABox is dynamic and new assertions are being added to it each time an expansion rule is triggered. It means that the algorithm will not terminate if there are new and different assertions being added at each expansion. In the case of an unfoldable TBox this situation is not possible because the size of each assertion is fixed and expansions will only cause new assertions of the same size or less to be created. However, this is not the case for terminologies with cyclic definitions or General Inclusion Axioms (GCIs) where special treatment is necessary to

ensure termination.

Blocking

Blocking ensures termination of the tableau algorithm. In the absence of cyclic definitions, termination is guaranteed by the expansion rules. Each of the expansion rules is essentially applicable to the same set of DL sentences only once, the rule will then trigger the expansion of the ABox which makes this set of DL sentences no longer applicable to the rule. However, this is not the case in the presence of cyclic definitions where expansion rules can be applied indefinitely because the $\exists$-rule could generate an infinite number of individuals. Blocking checks for certain conditions in the expanding ABox and causes certain rules (in particular, the $\exists$-rule) to become inapplicable at some stage during the expansion. Blocking should happen at the right point during the expansion to ensure completeness of the algorithm.

Different blocking conditions are needed for description logics with different expressivity. Subset blocking is appropriate for the description logic $\mathcal{ALC}$, equivalence blocking for description logics with inverse roles (e.g. $\mathcal{ALCI}$), and pairwise blocking for description logics with role hierarchy (e.g. $\mathcal{SHIQ}$ and $\mathcal{SHOIN}$). These blocking conditions are outlined below. Note that the notation $\mathcal{L}(x) = \{C \mid C(x) \in \mathcal{A}\}$ and $\mathcal{L}(x, y) = \{R \mid R(x, y) \in \mathcal{A}\}$

Definition 2.8 (Subset Blocking)
An individual y is blocked by x if and only if $\{C \mid C(y) \in \mathcal{A}\} \subseteq \{C' \mid C'(x) \in \mathcal{A}\}$ and $y > x$, where $y > x$ means that y was introduced later than x.

Definition 2.9 (Equivalence Blocking)
An individual y is blocked by x if and only if $\{C \mid C(y) \in \mathcal{A}\} \equiv \{C' \mid C'(x) \in \mathcal{A}\}$ and $y > x$, where $y > x$ means that y was introduced later than x.

Definition 2.10 (Pair-wise Blocking [HST99])

A node is blocked if and only if it is directly or indirectly blocked[3]*. A node x is directly*

[3]A node is indirectly blocked if at least one of its ancestors is directly blocked.

blocked if and only if none of its ancestors are blocked, and it has ancestors x', y and y' such that:

- *x is a successor of x' and y is a successor of y' and*
- *$\mathcal{L}(x) = \mathcal{L}(y)$ and $\mathcal{L}(x') = \mathcal{L}(y')$ and*
- *$\mathcal{L}(\langle x', x\rangle) = \mathcal{L}(\langle y', y\rangle)$.*

2.3 Other Description Logics

The family of description logics contains many other sublanguages. We introduce some of these sublanguages in this section.

2.3.1 Description Logic $\mathcal{ALCN}$

The description logic $\mathcal{ALCN}$ is a slight extension to $\mathcal{ALC}$ with Qualified Cardinality Restrictions. This extension has added two important concept constructs of the form: $(\leq nR.C)$ and $(\geq nR.C)$. The additional constructs constrain on the number of values of a given type. For example, $(\geq 3hasChild.Male)(mary) \in \mathcal{A}$ means that each interpretation satisfying this concept assertion will have an individual *mary* who has three children and they all belong to the class *Male*. Note that, these individuals can be either named (i.e., individuals that appear in the ABox) or unnamed (i.e., individuals that do not appear in the original ABox). More formally, an interpretation $\mathcal{I}$ satisfies the following: $(\leq nR.C)^{\mathcal{I}} = \{x \in \Delta^{\mathcal{I}} | \#R^{\mathcal{I}}(x, C) \leq n\}$ and $(\geq nR.C)^{\mathcal{I}} = \{x \in \Delta^{\mathcal{I}} | \#R^{\mathcal{I}}(x, C) \geq n\}$. We show in Fig. 2.3 the additional rules that are needed to form the tableau algorithm for the description logic $\mathcal{ALCN}$.

2.3.2 Description Logic $\mathcal{SHOIQ}$

Another important class of description logics is known as $\mathcal{SHOIQ}$ which is an extension of the classical $\mathcal{ALC}$ with existential and universal restrictions, at-least and at-most number restrictions, role hierarchy and inverse role constructors. The semantics of $\mathcal{SHOIQ}$

$\geq$-rule	**if** $\mathcal{A}$ contains $(\geq nR)(x)$, and there are no individual names $z_1, \ldots, z_n$ such that $R(x, z_i)$ $(1 \leq i \leq n)$ and $z_i \neq z_j$ $(1 \leq i < j \leq n)$ are contained in $\mathcal{A}$. **then** $\mathcal{A}' = \mathcal{A} \cup \{ R(x, y_i) \mid 1 \leq i \leq n\} \cup \{y_i \neq y_j \mid (1 \leq i < j \leq n)\}$, where $y_1, \ldots, y_2$ are distinct individual names not occurring in $\mathcal{A}$.
$\leq$-rule	**if** $\mathcal{A}$ contains distinct individual names $y_1, \ldots, y_{n+1}$ such that $(\leq nR)(x)$ and $R(x, y_1), \ldots, R(x, y_{n+1})$ are in $\mathcal{A}$, and $y_i \neq y_j$ is not in $\mathcal{A}$ for some $i \neq j$. **then** For each pair y_i, y_j such that $i > j$ and $y_i \neq y_j$ is not in $\mathcal{A}$, the ABox $\mathcal{A}_{i,j} = [y_i/y_j]\mathcal{A}$ is obtained from $\mathcal{A}$ by replacing each occurrence of y_i by y_j.

Figure 2.3: Tableau algorithm additional rules for $\mathcal{ALCN}$

is defined the way as $\mathcal{ALC}$. Formally, an interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, .^{\mathcal{I}})$ is a tuple, where $\Delta^{\mathcal{I}}$ contains the set of all elements in the domain of interest and $.^{\mathcal{I}}$ is a function that maps every concept to a set of individuals in $\Delta^{\mathcal{I}}$ and every role name to a set of pairs (of individuals) in $\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$. In addition, an interpretation $\mathcal{I}$ satisfies the following conditions:

$$
\begin{aligned}
(C \sqcap D)^{\mathcal{I}} &= C^{\mathcal{I}} \cap D^{\mathcal{I}}, \\
(C \sqcup D)^{\mathcal{I}} &= C^{\mathcal{I}} \cup D^{\mathcal{I}}, \\
\neg C^{\mathcal{I}} &= \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}, \\
\#o^{\mathcal{I}} &= 1 \text{ for all } o \in N_I, \\
(\exists R.C)^{\mathcal{I}} &= \{x \in \Delta^{\mathcal{I}} | R^{\mathcal{I}}(x, C) \neq \emptyset\}, \\
(\forall R.C)^{\mathcal{I}} &= \{x \in \Delta^{\mathcal{I}} | R^{\mathcal{I}}(x, \neg C) = \emptyset\}, \\
(\leq nR.C)^{\mathcal{I}} &= \{x \in \Delta^{\mathcal{I}} | \#R^{\mathcal{I}}(x, C) \leq n\}, \\
(\geq nR.C)^{\mathcal{I}} &= \{x \in \Delta^{\mathcal{I}} | \#R^{\mathcal{I}}(x, C) \geq n\},
\end{aligned}
$$

Moreover, $\mathcal{SHOIQ}$ supports transitive and inverse roles. The semantics of roles are described in [HS07]. In addition to subsumption and equivalence axioms, $\mathcal{SHOIQ}$ also

support role hierarchy on role names. That is, it allows role axioms of the form $R \sqsubseteq S$. An interpretation of this axiom satisfies $R^{\mathcal{I}} \subseteq S^{\mathcal{I}}$.

2.3.3 Description Logic $\mathcal{SROIQ}$

The description logic $\mathcal{SROIQ}$ was proposed in [HKS06] as a combination of $\mathcal{SHOIN}$ and $\mathcal{RIQ}$. It was designed to make some of the most expressive yet decidable fragments of description logics to be more useful in practise. This has led to the development of the OWL 1.1.

In particular, $\mathcal{SROIQ}$ introduces an additional component of a DL knowledge base known as the Role Box (or RBox for short). Consequently, a DL knowledge base in $\mathcal{SROIQ}$ is defined as $K = (\mathcal{T}, \mathcal{A}, \mathcal{R})$, where $\mathcal{T}$ and $\mathcal{A}$ are the TBox and ABox respectively, and the additional $\mathcal{R}$ is the RBox where all role axioms and assertions now reside. That is, the TBox no longer stores any role axioms. The description logic $\mathcal{SROIQ}$ extends $\mathcal{SHOIQ}$ [HS07] with the following features:

1. Disjoint roles
2. Reflexive and irreflexive roles
3. Negated role assertions
4. Complex role inclusion axioms
5. Universal role U
6. Local reflexivity of roles

The set $Diag^{\mathcal{I}}$ is defined as $\{\langle x, x\rangle \mid x \in \Delta^{\mathcal{I}}\}$. An interpretation $\mathcal{I}$ of $\mathcal{SROIQ}$ is an interpretation of $\mathcal{SHOIQ}$ with the following extensions:

$$
\begin{aligned}
(C \sqcap D)^{\mathcal{I}} &= C^{\mathcal{I}} \cap D^{\mathcal{I}}, \\
(C \sqcup D)^{\mathcal{I}} &= C^{\mathcal{I}} \cup D^{\mathcal{I}}, \\
\neg C^{\mathcal{I}} &= \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}, \\
\#o^{\mathcal{I}} &= 1 \text{ for all } o \in N_I,
\end{aligned}
$$

$$\mathcal{I} \models \neg R(x,y) \quad \textit{if} \quad \langle x,y\rangle \notin \mathcal{R}^{\mathcal{I}} \tag{2.10}$$

$$\mathcal{I} \models \mathbf{Sym}(R) \quad \textit{if} \quad \langle x,y\rangle \in R^{\mathcal{I}} \ \textit{implies} \ \langle y,x\rangle \in R^{\mathcal{I}} \tag{2.11}$$

$$\mathcal{I} \models \mathbf{Tra}(R) \quad \textit{if} \quad \langle x,y\rangle \in R^{\mathcal{I}} \ \textit{and} \ \langle y,z\rangle \in R^{\mathcal{I}} \ \textit{imply} \ \langle x,z\rangle \in R^{\mathcal{I}} \tag{2.12}$$

$$\mathcal{I} \models \mathbf{Ref}(R) \quad \textit{if} \quad Diag^{\mathcal{I}} \subseteq R^{\mathcal{I}} \tag{2.13}$$

$$\mathcal{I} \models \mathbf{Irr}(R) \quad \textit{if} \quad R^{\mathcal{I}} \cap Diag^{\mathcal{I}} = \emptyset \tag{2.14}$$

$$\mathcal{I} \models \mathbf{Dis}(R,S) \quad \textit{if} \quad R^{\mathcal{I}} \cap S^{\mathcal{I}} = \emptyset \tag{2.15}$$

The semantics of disjoint roles is captured by Equation 2.15, of reflexive and irreflexive roles by Equation 2.13 and 2.14 respectively and negated role assertion by Equation 2.10. The universal role $U^{\mathcal{I}}$ is simply defined as $\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$ and local reflexivity of a role R is defined as $(\exists R.Self)^{\mathcal{I}} = \{x \mid \langle x,x\rangle \in R^{\mathcal{I}}\}$. Note that $Self$ is a special concept name.

Definition 2.11 (Role Box [HKS06])
A $\mathcal{SROIQ}$-role box is a set $\mathcal{R} = \mathcal{R}_h \cup \mathcal{R}_a$, where $\mathcal{R}_h$ is a regular role hierarchy and $\mathcal{R}_a$ is a finite, simple set of role assertions. An interpretation satisfies a role box $\mathcal{R}$ (written $\mathcal{I} \models \mathcal{R}$) if $\mathcal{T} \models \mathcal{R}_h$ and $\mathcal{I} \models \phi$ for all role assertions $\phi \in R_a$. Such an interpretation is called a model of $\mathcal{R}$.

2.3.4 The DL-Lite Family [GLRV04]

The DL-Lite family of description logics is fragment of DLs that are designed to perform tractable reasoning. The DL-Lite family corresponds to the OWL 2 QL sublanguage of the OWL 2.0 standard. The syntax of DL-Lite core is defined below:

$$B ::= A \mid \exists R \mid \exists R^{-}$$

$$C ::= B \mid \neg B \mid C_1 \sqcap C_2$$

where A denotes an atomic concept, C denotes a complicated concept, R denotes a role name, R^{-} denotes the inverse of the role R and $\exists R$ denotes unqualified existential restriction on atomic role R.

2.3.5 The $\mathcal{EL}$ Family

The $\mathcal{EL}$ family represents a unique stream of research in description logics that focuses towards computational performance of reasoning [BLS06] rather than expressive power. Specifically, $\mathcal{EL}$ logics are tractable fragments of description logics. The class of $\mathcal{EL}$ logics consists of three variants: $\mathcal{EL}$, $\mathcal{EL}^+$ and $\mathcal{EL}^{++}$. In terms of reasoning these logics employ a technique known as structural subsumption, instead of traditional DL reasoning algorithms such as the tableau algorithm. The syntax of $\mathcal{EL}$ is presented below:

$$
\begin{aligned}
\langle concept \rangle ::= & \text{“(”} \langle concept \rangle \text{“} \sqcap \text{”} \langle concept \rangle \text{“)”} \quad | \\
& \text{“}\exists\text{”} \langle role\_name \rangle \text{“.”} \langle concept \rangle \\
& \langle concept\_name \rangle \\
\langle role \rangle ::= & \text{“(”} \langle concept \rangle \text{“} \circ \text{”} \langle concept \rangle \text{“)”} \quad | \\
& \text{“}\exists\text{”} \langle role\_name \rangle \text{“.”} \langle concept \rangle \\
& \langle concept\_name \rangle
\end{aligned}
$$

Since we do not make explicit use of $\mathcal{EL}^+$ or $\mathcal{EL}^{++}$ in this dissertation, we will omit the semantic descriptions of $\mathcal{EL}^+$ or $\mathcal{EL}^{++}$. Instead, we point the readers to the literature [BLS06, BBL05].

2.3.6 Reasoning with Rules

One of the limitations of the original version of OWL was that it had not been designed to express rules. This has led to the proposal of SWRL as an extension to OWL and later to the development of OWL 2 RL specifically for efficient handling of rules by giving up some expressive power of OWL.

Below is an example of a horn-like rule that cannot be expressed in OWL. The rule is composed of three binary predicates `has_uncle`, `has_father` and `has_brother`; and three variables `x`,`y` and `z`. It reads ‘if x has a father y and y has a brother z, then x has

an uncle z'.

$$has\_uncle(x, z) \leftarrow has\_father(x, y) \wedge has\_brother(y, z)$$

A typical SWRL rule consists of an antecedent and a consequent. The antecedent is also known as the body of the rule and the consequent as the head. Both the antecedent and the consequent are composed of predicates with arity r where $r > 0$. It is generally required that predicate and individual names appearing in rules must also appear in the original ontology. This excludes any individual that is generated by the inference engine of the OWL ontology. In other words, it only takes named individuals into account.

There are at least two known interpretations for the above rule in the community. One has been adopted in SWRL and another by the so-called SWRL-like rules. The difference lies in the way the arrow ($\leftarrow$) is interpreted. For SWRL-like rules the arrow is interpreted as in production systems. That is, if the antecedent of the rule is satisfied then the rule is fired and the consequent of the rule is triggered.

For SWRL rules, however, the arrow is interpreted as a logical implication where the contrapositive holds. For example, if we have a rule $penguin(x) \leftarrow bird(x) \wedge fly(x)$, then its contrapositive $\neg penguin \rightarrow \neg bird(x) \vee \neg fly(x)$ also holds.

The difference between the interpretations of the two types of rules is similar to the difference between logical implication and causality. This topic has been extensively studied in the context of Reasoning about Action.

This difference lead to the development of ontology reasoners with different treatments for rules.

Currently, both RacerPro [HMW03] and KAON2 [Mot08] support SWRL rules.

2.3.7 Summary and Discussion

In this chapter, we introduced description logics as a family of knowledge representation languages that are widely used to represent ontologies. We described the syntax and semantics of the classical description logic $\mathcal{ALC}$, as to lesser extent of some other description logics such as $\mathcal{SROIQ}$ and $\mathcal{EL}$. We presented the tableau-based reasoning

procedure that is used by most of the state-of-the-art ontology reasoners to perform various reasoning services supported by a description logic knowledge base. We also described some of the underlying assumptions of description logics (e.g. the open-world assumption) and how they differ from assumptions in relational databases. We showed that there are limitations to what can be expressed in DLs.

Chapter 3

Ontology Contraction

3.1 Chapter Introduction

In this chapter, we study the topic of ontology contraction as a means to retract knowledge from an ontology. The framework we adopt is based on the classical belief contraction literature and we focus on reformulating those techniques into the context of description logics. In particular, we explore the classical remainder set operator for propositional logic and show that such operator is not desirable for the description logic setting. Moreover, we introduce a remainder set operator for description logics and argue that such operator yields finer-grained results than classical remainder set. We establish a representation theorem for the remainder set operator for DL by showing correspondence between the operator and a set of revised postulates. Furthermore, we extend our results to the multiple contraction setting and present both the notions of package contraction and choice contraction for description logics.

3.2 Motivation

The goal of this chapter is to look at the problem of ontology contraction from a theoretical standpoint. To understand why this is important, consider a knowledge engineer who is reusing an ontology available in the public domain (e.g., SNOMED). In most real-world scenarios, it is unlikely that the existing ontology captures exactly the domain that

the knowledge engineer has in mind. Therefore, it is necessary for the knowledge engineer to modify the ontology in order to properly model the intended domain of interest. Typically, this process involves removing unwanted inferred knowledge from the existing ontology. To facilitate this process, the knowledge engineer may use a standard ontology editor (such as Protege) to make changes to the ontology and manually remove those sentences (axioms or assertions) that lead to the unwanted inferred knowledge. However, this poses a number of challenges. Firstly, an ontology can be huge and may contain millions of axioms so it may not be practical to perform the task manually. Secondly, it is not always easy to identify those sentences that lead to the unwanted inferred knowledge and at the same time ensure minimal information loss. It is therefore our goal to delve into the ontology contraction operation from the theoretical viewpoint. This work will have impact in understanding a number of operations that are related to contraction, including ontology merging, revision, alignment and mapping.

3.3 Related Work

There has been a large body of work in the area of Belief Change where the process of an agent's epistemic state undergoing changes is being modelled.

The most foundational piece of work in this area is the AGM framework of belief change [Gar88, Pep07], where the agent's epistemic state is represented as a set of formal sentences. While the framework is not tied to a specific formal logic, it is typically assumed to be in a propositional setting, and is subject to meeting certain properties and assumptions (satisfied by propositional logic). As we shall see in later sections, this poses many problems when one tries to cast the techniques in the AGM framework directly to description logics.

There have been many attempts in resolving inconsistencies with ontologies [QP07]. In particular, Flouris et al. [FHP$^+$06] pointed out the limitations of certain description logics in meeting some of the requirements of the AGM framework. These description logics are considered as non AGM-compliant, as termed by Flouris et al.

One line of active research [LLMW06, FPA05a, HWK06, FPA04, FPA05b] in the area

is to identify (or design new) description logic constructs that will make certain class of description logics AGM-compliant. In particular, the work by Flouris et al. [FHP+06] studied negation of TBox sentences, which is generally *not* supported in classical description logics. However, negations are fundamental in propositional logic, which allows negation of any arbitrary formula including double negations.

In relation to the AGM framework, the Levi identity [Lev77, Han99] and Harper identity [Har75, Han99] established connections between contraction and revision. The Levi identity allows revision to be formulated in terms of contraction and expansion. It is formally defined as follows: $K * \alpha = (K \div \neg\alpha) + \alpha$. Loosely speaking, the Levi identity states that revising a knowledge base K by a sentence α is the same as first contracting K by the negation of α (i.e., $\neg\alpha$) then expanding the result with α. Similarly, the Harper identity formulates contraction in terms of revision. Formally, it is defined as: $K \div \alpha = K \cap K * \neg\alpha$. The Harper identity states that contracting a knowledge base K by a sentence α is the same as first revising it by the negation of α then taking the conjunction of the result with K. Both notions, however, requires the use of negation in the target logic. Flouris et al. identified a number of conditions that negations of axioms must satisfy in order for the Levi identity [Lev77, Han99] and Harper identity [Har75, Han99] to be applicable. They have also identified a way to negate axioms that is applicable to most popular description logics.

Another line of research by Ribeiro et al. [RW06] looks at the the applicability of AGM in the absence of negations. The essence of the research was the investigation of the external revision operator that leads to so-called semi-revision. Semi-revision can be thought of as a weaker form of revision where certain postulated conditions in classical revision (or internal revision) are not fully met. For example, the success postulate is not guaranteed in the case of semi-revision. However, the external revision operator has the advantage that inconsistencies are being removed after new information has been incorporated. This allows to identify sets of sentences that lead to inconsistencies and avoids the explicit use of negations. In addition, they have presented two version of base revision operators based on kernel contraction that attempts to augment semi-revision with some versions of success. One satisfies their defined notion of weak-success and

another satisfies (full) success.

3.4 Belief Change

The problem of belief change has been considered in both the belief set and belief base contexts [AGM85, Han99]. The former considers sets of sentences closed under logical consequences, meaning that if a belief set contains a certain sentence, then it also contains all sentences implied by it. The latter considers sets of sentences *not* necessarily closed under logical consequence. It means that a belief base is simply a set of sentences. However, this does not mean that an agent that acquires a belief base model does not believe in the consequences of its belief base. From the computational point of view, the belief base model is considered to be a more promising approach compared to the belief set model. The belief set and the belief base model are sometimes connected to the so-called coherentist and fundamentalist viewpoints.

Analogously, we discuss both the belief set and belief base models in the ontology context. For convenience, we shall refer to these as the ontology set and the ontology base models respectively.

Although both of these models have been studied in the literature, we focus primarily on the ontology base model for computational reasons. In particular, since we are expressing ontologies in DL and most DLs are sufficiently expressive (compared to propositional logic at least) and allow for domains that are infinite, it is expected that ontology sets will be infinite. For example, an ontology set that contains $C(s)$ will necessarily contain all its consequences, including $(C \sqcup \forall R.D)(s)$.

An important problem that is to be highlighted here is that description logics do not natively support negation of sentences in general. In particular, we cannot express the negation of an TBox sentence in classical description logics, such as $\mathcal{ALC}$, $\mathcal{SHOIN}$, $\mathcal{SROIQ}$ and $\mathcal{EL}$. While one could turn to the literature [BBH96, BKW03] that provides other non-standard constructs, using these constructs is not always the best option because many of the existing reasoners do not yet provide support for them. Moreover, negation of ABox sentences is also not fully supported, although the description logic

$\mathcal{SROIQ}$ and $\mathcal{EL}^{++}$ allow for negation of role assertion. This is also made known in OWL 2.0. Many of the classical description logics, such as $\mathcal{SHIQ}$, $\mathcal{SHOIN}$ and $\mathcal{EL}$, do not support negation of role assertion. Nonetheless, many of the AGM constructions are formalised in terms of negation of sentences. This makes it prohibitive to directly apply AGM techniques to description logics. In order to solve this problem, one must either provide sufficient expressive power to support negation or to explore other means that do not use negations. We will explore some of these options in this chapter.

3.5 Ontology Contraction

In belief change, the contraction operation deals with the consistent removal of certain information from the agent's beliefs. The problem of ontology contraction can be considered in a similar fashion, in the sense that we could explore means to consistently remove information from an ontology. Instead of beliefs which is typically represented as a set of propositional formulas (closed or not closed under logical consequences), we represent an ontology as a set of DL sentences and consider the scenario where a single DL sentence is removed from the ontology. We proceed by introducing the notion of a remainder set which is a basic tool to build a partial-meet contraction operator. We then motivate our research by demonstrating how a remainder set can be used for DLs and describe some of its limitations. We then present a new version of remainder set for DL based on the notion of *exceptions*.

3.5.1 Remainder Set

We present below the classical notion of remainder set and an example to demonstrate how the remainder set operation works.

Definition 3.1 ([AM81])

Let A be a set of sentences and α a sentence. The set $A \perp \alpha$ is the set such that $B \in A \perp \alpha$ if and only if:
1. $B \subseteq A$

2. $\alpha \notin Cn(B)$

3. There is no set B' *such that* $B \subset B' \subseteq A$ *and* $\alpha \notin Cn(B')$

Example 3.2

Let $K = (T, A)$ where $\mathcal{T} = \{Bird \sqsubseteq Penguin\}$ and $\mathcal{A} = \{Bird(tweety), Bird(chirpy)\}$, and let α be $Penguin(tweety)$. Note that we consider K as a single set that is the union of $\mathcal{T}$ and $\mathcal{A}$.

$$K \perp \alpha = \{\{Bird \sqsubseteq Penguin, Bird(chirpy)\},$$
$$\{Bird(tweety), Bird(chirpy)\}\}$$

The remainder set is a commonly used construction for belief contraction, such construction is intuitive and has been shown to satisfy the classical belief contraction postulates. However, there are limitations to the remainder set which makes it inadequate to capture the structure of more expressive languages. In particular, remainder set has the property that it limits its solutions to only subsets of the original set of sentences. This is sufficient for propositional logic because the structure of propositional languages are (relatively) inexpressive. In the context of DL, however, it is possible to explore the structure of DL sentences further to produce finer-grained solutions. We develop a version of the remainder set construction based on the notion of exceptions.

The construction relies on the use of nominals a standard construct in almost all popular expressive description logics, including $\mathcal{ALCO}$, $\mathcal{SHOIQ}$ and $\mathcal{SRIOQ}$. It is important to note that nominals are not the only means to construct a remainder set. It is possible to replace nominals with some other cardinality based constructs that allow exceptions to be expressed adequately, such as the use of cardinality restrictions on concepts [BBH96] or existence axiom [HPS03].

Classical remainder set is constructed by computing maximal subsets that do *not* imply the sentence to be removed. This is natural for propositional logic because all sentences in the original set of sentences are, in a sense, treated as having equal value.

Unless one imposes an ordering on the set of sentences, there is no reason why a particular sentence should be seen more valuable than another one, especially if the two sentences have no logical relations with each other (i.e. they do not share any models). This is different from description logic knowledge bases where sentences are divided into TBox or ABox sentences. TBox sentences can be seen as intensional knowledge and they impose constraints onto sets of individuals in the domain. This makes them, in a way, more valuable than ABox sentences (extensional knowledge) because each ABox sentence refer only to certain individuals, thus affecting only a small portion of the domain. In constructing a remainder set for DL, we could give preference to TBox sentences by making use of the notion of exceptions.

Instead of removing a TBox sentence completely as we do in the classical remainder set construction, one could weaken TBox (and ABox) sentences in the original set of DL sentences until it no longer implies the sentence that we wish to retract. Note that weakening an ABox sentence is the same as removing it, in this sense the way we handle ABox sentences is no different from the classical notion of remainder set. As suggested by Qi et al. [QLB06], one could also weaken an ABox sentence $C(s)$ to $\top(s)$. This has the advantage of being able to retain the presence of the individual name s in the case where $C(s)$ is the last sentence that s is in presence[1]. The description logic $\mathcal{SROIQ}$ allows one to express universe for roles, therefore one could similarly express the weakening of ABox sentence of the form $R(s,t)$ to $U(s,t)$ where $U^{\mathcal{I}} = \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$.

We now introduce some definitions (Definition 3.3-Definition 3.5) to describe the notion of a weakened DL knowledge base. These definitions will be used to define our refined version of remainder set for DL (Definition 3.6).

Definition 3.3 (Weakened TBox adapted from [QLB06])

Let i be an integer and $\pi_i(S) = \{S' \subseteq S \,|\, |S'| = i\}$. Also, let $\mathcal{N}_s^{\mathcal{K}}$ be the set all individual names appearing in the ontology $\mathcal{K}$. Suppose ϕ is a TBox sentence of the form

[1]It is good to weaken an ABox to Top in cases where retaining the information about the individual names is important, but this is not always needed.

$C \sqsubseteq D$, the i-weakening of ϕ, written $d_i(\phi)$, is defined as follows:

$$d_i(\phi) = \begin{cases} \Big[(C \sqcap \neg\{a_1\} \sqcap \ldots \sqcap \neg\{a_i\}) \sqsubseteq D \,|\, \{a_1, \ldots, a_i\} \in \pi_i(\mathcal{N}_s^{\mathcal{K}})\Big] & \textit{for} \quad i \leq |\mathcal{N}_s^{\mathcal{K}}| \\ \Big[\bot \sqsubseteq \top\Big] & \textit{for} \quad i > |\mathcal{N}_s^{\mathcal{K}}| \end{cases}$$

Informally, i is the number of exceptions made to ϕ. Notice that, the $(|\mathcal{N}_s^{\mathcal{K}}| + 1)$-weakening of a TBox sentence is equivalent to $\bot \sqsubseteq \top$. This is the same as dropping the sentence. The m-weakening of a TBox $\mathcal{T}$ is defined as:

$$d_m(\mathcal{T}) = \Big[\{\phi'_1, \ldots, \phi'_n\} \mid \phi'_j \in d_{i_j}(\phi_j) \textit{ for all } 1 \leq j \leq n; \sum_{k=1}^{n} i_k = m\Big]$$

where $\phi_1, \ldots, \phi_n$ are elements (axioms) of $\mathcal{T}$.

Definition 3.4 (Weakened ABox)
The m-weakening of an ABox $\mathcal{A}$ is defined as:

$$d_m(\mathcal{A}) = \Big[A' \subseteq A \mid |A| - |A'| = m\Big]$$

Definition 3.5 (Weakened DL Knowledge Base)

Let $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ be a DL knowledge base, where $\mathcal{T}$ is the TBox and $\mathcal{A}$ is the ABox. The m-weakening of K is defined as:

$$d_m(\mathcal{K}) = \Big[\mathcal{T}' \cup \mathcal{A}' \mid T' \in d_j(\mathcal{T}); A' \in d_k(\mathcal{A}); \\ m = j + k; j \leq |\mathcal{T}| \times (|\mathcal{N}_s^{\mathcal{K}}| + 1); k \leq |\mathcal{A}|\Big]$$

where $|\mathcal{T}|$ and $|\mathcal{A}|$ are the sizes of the TBox and ABox respectively, and $|\mathcal{N}_s^{\mathcal{K}}|$ is the number of individual names in $\mathcal{K}$.

We define below the notion of remainder set for DL, which is a refinement of the classical notion of remainder set. The idea is that, we want to treat TBox sentences as

being more important than ABox sentences.

The three conditions we specify below (Definition 3.6) are analogous to that defined for the classical remainder set. The first condition requires that B (element of the remainder set for DL) be an element of the i-weakening of A, which consists of both weakened TBox and ABox sentences. The second condition is the same as that defined in the classical remainder set it specifies that B does not imply α. The third condition requires that B be a weakened knowledge base of A and that there is no weakened knowledge base B', such that B' is also a weakened knowledge base of A, and B is a weakened knowledge base of B'. In other words, this condition ensures that B is "maximal" in the sense that it contains sentences that are least weakened with respect to A.

Definition 3.6 (Remainder Set for DL)

Let A be a set of DL sentences and α be a DL sentence. $A\perp_{dl}\alpha$ is the set such that $B \in A\perp_{dl}\alpha$ if and only if:
1. $B \in d_i(A)$ for some i.
2. $\alpha \notin Cn(B)$.
3. There is no B' such that $B' \in d_i(A)$ and $B \in d_j(B')$ for some $0 \leq i$ and $0 < j$, and $\alpha \notin Cn(B')$.

We demonstrate below the notion of remainder set for DL with a simple example.

Example 3.7

Consider Ex. 3.2 again but this time we compute the remainder set for DL instead. Let $K = (T, A)$ where $\mathcal{T} = \{Bird \sqsubseteq Penguin\}$ and $\mathcal{A} = \{Bird(tweety), Bird(chirpy)\}$, and let α be $Penguin(tweety)$.

$$K\perp_{dl}\alpha = \{\{Bird \sqsubseteq Penguin, Bird(chirpy)\},$$
$$\{(Bird \sqcap \neg\{tweety\}) \sqsubseteq Penguin, Bird(tweety), Bird(chirpy)\}\}$$

In Ex. 3.2 and Ex. 3.7, we see that both examples are able to conclude that removing the statement "Tweety is a bird" is a possible solution. But for the other solution, classical remainder set requires dropping the statement that birds are penguins, while our notion of remainder set for DL retains this statement, but makes an exception to specify that *all* individuals except Tweety satisfy this statement. It is clear from this example (Ex. 3.7) that our notion of remainder set for DL produces more intuitive results.

Proposition 3.8

Let A be a set of DL sentences and α a DL sentence, then $B' \in A\perp\alpha$ if and only if there is some set $B \in A\perp_{dl}\alpha$ such that $B' \subseteq B$.

Proof. ($\Longrightarrow$)
Suppose $B' \in A\perp\alpha$ and there is no $B \in \mathcal{A}\perp_{dl}\alpha$ such that $B' \subseteq B$. By Definition 3.1, $\alpha \notin Cn(B')$ but we also know that $\alpha \in B$ for all B such that $B' \subset B \subseteq A$. Consider two cases (1) B' is A, and (2) $B' \subset A$. If B' is A, then by Definition 3.1 and Definition 3.6, we have $A\perp\alpha = A\perp_{dl}\alpha = A$. If $B' \subset A$, then by Definition 3.1 we know that there exists some $\phi \in A$ such that $\alpha \in Cn(B' \cup \{\phi\})$ (Note that, there does not exist $B'' \in d_i(A)$ such that $B' \cup \{\phi\} \in d_j(B'')$ for $0 \leq i$ and $0 < j$, and $\alpha \notin Cn(B'')$, because $\alpha \in Cn(B' \cup \{\phi\})$). If ϕ is an ABox sentence, then by Definition 3.4 weakening it will have it removed so we have B' and we know that $\alpha \notin Cn(B')$. We have now two cases to consider: (1) B' is minimal, which means $B' \in A\perp_{dl}\alpha$, or (2) B' is not minimal in which case there exists another set B'' such that $B'' \in d_i(A)$ and $B' \in d_j(B'')$ for some $0 \leq i$ and $0 < j$ and $\alpha \notin Cn(B'')$, and B'' is minimal. That means, $B'' \in A\perp_{dl}\alpha$. If ϕ is a TBox sentence, then by Definition 3.4 it will be weakened to $d_1(B' \cup \{\phi\})$ where $B' \cup \{d_1(\phi)\} \in d_1(B' \cup \{\phi\})$. If $\alpha \notin Cn(B' \cup \{d_1(\phi)\}$ then $B' \in A\perp_{dl}\alpha$. Otherwise, $B' \cup \{d_1(\phi)\}$ is weakened to $d_1(B' \cup \{d_1(\phi)\})$ where $B' \cup \{d_2(\phi)\} \in d_1(B' \cup \{d_1(\phi)\})$. And so on, until either $\alpha \notin Cn(B' \cup \{d_n(\phi)\}$ for some $n < |s|$, or we have $B' \cup \{d_n(\phi)\} = B'$ for $n = |s|$. Hence, either $B' \in A\perp_{dl}\alpha$ or there exists some set B'' such that $B'' \in A\perp_{dl}\alpha$.
($\Longleftarrow$)
Suppose there is a set $B \in A\perp_{dl}\alpha$, then by Definition 3.6, we have $\alpha \notin Cn(B)$. So there exists a subset B' of B, such that $\alpha \notin Cn(B')$. If $B' \notin A\perp\alpha$ and $\alpha \notin Cn(B')$, then by

Definition 3.1, there exists a set B'' such that $B' \subset B''$, $B'' \in A \perp \alpha$ and $\alpha \notin Cn(B'')$. But by ($\Longrightarrow$), we know that $B'' \in A \perp \alpha$ means there exists some set $B''' \in A \perp_{dl} \alpha$ where $B'' \subseteq B'''$. □

3.5.2 Contraction Postulates

We present below the set of well-known partial meet contraction postulates for belief bases [Han99]. The set of postulates is intended to be a guide for a contraction operator with good behaviours. It does not specify a particular contraction operation. Instead, it is a legitimate set of rules that governs a class of contraction operators. Each of the postulates describes a property that a good contraction operator exhibits. For example, the success postulate requires that a contraction operator be always able to remove a sentence unless the sentence itself is a tautology (i.e. $\alpha \notin Cn(\emptyset)$).

We present below the set of four partial meet contraction postulates for belief bases proposed by Hansson et al. [Han99]. This is followed by a brief description of each postulate.

Definition 3.9 ([Han99])

The set of contraction postulates are listed below:

(Success) *If* $\alpha \notin Cn(\emptyset)$, *then* $\alpha \notin Cn(A \div \alpha)$

(Inclusion) $A \div \alpha \subseteq A$

(Relevance) *If* $\beta \in A$ *and* $\beta \notin A \div \alpha$, *then there is a set* A' *such that* $A \div \alpha \subseteq A' \subseteq A$ *and that* $\alpha \notin Cn(A')$ *but* $\alpha \in Cn(A' \cup \{\beta\})$

(Uniformity) *If it holds for all subsets* A' *of* A *that* $p \in Cn(A')$ *if and only if* $q \in Cn(A')$, *then* $A \div p = A \div q$.

The success postulate ensures that the sentence we would like to be removed is actually removed after contraction. The inclusion postulate says that the result of contraction must be a subset of the original set of sentences, which means one cannot introduce unnecessary sentences during contraction. The relevance postulate justifies those sentences that are in the original set of sentences (i.e., $\beta \in A$) but are removed after the contraction (i.e., $\beta \notin A \div \alpha$). It ensures that no sentences are removed for no reason. Uniformity imposes a view on looking at sentences from the perspective of the original set of sentences A. It requires that sentences, say α and β, that are implied by the same subset of sentences from A to yield identical outcomes when contracting A by these sentences. These four postulates together form the axiomatic characterisation of the partial meet contraction operator for belief bases. That is, not only that partial meet contraction operator satisfies these four postulates but an operator that satisfies these four postulates are necessarily a partial meet contraction operator.

Let γ be a selection function of $A \perp \alpha$, written $\gamma(A \perp \alpha)$, as defined in the classical partial meet contraction.

That is, $\gamma(A \perp \alpha) = A$ if $A \perp \alpha$ is an empty set, otherwise γ selects at least one element from $A \perp \alpha$.

Also, let $\div$ be an operator defined as:

$$A \div \alpha = \bigcap \gamma(A \perp_{dl} \alpha) \tag{3.1}$$

Our contraction operator does not satisfy all of the classical contraction postulates, specifically, it does not satisfy the inclusion and the relevance postulates. This is because our definition of remainder set is more refined than that of the classical remainder set. It is therefore necessary for us to also revise the postulates accordingly.

The inclusion postulate is revised so that we do not talk about subsets of A.

Instead when we look at the sentences in $A \div \alpha$, we would like to capture the idea that each of these sentences is somehow originated from A. We weakened the inclusion postulate to enforce that each sentence in $A \div \alpha$ be implied by some sentence in A.

It should be noted that the original version of the inclusion postulate is stronger than our version, though it is not appropriate for our construction. It means that every contraction operator satisfying the original inclusion postulate will also satisfy ours, but not the other way around.

Similarly, the relevance postulate is inappropriate in our case because the subset operator is not sufficient to capture the weakening approach we adopted for DL. To capture weakening properly, we incorporate a finer-grained subset operator in the definition of relevance to allow only those sentences that have a valid reason to be weakened. Intuitively, our definition required that sentences are weakened only if previous weakenings failed.

The new revised postulates are listed below. Note that we have introduced a notation $\hat{\sqsubseteq}$, where $A' \hat{\sqsubseteq} A$ if and only if $A' \in d_i(A)$ for some i.

Definition 3.10 (Revised Contraction Postulates)
The revised postulates are listed below:

(Success) *If* $\alpha \notin Cn(\emptyset)$, *then* $\alpha \notin Cn(A \div \alpha)$

(Inclusion') *If* $\beta' \in A \div \alpha$ *then* $\beta' \in Cn(\{\beta\})$ *for some* $\beta \in A$

(Relevance') *If* $\beta \in A$ *and* $\beta \notin A \div \alpha$, *then there is a set* A' *such that* $A \div \alpha \hat{\sqsubseteq} A' \hat{\sqsubseteq} A$ *and that* $\alpha \notin Cn(A')$ *but* $\alpha \in Cn(A' \cup \{\beta\})$

(Uniformity) *If it holds for all subsets* A' *of* A *that* $\alpha \in Cn(A')$ *if and only if* $\beta \in Cn(A')$, *then* $A \div \alpha = A \div \beta$.

Lemma 3.11 (Representation Theorem for Partial Meet Contraction for DL)
The operator $\div$ *is an operator of partial meet contraction for DL for a set* A *if and only if it satisfies the postulates of success, inclusion', relevance' and uniformity.*

Proof. This proof is divided into two parts, we first show that our construction satisfies the four postulates above then we show that the four postulates yield our construction above. We start with the case of construction-to-postulates. Note that these proofs are adapted from those in [Han99].

(Success) Suppose $\alpha \notin Cn(\emptyset)$ and $\alpha \in Cn(A \div \alpha)$. By Eq. 3.1, it follows that $\alpha \in Cn(\bigcap \gamma(A\bot_{dl}\alpha))$. If $\bigcap \gamma(A\bot_{dl}\alpha)$ is non-empty, then there is some subset B' of B such that $\alpha \in Cn(B')$ and $B \in A\bot_{dl}\alpha$. Hence $\alpha \in Cn(B)$ but by Definition 3.6 $\alpha \notin Cn(B)$. Therefore, $\bigcap \gamma(A\bot_{dl}\alpha)$ must be an empty set. That is, $\alpha \in Cn(\emptyset)$. So, we have a contradiction.

(Inclusion') If $\beta' \in A \div \alpha$, then by Eq. 3.1, $\beta' \in \bigcap \gamma(A\bot_{dl}\alpha)$. It follows that there is a set B' where $\beta' \in \mathcal{B}'$ and there is a set B such that $B' \subseteq B$ and $B \in A\bot_{dl}\alpha$, hence $\beta' \in B$. By Definition 3.6, $B \in A\bot_{dl}\alpha$ means $\beta \in d_i(A)$ for some i. By Definition 3.5, we know that $B = d_j(\mathcal{T}) \cup d_k(\mathcal{A})$ where $i = j + k$ and $A = \mathcal{T} \cup \mathcal{A}$. If $\beta' \in B$ then it is either in $d_j(\mathcal{T})$ or $d_k(\mathcal{A})$. Consider $\beta' \in d_k(\mathcal{A})$, then by Definition 3.4, we have $\beta' \in \mathcal{A}$, thus $\beta' \in A$. Thus, $\beta' \in Cn(\beta')$. Consider $\beta' \in d_j(\mathcal{T})$, then by Definition 3.3, we have $\beta' \in d_n(\phi)$ where $n \leq j$ and $\beta \in \mathcal{T}$. β is of the form $C \sqsubseteq D$ where C and D are concept descriptions, and $d_n(\beta')$ is defined as $C \sqcap \neg\{a_1\} \sqcap \ldots \sqcap \neg\{a_n\} \sqsubseteq D$. From the semantics of inclusion axioms and nominals, it then follows that $C \sqcap \neg\{a_1\} \sqcap \ldots \sqcap \neg\{a_n\} \sqsubseteq D \in Cn(\{C \sqsubseteq D\})$. Hence, we have $\beta' \in Cn(\beta)$ where $\beta \in A$.

(Relevance') There are two cases to consider: (1) $\alpha \in Cn(\emptyset)$; and (2) $\alpha \notin Cn(\emptyset)$. For (1), it is easy to see that it satisfies the postulate because the precondition of the postulate is not met (as in the classical case). For (2), since $\alpha \notin Cn(\emptyset)$, we know that $\gamma(A\bot_{dl}\alpha)$ is a non-empty subset of $A\bot_{dl}\alpha$. If $\beta \in A$ and $\beta \notin A \div \alpha$, then $\beta \notin \bigcap \gamma(A\bot_{dl}\alpha)$. Hence, there exists $A' \in \gamma(A \div \alpha)$ such that $\beta \notin A'$. It then follows that $A' \in A\bot_{dl}\alpha$, so $\alpha \notin Cn(A')$ by Definition 3.6. Now, using the fact that $A' \in A\bot_{dl}\alpha$ and $A' \sqsubset A' \cup \{\beta\}\hat{\sqsubseteq}A$, we can derive that $\alpha \in Cn(A' \cup \{\beta\})$. Therefore, it satisfies relevance'.

(Uniformity) This is the same as the classical case and the proof for this is available in [Han99].

We now show that an operator $\div$ that satisfies the four postulates above is a partial meet contraction. We adapt from the proof of partial meet contraction in [Han99]. We

let γ be as follows: (1) If $A\perp\alpha \neq \emptyset$, then $\gamma(A\perp\alpha) = \{X \in A\perp\alpha | A\div \subseteq X\}$; (2) If $A\perp\alpha = \emptyset$, then $\gamma(A\perp\alpha) = \{A\}$. There are three parts to this proof: (1) show that γ is a function; (2) show that γ is a selection function; and (3) show that for all α, $\bigcap\gamma(A\perp\alpha) = A \div \alpha$. It is easy to see that (1) can be shown the same way as classical partial meet contraction since we did not have to revise the uniformity postulate. We can make use of *uniformity* the same way to show that γ is a well-defined function. For (2), we need to show that $A\perp_{dl}\alpha$ is non-empty implies $\gamma(A\perp_{dl}\alpha)$ is also non-empty. Suppose$A\perp_{dl}\alpha$ is non-empty, then $\alpha \notin Cn(\emptyset)$ by definition of γ. Since $\alpha \notin Cn(\emptyset)$, it follows from *success* that $\alpha \notin Cn(A \div \alpha)$. It also follows from *inclusion'* that $A \div \alpha \hat{\subseteq} A$. It then follows from the revised upper bound property[2] that $X \hat{\subseteq} X' \in A\perp\alpha$. Hence, we have $\gamma(A\perp_{dl}\alpha)$ is also non-empty as required by definition of γ. For (3), the proof is analagously to that of classical partial meet contraction for belief bases.

□

3.5.3 Kernel Contraction

Kernel contraction represents another stream of research that is complementary to partial meet contraction. As opposed to the use of remainder sets where the idea was to directly collect maximal sets of sentences that do not imply α and using a selection function to pick important subsets, the main tool in kernel contraction is the construction of a kernel that finds all minimal subsets of the original set of sentences A that imply α. In other words, the kernel defines sets of sentences that are to be removed from A and an incision function is used to further filter the sentences. The definition of a kernel is provided below.

It states that X is a subset of A, X implies α, and X is a minimal subset of A that implies α. That is, any proper subset of X does not imply α.

Definition 3.12 (Kernel [Han94])
Let A be a set in $\mathcal{L}$ and α a sentence. Then $A \perp\!\!\!\perp \alpha$ is the set such that $X \in A \perp\!\!\!\perp \alpha$ if and only if:

[2]Our revised upper bound property is the same as the classical upper bound property but replaces the $\subseteq$ relation with the $\hat{\subseteq}$ relation.

(1) $X \subseteq A$

(2) $X \vdash \alpha$, *and*

(3) If $Y \subset X$, *then* $Y \nvdash \alpha$.

Next, we consider the incision function introduced in [Han94]. Eq. 3.2 requires that $\sigma(A \perp\!\!\!\perp \alpha)$ be a subset of the union of all sets in $A \perp\!\!\!\perp \alpha$. Eq. 3.3 enforces that B be overlapping with $\sigma(A \perp\!\!\!\perp \alpha)$ for all sets $B \in A \perp\!\!\!\perp \alpha$ that are non-empty.

Definition 3.13 (Incision Function [Han94])
Let A be a set of sentences. Let $A \perp\!\!\!\perp \alpha$ be the kernel set of A with respect to α. An incision function σ for A is a function such that for all sentences α:

$$\sigma(A \perp\!\!\!\perp \alpha) \subseteq \bigcup (A \perp\!\!\!\perp \alpha) \tag{3.2}$$

$$\emptyset \neq B \in A \perp\!\!\!\perp \alpha, \text{ then } B \cap \sigma(A \perp\!\!\!\perp \alpha) \neq \emptyset \tag{3.3}$$

Definition 3.14 ([Han94])
Let A be a set of sentences and σ an incision function for A. The kernel contraction $-_\sigma$ for A is defined as follows:

$$A -_\sigma \alpha = A \setminus \sigma(A \perp\!\!\!\perp \alpha) \tag{3.4}$$

It is important to note that kernel contraction requires the use of set subtraction to remove elements from the original set of sentences. This makes it difficult to introduce a similar refinement approach with kernel contraction as we did in Definition 3.6, because weakened sentences are not necessarily elements of the original set of sentences. Therefore one would have to allow for subtraction of weakened sentences. This will be explored in our future work. There are several techniques [RW09, QHH+08] that have been proposed in the literature that deal with the use of kernel contraction as a way to revise a description logic knowledge base.

In particular, Qi et al. [QHH+08] proposed a kernel revision operator for TBox axioms. One of the main contributions of this work is the introduction of a cardinality-minimal incision function, which is an incision function designed to satisfy a number of postulates in the corresponding revision operator. The authors have also presented a number of incision functions formulated using Reiter's Hitting Set Tree (HST) algorithm. One of these algorithms was based on a scoring function and two others based on confidence values. In contrast to our approach, we do not make explicit use of the HST algorithm in our contraction operator, and also our approach presents finer-grained results than their approach. Their approach works on the level of TBox sentences and treats each TBox sentence as the finest unit for removal (i.e., a TBox sentence is either completely removed or retained, but not weakened).

3.5.4 Multiple Contraction

The idea of multiple contraction was explored by Hansson et al. [FH94] and its modification multiple kernel contraction by Ferme et al. [FSS03]. The problem of multiple contraction considers contraction of a set of sentences instead of classical contraction where only a single sentence is being contracted at each operation. This is particularly useful for description logics because conjunction of DL sentences cannot be expressed and it is generally not possible to combine multiple sentences into a single one. It means that one will have to perform a sequence of contraction operations on the set of sentences. For example, it is syntactically invalid to create a sentence of the form $C(s) \wedge R(s,t)$ that would allow us to perform $K \div C(s) \wedge R(s,t)$. Further, the result obtained from $K \div C(s) \wedge \{C(s), R(s,t)\}$ and $K \div \{C(s), R(s,t)\}$ could be quite different sometimes because the latter leads to more sentences being deleted. To contract both $C(s)$ and $R(s,t)$ one would have to contract $C(s)$ and $R(s,t)$ on independent operations one after the other in some order. In many cases, contraction by removing simultaneously a set of sentences in one operation and removing a set of sentences in a sequence of contraction operations could lead to very different results.

Multiple contraction requires to define what exactly is meant by contracting by a set of sentences. There are two known notions: package contraction and choice contraction.

The former concerns with the consistent and simultaneous removal of every sentence in the set. The latter requires at least one sentence from the set to be removed.

These two variants lead to two methodologies to handle multiple contraction. We define below the corresponding DL versions of package remainder set and choice remainder set [FH94] based on Def. 3.6. The definition of package and choice selection functions are also adopted into our context.

Definition 3.15 (Package Remainder Set for DL)

Let A and B be sets of DL sentences. $A\perp_{dl}B$ is the set such that $X \in A\perp_{dl}B$ if and only if:

1. $X \in d_i(A)$ for some i.

2. $B \cap Cn(X) = \emptyset$.

3. There is no B'' such that $B'' \in d_i(A)$ and $X \in d_j(B'')$ for some $0 \leq i$ and $0 < j$, and $X \cap Cn(B'') = \emptyset$.

Definition 3.16 (Choice Remainder Set for DL)

Let A and B be sets of DL sentences. $A\angle_{dl}B$ is the set such that $X \in A\angle_{dl}B$ if and only if:

1. $X \in d_i(A)$ for some i.

2. $B \not\subseteq Cn(X)$.

3. There is no B'' such that $B'' \in d_i(A)$ and $X \in d_j(B'')$ for some $0 \leq i$ and $0 < j$, and $X \not\subseteq Cn(B'')$.

Similar to classical multiple contraction, the package remainder set for DL aims to remove all the sentences in the set B, whereas the choice remainder set for DL aims to remove only one of the sentences in B. Just like partial meet contraction, we make use of a selection function γ that chooses from the package or choice remainder set for DL the sentences that are most worth retaining.

Definition 3.17

γ is a package selection function for A if and only if for all sets B:
(1) If $A\perp_{dl}B$ is non-empty, then $\gamma(A\perp_{dl}B)$ is a non-empty subset of $A\perp_{dl}B$.
(2) If $A\perp_{dl}B$ is empty, then $\gamma(A\perp_{dl}B) = A$.

Definition 3.18

γ is a choice selection function for A if and only if for all sets B:
(1) If $A\angle_{dl}B$ is non-empty, then $\gamma(A\angle_{dl}B)$ is a non-empty subset of $A\angle_{dl}B$.
(2) If $A\angle_{dl}B$ is empty, then $\gamma(A\angle_{dl}B) = A$.

Similar to contraction by a single sentence we are able to show some properties of our package and choice remainder set for DL. These properties are adapted from [FH94].

Definition 3.19

An operator $\div$ for a set A is an operator of partial meet package contraction if and only if it satisfies the following conditions:

(P-success) *If $B \cap Cn(\emptyset) = \emptyset$ then $B \cap Cn(A \div B) = \emptyset$.*

(P-inclusion') *If $\beta' \in A \div B$, then $\beta' \in Cn(\beta)$ for some $\beta \in A$.*

(P-relevance') *If $\beta \in A$ and $\beta \notin A \div \alpha$, then there is a set A' such that for all $\delta' \in A', \delta' \in Cn(\{\delta\})$ for some $\delta \in A$, and $B \cap Cn(A') = \emptyset$ but $B \cap Cn(A' \cup \{\beta\}) \neq \emptyset$.*

(P-uniformity') *If every set of sentences A', such that for all $\beta \in A'$, $\beta' \in Cn(\{\beta\})$ for some $\beta \in A$, implies some element of B if and only if A' implies some element of C, then $A \div B = A \div C$.*

Lemma 3.20

Package remainder set for DL (Definition 3.15) satisfies P-success, P-inclusion', P-relevance' and P-uniformity'.

Proof. (P-success) This proof follows immediately from Definition 3.15.
(P-inclusion') Suppose $\beta' \in A \div B$. Then by Definition, $\beta' \in \bigcap \gamma(A \perp_{dl} B)$. There are two cases to consider: (1) $A \perp_{dl} B$ is an empty set. (2) $A \perp_{dl} B$ is a *not* an empty set. For (1), since $A \perp_{dl} B$ is an empty set, we know that $\gamma(A \perp_{dl} B) = A = \bigcap \gamma(A \perp_{dl} B) = A \div B$. It follows immediately that $\beta' \in Cn(\beta)$ for some $\beta \in A$. For (2), since $A \perp_{dl} B$ is not an empty set, we know that there is some set A' such that $A \div B \subseteq A'$, $A' \in A \perp_{dl} B$ and $\beta' \in A'$. Since $A' \in A \perp_{dl} B$, by definition 3.15 $A' \in d_i(A)$. Hence $\beta' \in A'$ meaning $\beta' \in Cn(\beta)$ for some $\beta \in A$.
(P-relevance') Suppose $\beta \in A$ and $\beta \notin A \div \alpha$. Then there exists $\beta' \in A \div \alpha$ such that $\beta' \in Cn(\{\beta\})$. Since $\beta' \in A \div \alpha$, by Definition 3.15 there is a set A' such that $\beta' \in A'$ and $B \cap Cn(A') = \emptyset$. By Definition 3.15, we know that $B \cap Cn(A' \cup \{\beta\}) \neq \emptyset$.
(P-uniformity') Suppose a weakened set of sentences A' is one such that for all $\beta' \in A'$, $\beta' \in Cn(\beta)$ for some $\beta \in A$. Also suppose every weakened set A' implies some element of B if and only if A' implies some element of C. By Definition 3.15, we know that each element $A' \in d^i(A)$ (for any i) is a weakened set of sentences. Therefore, A' implies some element of B if and only if A' implies some element of C. If $\beta' \in A \div B$, then it follows that $\beta' \in \bigcap \gamma(A \perp B)$. There are two cases to consider: (1) $A \perp B$ is an empty set, and (2) $A \perp B$ is *not* an empty set. For (1), if $A \perp B$ is an empty set, then $\gamma(A \perp B) = A$, and so $\beta' \in A$. Since $A \perp B$, so by Definition 3.15, we know that there is no weakened set A' that implies any element of B, so it follows from our assumption that no weakened set implies any element of C. Therefore, $A \perp C$ is also an empty set, so $\gamma(A \perp C) = A = A \div C$. Since $\beta' \in A$, hence $\beta' \in A \div C$. Case (2) can be shown similarly.

□

Definition 3.21

An operator $\div$ for a set A is an operator of partial meet choice contraction if and only if it satisfies the following conditions:

(C-success) *If $B \cap Cn(\emptyset) = \emptyset$ then $B \cap Cn(A \div B) = \emptyset$.*

(C-inclusion') *If* $\beta' \in A \div B$*, then* $\beta' \in Cn(\{\beta\})$ *for some* $\beta \in A$*.*

(C-relevance') *If* $\beta \in A$ *and* $\beta \notin A \div \alpha$*, then there is a set* A' *such that for all* $\delta' \in A', \delta' \in Cn(\{\delta\})$ *for some* $\delta \in A$*, and* $B \not\subseteq Cn(A')$ *but* $B \subseteq Cn(A' \cup \{\beta\})$*.*

(C-uniformity') *If every set of sentences* A'*, such that for all* $\beta \in A'$*,* $\beta' \in Cn(\{\beta\})$ *for some* $\beta \in A$*, implies some element of* B *if and only if* A' *implies some element of* C*, then* $A \div B = A \div C$*.*

Lemma 3.22

Partial meet choice contraction for DL (Definition 3.16) satisfies C-success, C-inclusion', C-relevance' and C-uniformity'.

Proof. This proof is analagous to that of Lemma 3.20 □

3.6 Ontology Revision

In this section, we demonstrate the close relationship between ontology contraction and ontology revision, based on the established notions in the belief change literature. More particularly, we show how one might be able to achieve ontology revision from the contraction operators we proposed.

Revision concerns with the inconsistent incorporation of new information. In the AGM model, a typical strategy is to revise a set of sentences A with a single sentence α by first contracting the negation of α and then adding α by the simple expansion operation.

This formulation is known as the Levi identity [Lev77, Han99]:

$$K * \alpha = (K \div \neg\alpha) + \alpha$$

The Levi identity allows revision to be formulated in terms of contraction ($\div$) and expansion ($+$). Loosely speaking, the Levi identity states that revising a knowledge base K by a sentence α is the same as first contracting K by the negation of α (i.e., $\neg\alpha$)

then expanding the result with α. This, however, cannot always be easily achieved in DL where classical negation of DL sentences is generally in absence. More particularly, negation of TBox sentences such as $C \sqsubseteq D$ and $C \equiv D$ are not valid DL constructs, though we could derive from the semantics what their negations might be. For example, from the semantics we know that $C \sqsubseteq D$ can be expressed as $\top \sqsubseteq (\neg C \sqcup D)$ (assuming that negation of concepts is defined in the language) so its negation can be expressed as $\top \sqsubseteq (C \sqcap \neg D)$.

One approach to overcome this challenge is by defining negation of DL sentences, either by introducing new constructs or by identifying ways to make use of existing constructs to produce a form of negation.

3.6.1 Revision with Negation

In [FHP+06], the authors proposed two negation conditions. The negation conditions below generalises the requirements for using the Levi identity as a way to obtain a revision operator from a contraction operator. The two negation conditions are very similar to each other, except that one is designed for consistency while the other for coherence.

Definition 3.23 (Consistency-Negation Flouris et al. [FHP+06])

An axiom ψ is said to be a consistency-negation *of an axiom ϕ, written $\psi = \neg\phi$, if and only if:*
(1) (Inconsistency) $\{\phi, \psi\}$ is inconsistent;
(2) (Minimality) There exists no other ψ' such that ψ' satisfies condition (1) and $Cn(\{\psi'\}) \subset Cn(\{\psi\})$.

It is easy to see that this is a weaker notion of negation than classical negation. Hence, we know that concept assertions such as $\neg C(s)$ would satisfy the condition. Negation of concepts is present in DLs such as $\mathcal{ALC}$ and its extensions (e.g., $\mathcal{SHOIQ}$ and $\mathcal{SROIQ}$). However, DLs from the class of $\mathcal{EL}$ do not generally define negations on concepts (or have only limited forms of negation such as negation of atomic concepts).

Definition 3.24 (Coherence-Negation Flouris et al. [FHP$^+$06])

An axiom ψ is said to be a coherence-negation *of an axiom ϕ, written $\psi = \neg\phi$, if and only if:*
(1) (Inconherence) $\{\phi, \psi\}$ is incoherent;
(2) (Minimality) There exists no other ψ' such that ψ' satisfies condition (1) and $Cn(\{\psi'\}) \subset Cn(\{\psi\})$.

3.6.2 Revision without Negation

An alternative method to revise a set of DL sentences is by using the reversed Levi identity. The idea is that, instead of retracting the negation of α from A and then adding α (the Levi identity), one could add α and then retract the negation of α.

$$K * \alpha = (K + \alpha) \div \neg\alpha$$

Note that, this approach is exclusively for belief bases and *not* for belief sets. An expansion that leads to an inconsistency would invalidate a belief set, as the belief set would contain all sentences. A revision approach was applied in the context of DL in [RW09]. Instead of contracting by $\neg\alpha$ in the reversed Levi identity, it can be replaced with $\bot$. The resulting set of sentences will necessary be consistent as long as the contraction operator satisfies the success postulate.

3.7 Summary and Discussion

Ontology change is a challenging problem, yet it is foundational to the development of Semantic Technologies. We presented in this chapter numerous techniques that are applicable to addressing ontology change. In particular, we studied the notion of classical remainder set and showed by example that it is directly applicable to DL. However, we argued that this would lead to counter-intuitive results because classical reminder set

fails to capture the structure of DL sentences properly. In response to this, we introduced our original notion of exceptions and presented the notion of remainder set for DL based on exceptions. We showed that this would lead to more desirable results. Furthermore, we investigated our notion of remainder set further to produce a partial meet contraction similar to that of the classical remainder set. However, we discovered that the remainder set for DL does not satisfy the classical partial meet contraction postulates for belief bases. More specifically, it does not satisfy the inclusion and relevance postulates. Therefore, we provided our original revised contraction postulates and showed that our notion of remainder set is able to satisfy them.

In the next chapter, we investigate the problem of ontology integration. More specifically, we consider stratified knowledge bases where each stratum is a set of sentences and the goal is to produce a single consistent set of sentences.

Chapter 4

Ontology Integration

4.1 Chapter Introduction

In this chapter, we investigate the problem of knowledge integration in the description logic context. In propositional logic, the problem of knowledge integration amounts to integrating multiple sources of information that may not be logically consistent with each other. In order for these sources to be successfully integrated, it is necessary to resolve the inconsistent information. Analogously, we consider the problem of ontology integration in DLs where ontologies may be inconsistent with each other, and conflict resolution strategies are necessary to ensure successful integration. Analogously to propositional knowledge integration, we adhere to the principle of minimal change as the underlying criterion for successful integration, where loss of information is to be kept to a minimum. We propose to recast techniques that have been extensively studied in propositional knowledge integration to the description logic context. We focus our study to a class of conflict resolution strategies known as adjustments [BKLBW04], which is known to possess desirable properties. For example, they satisfy the AGM revision postulates [BKLBW04]. We present a variant of the (whole) Disjunctive Maxi-Adjustment (DMA) called Conjunctive Maxi-Adjustment (CMA) [MLB05]. We then show that the additional structure of description logics can be exploited to produce finer-grained adjustment strategies and we propose two versions of adjustments for Description Logics (DL). One is a direct translation of Conjunctive Maxi-Adjustment (CMA) into the DL

context, while the other is a refinement of the CMA strategy called RCMA-DL that takes into account the structure of both axioms and assertions in DL.

4.2 Motivation

The problem of ontology integration is fundamental to the development of the Semantic Web and its applications to real-world systems. To elaborate on this problem, consider a typical lifecycle of an ontology. In many cases, an ontology is created manually by knowledge engineers (e.g., using Protege) or it is translated and formalised from existing data sources (e.g., semi-structured text). At some stage during the ontology lifecycle, the ontology is updated with newly arrived information for various reasons. For example, to refine the modelling of the domain of interest or to broaden the scope of the model. As we have discussed in earlier chapters, new information can arrive in different forms. They can arrive sentence by sentence (as in the case of using an ontology editor to update an ontology) or many sentences can arrive at the same time (e.g., in the form of an existing ontology). The latter is more challenging because it is more likely to introduce logical errors and it is a time-consuming and error-prone process. Therefore, automating knowledge integration is essential and our goal is to study the ontology integration operation in a theoretically rigorous manner. In particular, we consider approaches from propositional belief integration and incorporate them into the context of description logic.

Adjustments

Adjustment strategies [BKLBW04] are used as a means to solve the knowledge integration problem in the propositional context. The knowledge integration problem is usually considered in a setting where there is a stratified knowledge base K and a sentence ϕ. A stratified knowledge base $K = (S_1, \ldots, S_n)$ consists of n levels, where each level S_i is a set of sentences called stratum. The index i attached to each stratum denotes the degree of reliability with respect to other strata in K. Sentences in S_i are more reliable than sentences in S_j if and only if $i < j$. Sentences in the same stratum are equally

reliable. The goal of the problem is to incorporate the sentence ϕ into the stratified knowledge base K and produce a single consistent set of sentences. The sentence ϕ is assumed to be always present in the resulting set of sentences, unless ϕ itself is inconsistent. The general approach is to maintain a set of consistent sentences $\mathcal{B}$ while traversing through each stratum S_i of the stratified knowledge base from i to n. For each stratum S_i, a set of sentences S_i' that satisfies $\forall \psi \in S_i', S_i \models \psi$ is derived from S_i and is added to $\mathcal{B}$ only if they are consistent with each other. In other words, the resulting set of sentences $\mathcal{B}$ is always consistent. The fundamental difference between the various adjustment strategies is in the way S_i' is being derived for each stratum S_i. In Adjustment and Maxi-Adjustment [BKLBW04], sentences in S_i are simply discarded to derive S_i'. That is, it satisfies $S_i' \subseteq S_i$. A more finer-grained approach is to consider weakening sentences instead of discarding them.

Disjunctive Maxi-Adjustment (DMA) [BKLBW04] is an example of this approach. The idea of DMA is to weaken sentences by taking disjunctions of the original set of sentences. Initially, DMA considers disjunctions of pairs of sentences. If the result turns out to be consistent then the algorithm moves to the next stratum, otherwise it continues to weaken the set of sentences by considering disjunctions of three sentences and then four and five and so on. Eventually, the algorithm reaches a point where there is no conflict because the set of sentences will eventually be weakened to $\top$. Note also that this approach will necessarily terminate since the input is a finite set of sentences.

4.3 Related Work

One of the first attempts to deal with inconsistency in logic-based terminological systems can be found in [Neb90], where it is phrased as a belief revision problem. More recently the solution of [SC03] is to provide support for inconsistency by correcting it. They propose a non-standard reasoning system for debugging inconsistent terminologies. The idea is to provide an explanation by pinpointing the source of the inconsistency, while correction is left to human experts. In contrast, the approach taken in [HVHT05] assumes that ontology reparation will be too difficult. They propose to tolerate inconsistency and

apply a non-classical form of inference to obtain meaningful results. Our approach is a hybrid of these. We employ a version of lexicographic entailment to determine the consequences of an inconsistent DL knowledge base, with the original knowledge base also weakened so that its classical consequences correspond exactly to the non-monotonic consequences of the original knowledge base. In [QR92] a technique is described for assigning a preference semantics for defaults in terminological logics. This technique uses exceptions, and therefore has some similarities to our work. They draw a distinction between strict inclusions (TBox statements of the form $A \sqsubseteq B$) and defaults, which are interpreted as "soft" inclusions. In our framework, this distinction can be modelled with two strata in which all strict inclusions occur in S_1 and all soft inclusions in S_2. In this sense our framework is more expressive than theirs. More importantly, their formal semantics is not cardinality-based, and therefore yields quite different results from ours. And finally, unlike our framework, their method does not provide weakening of the original knowledge base.

A completely different approach is the explicit introduction of nonmonotonicity into DLs, usually some variant of default logic. See [BKW03] for an overview. While it is difficult to draw direct comparisons with our work, it is similar intuitions might be identified and exploited.

4.4 Propositional Knowledge Integration

Propositional knowledge integration, as described in [BKLBW04], takes as input a stratified knowledge base $K = (S_1, \ldots, S_n)$ where, for $i \in \{1, \ldots, n\}$, S_i is a knowledge base, or a finite set of propositional sentences (of a finitely generated propositional logic). Sentences in a stratum S_i are all judeged to be of equal reliability, while sentences contained in a higher stratum, i.e. in any S_j for $j > i$, are seen as less reliable. In [BKLBW04] the strategies proposed to minimise the loss of information that occurs when a stratified knowledge base is inconsistent are shown to yield identical results to the lexicographic system for knowledge integration [BCD$^+$93]. It is well-known that lexicographic entailment is a versatile system with desirable theoretical properties. For example, it has

been shown in [GSK00] that it can be used to model all classical AGM belief revision operators. We provide here an alternative semantic characterisation of lexicographic entailment based on the notion of exceptions. The advantage of this characterisation is that it makes a clear distinction between the following two distinct principles at work:

1. **Independence** - sentences in a stratum are assumed to have been obtained independently

2. **Precedence** - More reliable information should take complete precedence over less reliable information

Independence is used implicitly in the work of [BKLBW04]. It is applied to sentences in each stratum S_i and is formalised in terms of exceptions. The number of S_i-exceptions relative to valuation v is the number of sentences in S_i false in v; the fewer S_i-exceptions, the more preferred v is.

4.4.1 Propositional Lexicographic Entailment

Here we define the notion of lexicographic entailment in the propositional context. We do this by introducing a way to make comparison between valuations. This allows us to create an ordering on the valuations of a formula and in turns to sets of formulas. We then extend this definition to create the lexicographic ordering on valuations for stratified knowledge bases, where each stratum in the knowledge base is a finite set of propositional formulas.

Note that $M(\phi)$ denotes the set of all valuations for a propositional formula ϕ.

Definition 4.1 (Ordering Propositional Interpretations)
The number of ϕ-exceptions $e^{\phi}(v)$ for a valuation v is 0 if $v \in M(\phi)$ and 1 otherwise. For a finite set of sentences Φ, the number of Φ-exceptions for a valuation v is $e^{\Phi}(v) = \sum_{\phi \in \Phi} e^{\phi}(v)$. The ordering $\preceq_{\Phi}$ on a set of valuations V is defined as:

$$v \preceq_{\Phi} w \text{ if and only if } e^{\Phi}(v) \leq e^{\Phi}(w)$$

It is only after the Independence principle has been applied to all strata that the

Precedence principle is applied to the orderings associated with the different strata to obtain a lexicographically combined preference ordering.

Observation 4.2

Let Φ be a set of sentences. The number of Φ-exceptions of v, written $e^{\Phi}(v)$, satisfies $0 \leq e^{\Phi}(v) \leq n$, where $n = |\Phi|$.

Proof. This proposition follows immediately from Def.4.1. Let $\Phi = \{\phi_1, \ldots, \phi_n\}$, where $n = |\Phi|$. By Def. 4.1, we have $e^{\phi_i}(v) \in \{0, 1\}$ for all i where $1 \leq i \leq n$. Therefore, we have $0 \leq e^{\Phi}(v) \leq n$. □

Definition 4.3 (Lexicographic Ordering)

Let $K = (S_1, \ldots, S_n)$ be a stratified knowledge base, and let $\preceq_{S_1}, \ldots, \preceq_{S_n}$ be the total preorders for each stratum of K on the set of valuations V. The lexicographic ordering on the set of valuations V, written $v \preceq_{lex} w$, is defined as follows:

$$v \preceq_{lex} w \text{ if and only if } \forall j \in \{1, \ldots, n\}, [v \preceq_{S_j} w \text{ or } v \prec_{S_i} w \text{ for some } i < j].$$

Alternatively, $v \preceq_{lex} w$ if and only if $(\exists i > 0)(\forall j < m)(v \approx_{S_i} w) \wedge (v \prec_{S_i} w)$ where $v \approx_{S_i} w$ if and only if $v \preceq_{S_i} w$ and $w \preceq_{S_i} v$.

Definition 4.4 (Lexicographic Entailment)

A stratified knowledge base K lexicographically entails ϕ, written $K \models_{lex} \phi$, if and only if the $(\preceq_{lex})$-minimal models satisfy ϕ. A valuation v is a $(\preceq_{lex})$-minimal model if and only if $v \preceq_{lex} w$ for all w.

The view of lexicographic entailment as an application of Independence and Precedence is also present in a new strategy for knowledge integration, conjunctive maxi-adjustment or CMA, that we propose in Algorithm 2. The idea is to work through K stratum by stratum in order of decreasing precedence, and to construct a consistent knowledge base B, adding as many sentences as possible while maintaining consistency, and weakening those strata responsible for inconsistencies. In this sense it is similar

to whole disjunctive maxi-adjustment [BKLBW04], but the way in which strata are weakened is (syntactically) different. With CMA, if S_i is inconsistent with part of B constructed so far, it is replaced by the disjunction of the cardinality-maximal conjunctions of S_i-formulas consistent with B.

4.4.2 Conjunctive Maxi-Adjustment

The Conjunctive Maxi-Adjustment procedure that we are presenting below is a variation of the Disjunctive Maxi-Adjustment [BKLBW04]. Specifically, it is a variation of the whole DMA strategy.

input : $K = (S_1, \ldots, S_n)$
output: A consistent classical KB
$\mathcal{B}_0 \leftarrow \emptyset$;
for $i \leftarrow 1$ **to** n **do**
  $j \leftarrow 0$;
  repeat
    $\phi \leftarrow \bigvee$ of all $\bigwedge$s of size $(|S_i| - j)$ of formulas of S_i;
    $\mathcal{W}_{ij} \leftarrow \mathcal{B}_{i-1} \cup \{\phi\}$;
    $j \leftarrow j + 1$;
  until $\mathcal{W}_{ij}$ *is consistent*;
  $\mathcal{B}_i \leftarrow \mathcal{W}_{ij}$;
end
return $\mathcal{B}_n$

Algorithm 2: The Conjunctive Maxi-Adjustment (CMA) Algorithm.

It is easily verified that Algorithm 2 always terminates.

Furthermore, it produces results that are equivalent to lexicographic entailment (i.e., B corresponds to the sentences that are most highly ranked based on the lexicographic ordering), and therefore also to the strategies discussed in [BKLBW04].

Example 4.5

Below is an example that demonstrates the CMA algorithm shown in Algorithm 2.

$$S_1 = \{\neg(p \wedge q), \neg(q \wedge r), \neg(p \wedge r)\}$$

$$S_2 = \{p, q, r\}$$

Let $K = (S_1, S_2)$ be a stratified propositional knowledge base, where S_1 and S_2 are shown above. We begin by setting $\mathcal{B}_0$ to $\emptyset$. We then enter the most outer for loop and visit the strata from the one with the highest preference to the least. When i is 1, we set j to 0 and consider conjunctions of $|S_1| - j = 3$ elements, hence we set ϕ to be the conjunction of all the elements in S_1. That is, ϕ is set to $\neg(p \wedge q) \wedge \neg(q \wedge r) \wedge \neg(p \wedge r)$. Since ϕ is consistent with $\mathcal{B}_0$, we simply set $\mathcal{B}_1$ to $\mathcal{B}_0 \cup \{\phi\}$. When i is 2, we set j to 0 and consider conjunctions of $|S_2| - j = 3$. Hence we set ϕ to be the conjunction of all the elements in S_2 and we have ϕ set to $p \wedge q \wedge r$. Since ϕ is inconsistent with $\mathcal{B}_1$, we cannot break out of the repeat loop. Instead we increment j to 1 and consider the disjunction of all the conjunctions of size $|S_2| - j = 2$. Thus, we have ϕ set to $(p \wedge q) \vee (p \wedge r) \vee (q \wedge r)$. However, ϕ is again inconsistent with $\mathcal{B}_1$ and so we increment j again to 2. We consider the disjunction of all the conjunctions of size $|S_2| - j = 1$, hence we set ϕ to $p \vee q \vee r$. Since ϕ is now consistent with $\mathcal{B}_1$, so we break out of the repeat loop and we set B_2 to $\{\neg(p \wedge q) \wedge \neg(q \wedge r) \wedge \neg(p \wedge r),\ (p \vee q \vee r)\}$. We then break out of the outer loop and return B_2.

Proposition 4.6 *Let $K = \{S_1, \ldots, S_n\}$ be a stratified propositional knowledge base. The CMA algorithm in Algorithm 2 generates a consistent knowledge base $\mathcal{B}_n$ with at most $\sum_{i=1}^{n} |S_i|$ consistency checks.*

Proof. (sketch) Consider the most inner repeat loop in the CMA algorithm where weakenings of the formulas in S_i are being used to construct ϕ. It should be noted that the way CMA constructs $\mathcal{B}_{i-1}$ ensures that it is always consistent. For the first iteration, ϕ is set to the disjunction of the conjunctions of size $|S_i|$ of S_i. If $|S_i| > 0$ then ϕ is simply the conjunction of all the elements in S_i, and we check whether this $\mathcal{B}_{i-1} \cup \phi$ is consistent. If it is, we can then work on the next stratum, otherwise we consider further weakening of the same stratum. For the second iteration, ϕ is set to the disjunction of the conjunctions of size $|S_i| - 1$ of S_i and so on. Since we are considering conjunctions with one element less each time we repeat the loop, each stratum can be weakened for no more than $|S_i|$ times. The size of the conjunctions will eventually reach 0, and so ϕ

will be set to $\top$. We then break out of the repeat loop, since $\mathcal{B}_{i-1} \cup \top$ is always consistent if $\mathcal{B}_{i-1}$ is consistent. Therefore, each stratum S_i requires no more $|S_i|$ consistency checks. In total, we consider n strata, so the algorithm will terminate and return a set of consistent sentences in no more than $\sum_{i=1}^{n} |S_i|$ consistency checks. □

It should be noted that the above proposition may *not* necessarily be a good indicator of computational time since the formulas generated by the algorithm can be exponential to the size of the input sentences.

The principal reason for the introduction of CMA is that the two algorithms for knowledge integration described in the next section are natural extensions of it.

We modify our definition of a stratified knowledge base so that each stratum is a multiset of sentences, denoted by square brackets (so a stratum is of the form $[\phi_1, \ldots, \phi_n]$).

This allows us to prove a result (Corollary 4.10) that does not hold if strata are represented as sets.

Properties of Conjunctive Maxi-Adjustment

Conjunctive Max-Adjustment has a number of interesting properties. For convenience, we introduce here some notations which we will use in the proofs. We define $\delta^i_{cma}(\Phi)$ as the result of the i^{th} weakening of a set of sentences Φ. For example, if $\Phi = \{p, q, r\}$, then we have the following:

$$\delta^0_{cma}(\Phi) = p \wedge q \wedge r$$
$$\delta^1_{cma}(\Phi) = (p \wedge q) \vee (p \wedge r) \vee (q \wedge r)$$
$$\delta^2_{cma}(\Phi) = p \vee q \vee r$$
$$\delta^3_{cma}(\Phi) = \top$$

Furthermore, we define $\delta_{cma}(\Phi, K) = \delta^i_{cma}(\Phi)$ where $\delta^i_{cma}(\Phi)$ is consistent with K and $\delta^j_{cma}(\Phi)$ is inconsistent with K for all $j < i$. In other words, if we produce $\delta^i_{cma}(\Phi)$ for $i = 1 \rightarrow n$, then $\delta_{cma}(\Phi, K)$ corresponds to the first weakened sentence of Φ that is

consistent with K. It should be noted that $\delta_{cma}(\Phi, K)$ exists for any set of sentences Φ if K is consistent since $\delta_{cma}^{|\Phi|}(\Phi) = \top$.

In a similar manner, we define $\delta_{dma}^{i}(\Phi)$ as the result of the i^{th} weakening of a set of sentences Φ. However, the set of sentences in Φ are weakened differently from CMA. For DMA, we first consider all clauses of size 1 and take the conjunction of these clauses. This forms the first level of our weakening or $\delta_{dma}^{0}(\Phi)$. If this is not consistent with K then we consider a further weakening by taking the conjunction of all clauses of size 2 (this forms $\delta_{dma}^{1}(\Phi)$). And then conjunction of clauses with size 3, and so on. The example below illustrates the construction of $\delta_{dma}^{i}(\Phi)$ for $i = \{0, \ldots, 3\}$. Note that, $\delta_{cma}^{i}(\Phi)$ and $\delta_{dma}^{i}(\Phi)$ are logically equivalent sentences but represented in different normal forms (Disjunctive Normal Form (DNF) and Conjunctive Normal Form (CNF) respectively).

$$\delta_{dma}^{0}(\Phi) = p \wedge q \wedge r$$
$$\delta_{dma}^{1}(\Phi) = (p \vee q) \wedge (p \vee r) \wedge (q \vee r)$$
$$\delta_{dma}^{2}(\Phi) = p \vee q \vee r$$
$$\delta_{dma}^{3}(\Phi) = \top$$

We define $\delta_{dma}(\Phi, K) = \delta_{dma}^{i}(\Phi)$ where $\delta_{dma}^{i}(\Phi)$ is consistent with K and $\delta_{dma}^{j}(\Phi)$ is inconsistent with K for all $j < i$.

CMA and DMA present two representations for adjustments. There are no obvious advantages of one representation over the other for propositional sentences.

Proposition 4.7 *Let Φ be a set of sentences. A valuation v satisfies $\delta_{cma}(\Phi, K)$ if and only if v satisfies $\delta_{dma}(\Phi, K)$. That is, $\delta_{cma}(\Phi, K) \equiv \delta_{dma}(\Phi, K)$.*

Proof. $\delta_{cma}(\Phi, K)$ is simply $\delta_{dma}(\Phi, K)$ in disjunctive normal form, and $\delta_{dma}(\Phi, K)$ is $\delta_{cma}(\Phi, K)$ in conjunctive normal form. □

Lemma 4.8, Lemma 4.9 and Corollary 4.10 establish the relationship between the classical lexicographic entailment operation and the notion of Conjunctive Maxi-Adjustment. Note that Corollary 4.10 is established using Lemma 4.8, Lemma 4.9.

Lemma 4.8

Let Φ be a set of sentences $\{\phi_1, \ldots, \phi_n\}$, where $|\Phi| = n$. A valuation v is a $(\preceq_\Phi)$-minimal model if and only if v satisfies $\delta_{cma}(\Phi)$.

Proof. We show in this proof the connection between a $(\preceq_\Phi)$-minimal model and the weakened sentence of a set of sentences Φ.

1. ($\Longrightarrow$) Suppose v is a $(\preceq_\Phi)$-minimal model. We need to show that v satisfies $\delta_{cma}(\Phi)$. This amounts to showing that there is some i, such that: (1) v satisfies $\delta^i_{cma}(\Phi)$ for some i, and (2) $\delta^j_{cma}(\Phi)$ is inconsistent for all $j < i$. That is, there does not exist an valuation such that w satisfies $\delta^j_{cma}(\Phi)$ for some $j < i$.

   Consider case (1). By Def.4.1, v satisfies all but $e^\Phi(v)$ elements in Φ, where $0 \leq e^\Phi(v) \leq |\Phi|$. That is, there exists a set $\Phi' \subseteq \Phi$ with $|\Phi'| = |\Phi| - e^\Phi(v)$ such that v satisfies all $\phi \in \Phi'$. It then follows that v satisfies the conjunction of all the sentences in Φ', i.e. v satisfies $\bigwedge_{\phi\in\Phi'} \phi$. By definition, $\delta^{e^\Phi(v)}_{cma}(\Phi)$ is the disjunction of all the conjunctions of size $|\Phi| - e^\Phi(v) = |\Phi'|$ of Φ, so $\bigwedge_{\phi\in\Phi'}$ is a disjunct of $\delta^{e^\Phi(v)}_{cma}(\Phi)$, and therefore v satisfies $\delta^{e^\Phi(v)}_{cma}(\Phi)$. Therefore (1) holds.

   Consider case (2). To show (2), suppose w satisfies $\delta^j_{cma}(\Phi)$ for some $j < e^\Phi(v)$. By definition, $\delta^j_{cma}(\Phi)$ is a disjunction that contains all the conjunctions of size $|\Phi| - j$ of Φ. For w to satisfy $\delta^j_{cma}(\Phi)$, w must satisfy at least one of its conjunctions. This conjunction would contain $|\Phi| - j$ elements, so w satisfies $|\Phi| - j$ elements of Φ. By Def. 4.1 and the fact that $j < e^\Phi(v)$, this means $\sum_{\phi\in S} e^\phi(w) < e^\Phi(v)$, hence $e^\Phi(w) < e^\Phi(v)$. So $w \prec v$. This is a contradiction since v is a $(\preceq_\Phi)$-minimal model. Thus, we have shown that (1) and (2) holds for $i = e^\Phi(v)$.

2. ($\Longleftarrow$) Suppose v satisfies $\delta_{cma}(\Phi)$, and v is *not* a $(\preceq_\Phi)$-minimal model. We know that $\delta_{cma}(\Phi) = \delta^i_{cma}(\Phi)$ for some i, and $\delta^j_{cma}(\Phi)$ is not consistent for all $j < i$. Since v satisfies $\delta_{cma}(\Phi)$, it also satisfies $\delta^i_{cma}(\Phi)$ for some i. That means v satisfies the disjunction of all the conjunctions of size $|\Phi| - i$, so it satisfies at least one of the conjunctions. It follows that v satisfies more than or equal to $|\Phi| - i$ elements in Φ. But since $\delta^j_{cma}(\Phi)$ is not consistent for all $j < i$, so v does not satisfy

$\delta^{j}_{cma}(\Phi)$ for all $j < i$. It means that v cannot satisfy more than $|\Phi| - i$ elements in Φ. Therefore v satisfies exactly $|\Phi| - i$ elements in Φ. By Def. 4.1, we have $e^{\Phi}(v) = i$. If v is not the $(\preceq_{\Phi})$-minimal model then there exists a valuation w such that $e^{\Phi}(w) < e^{\Phi}(v)$. That is, w satisfies more sentences of Φ than v, so w satisfies $\delta^{e^{\Phi}(w)}_{cma}(\Phi)$ where $e^{\Phi}(w) < e^{\Phi}(v)$. So $\delta^{e^{\Phi}(w)}_{cma}(\Phi)$ is consistent where $e^{\Phi}(w) < i$, which is a contradiction.

□

We show below the important correspondence between lexicographic ordering and the output of Algorithm 2. For convenience, we denote $\delta_{cma}(K)$ to be the output of Algorithm 2.

Lemma 4.9

Let K be a stratified knowledge base. A valuation v is a $(\preceq_{lex})$-minimal model of K if and only if v satisfies $\delta_{cma}(K)$.

Proof. Let $K = K_n$, $K_i = \{S_1, \ldots, S_i\}$ for $i > 0$, and $K_0 = \emptyset$. Also, let $(\preceq^{i}_{lex})$ be the lexicographic ordering of K_i, and $\delta_{cma}(K_i)$ be the output of K_i from Algorithm 2.

Consider $n = 1$. By Def. 4.3, we can derive that $(\preceq_{lex})$ of K_1 is equivalent to $(\preceq_{K_1})$. Also, we can derive that $\delta_{cma}(K_1) \equiv \delta_{cma}(S_1)$. It then follows from Lemma 4.8 that v is a $(\preceq_{lex})$-minimal model of K_1 if and only if $\delta_{cma}(K_1)$.

Consider $n = k$. Assume that v is a $(\preceq_{lex})$-minimal model of K_k if and only if v satisfies $\delta_{cma}(K_k)$. Consider $n = k + 1$.

($\Longrightarrow$) Suppose v is a $(\preceq_{lex})$-minimal model of K_{k+1}. It implies that v is a $(\preceq_{lex})$-minimal model of K_k and a $(\preceq_{S_{k+1}})$-minimal model. From our assumption, v is a $(\preceq_{lex})$-minimal model of K_k, so we know that v satisfies $\delta_{cma}(K_k)$. Since v is a $(\preceq_{S_{k+1}})$-minimal model, by Proposition 4.8 we know that v satisfies $\delta_{cma}(S_{k+1})$. It then follows that v satisfies $\delta_{cma}(K_k) \cup \delta_{cma}(S_{k+1})$ which means that v satisfies $\delta_{cma}(K_{k+1})$.

($\Longleftarrow$) This direction can be constructed by reversing all steps of ($\Longrightarrow$).

□

Corollary 4.10 *Let K be a stratified knowledge base and $\delta_{cma}(K)$ the knowledge base obtained from K by the CMA algorithm (shown in Algorithm 2). Then $K \models_{lex} \phi$ if and only if $\delta_{cma}(K) \models \phi$.*

Proof. This result follows immediately from Lemma 4.9. Since $K \models_{lex} \phi$, so every $(\preceq_{lex})$-minimal model of K satisfies ϕ. By Lemma 4.9, every model of $\delta_{cma}(K)$ is also a $(\preceq_{lex})$-minimal model of K, so every model of $\delta_{cma}(K)$ satisfies ϕ. Therefore, $\delta_{cma}(K) \models \phi$. Similarly, it can be shown that the other direction also holds. □

4.5 Knowledge Integration in Description Logics

In this section, we present an adaptation of adjustments into the DL context. We highlight some of the problems in the expressivity of DL that introduces difficulties to perform adjustments, and also ways to overcome these problems. Similar to propositional adjustments, we consider a stratified knowledge base $K = (S_1, \ldots, S_n)$, where each stratum S_i is a finite multiset of DL sentences for $1 \leq i \leq n$.

We present two versions of the CMA strategy in DL, as well as versions of lexicographic entailment corresponding to these CMA strategies. We present a direct translation of the propositional CMA strategy to DL called CMA-DL, where sentences in DL are simply treated just like sentences in propositional adjustments. That is, DL sentences are regarded as the atomic elements to compose weakened sentences. As we shall see in later sections, this would lead to counter-intuitive results. This prompted us to explore an original refined version of the CMA strategy named RCMA-DL. This version of CMA exploits the expressivity of DLs adequately by taking the structure of DL sentences into account. RCMA-DL produces more desirable solutions than CMA-DL.

We outline below a number of technical issues that prohibit direct conversion of propositional adjustments into DL.

Lack of Expressivity

There are differences in the way sentences are expressed in propositional languages and in description languages. Classical propositional languages generally allow logical connectives to be freely used between propositional formulas. It is syntactically valid to form disjunctions (or conjunctions) freely among propositional formulas. This makes it possible to weaken sentences in a desirable manner. In fact, for any propositional formula ϕ and any subset V' of all valuations of ϕ, it is possible to find a formula ϕ' such that V' is exactly the set of all models of ϕ'.

This virtually means that one could construct any weakening of ϕ. Note also that the existence of ϕ' is guaranteed only because the language is finite (as in the case of propositional languages). On the other hand, description languages do not allow disjunctions to form over arbitrary sentences. In particular, disjunctions of TBox and ABox elements are not syntactically valid. For example, the following statements are considered ill-formed:

$$(C \sqsubseteq D) \vee C(s)$$
$$C(s) \vee C(t)$$
$$C(s) \vee R(s,t)$$
$$R(s,t) \vee R(t,u)$$

It should be noted that there are exceptional cases where disjunctions of sentences can be expressed in DLs. Disjunctions over subsumption (or equality) axioms can be expressed by converting each axiom into its corresponding disjunction form and then reassembling them into an axiom. For example, disjunction of $C \sqsubseteq D$ and $E \sqsubseteq F$ can be expressed as $(\neg C \sqcup D) \sqcup (\neg E \sqcup F)$ or as an axiom $\neg(\neg C \sqcup D) \sqsubseteq (\neg E \sqcup F)$. There are two ways to compensate for this shortcoming in expressivity:

1. Make the language more expressive by introducing additional constructs so that the weakened sentence can be expressed adequately.

2. Keep the same level of expressivity and provide an expressible sentence that is

semantically closest to the desired inexpressible sentence.

We adopt the first approach. In order to express disjunction over sentences, we introduce the notion of a disjunctive DL knowledge base, or DKB, as a set of DL knowledge bases. The semantics of DKBs is defined as follows.

Definition 4.11
A DKB $\mathcal{B}$ is satisfied by an interpretation $\mathcal{I}$ (or $\mathcal{I}$ is a model of $\mathcal{B}$) if and only if $\mathcal{I}$ is a model of at least one element in $\mathcal{B}$. $\mathcal{B}$ entails a DKB Φ, written $\mathcal{B} \models \Phi$, if and only if every model of $\mathcal{B}$ is a model of Φ.

Informally $\mathcal{B}$ is read as the disjunction of its elements, where each element of $\mathcal{B}$ is a conjunction of the sentences contained in it.

Here disjunction is treated as classical disjunction in propositional logic. That is, one of the elements in the disjunction must hold.

For example, $\{[C \sqsubseteq D, C(a)], [C \sqsubseteq D, D(a)]\}$ states that one of the following holds: (1) $[C \sqsubseteq D,\ C(a)]$; (2) $[C \sqsubseteq D,\ D(a)]$. There are more fundamental reasons for the use of DKBs as well, which will briefly be touched on in the later sections.

Comparability of DL Interpretations

The semantics of propositional lexicographic entailment relies on the fact that propositional interpretations are comparable to each other with respect to some preference metrics. This is easily achieved in the propositional context because a propositional interpretation is merely a function $f : A \rightarrow B$ where A is the set of propositional variables and B is the boolean domain $\{0, 1\}$. In other words, each interpretation maps every propositional variable to a truth value. Therefore, propositional interpretations are naturally comparable because they share the same set of propositional variables, and identical variables denote the same proposition. However, this is not the case in DL interpretations. Recall that, a DL interpretation is defined as $\mathcal{I} = (\Delta^{\mathcal{I}}, .^{\mathcal{I}})$ where $\Delta^{\mathcal{I}}$ is the domain of discourse and $.^{\mathcal{I}}$ is the interpretation function. Since each interpretation acquires its own domain and interpretation function, it is no longer possible to maintain comparability of DL interpretations in the semantics. In particular, whenever two

interpretations have different domains or do not map the same individual names to the same elements in the domain, it is counter-intuitive to insist that they are comparable in terms of preference. This issue is solved by requiring that only interpretations obtained from the same pre-interpretation be comparable. A pre-interpretation is an ordered pair $\pi = (\Delta^\pi, d^\pi)$, where Δ^π is a domain and d^π is a denotation function. Let Π be the class of all pre-interpretations. For every pre-interpretation $\pi = (\Delta^\pi, d^\pi)$, let $\mathcal{I}^\pi$ be the class of interpretations $\mathcal{I}$ with $\Delta^\mathcal{I} = \Delta^\pi$ and $d^\mathcal{I} = d^\pi$.

4.5.1 Conjunctive Maxi-Adjustment (CMA) for DL

We provide a semantics similar to that of propositional lexicographic entailment, but each ordering $\preceq_{S_i}$ on valuations associated with a stratum S_i will, in the case of DLs, be replaced by a class of ordering $\preceq^\pi_{S_i}$: one for each pre-interpretation π in Π. For a fixed π, the orderings $\preceq^\pi_{S_i}$ for $i \in \{1, \ldots, n\}$ are then lexicographically combined using Definition 4.3 to obtain the ordering $\preceq^\pi_{lex}$. Lexicographical entailment is then defined in terms of the minimal models of all these ordering. That is, given a preference ordering $\preceq^\pi_{lex}$ for each $\pi \in \Pi$, lexicographic entailment ($\models_{lex}$) for stratified DL knowledge bases is defined as follows:

$$K \models_{lex} \Phi \text{ if and only if } \bigcup_{\pi \in \Pi} min_{\preceq^\pi_{lex}} \subseteq M(\Phi) \tag{4.1}$$

where $min_{\preceq^\pi_{lex}}$ refers to the ($\preceq^\pi_{lex}$)-minimal models. The one remaining question is how the preference orderings $\preceq^\pi_{S_i}$ used in the construction of $\preceq^\pi_{lex}$ should be obtained. A first attempt is to use the same technique as that used for propositional lexicographic entailment. That is, for each $\mathcal{I} \in \mathcal{I}^\pi$ and each stratum S_i, let the number of S_i-exceptions w.r.t. $\mathcal{I}$ be the number of sentences in S_i falsified by $\mathcal{I}$, and use these exceptions to generate the ordering $\preceq^\pi_{S_i}$.

The CMA algorithm for DL (shown in 3 and named CMA-DL) takes a stratified knowledge base $K = (S_1, \ldots, S_n)$ as argument. It maintains globally a set of Disjunctive Knowledge Bases (DKBs) $\mathcal{B}_0, \ldots, \mathcal{B}_n$, with $\mathcal{B}_0$ initialised as a set containing the empty set

input : $K = (S_1, \ldots, S_n)$
output: A consistent DKB
$\mathcal{B}_0 \leftarrow \{\emptyset\}$;
for $i \leftarrow 1$ **to** n **do**
  $\mathcal{B}_i \leftarrow \emptyset$;
  foreach $B \in \mathcal{B}_{i-1}$ **do**
    $j \leftarrow 0$;
    repeat
      $\mathcal{W} \leftarrow \{B \cup \Phi \mid \Phi \subseteq S_i$ **and** $|\Phi| = |S_i| - j$ **and** $B \cup \Phi \not\models \bot\}$;
      $j \leftarrow j + 1$;
    until $\mathcal{W} \neq \emptyset$;
    $\mathcal{B}_i \leftarrow \mathcal{B}_i \cup \mathcal{W}$;
  end
end
return $\mathcal{B}_n$

Algorithm 3: The CMA Algorithm for Description Logics (CMA-DL).

and $\mathcal{B}_1, \ldots, \mathcal{B}_n$ are initially empty sets. The core part of the algorithm has three nested loops. The most outer loop iterates through each stratum S_i of the stratified knowledge base from $i = 1$ to n. For each stratum S_i, the algorithm uses the DKB $\mathcal{B}_{i-1}$ generated from the previous stratum S_{i-1} and extends each DKB in $\mathcal{B}_{i-1}$ by adding subsets of the elements in S_i such that the resulting DKB is consistent. This is then added to the set of DKB $\mathcal{B}_i$. This algorithm returns $\mathcal{B}_n$, which is the set of DKBs constructed after going through the n^{th} stratum.

Example 4.12

This is a simple example demonstrating the CMA-DL algorithm in (Algorithm 3):

$$S_1 = [C \sqsubseteq \neg D, C \sqsubseteq \neg E, D \sqsubseteq \neg E]$$
$$S_2 = [C(a), D(a), E(a)]$$

Let $K = (S_1, S_2)$, where S_1 and S_2 are shown above. Initially, $\mathcal{B}_0$ is set to $\{\emptyset\}$ and we consider i from 1 to n. When i is 1, $\mathcal{B}_1$ is set to an empty set. Since $\mathcal{B}_0$ has only one element, namely $\emptyset$, so B is $\emptyset$. j is then set to 0 and we consider the most inner repeat loop. Since j is 0, we look at subsets of S_1 with size $|S_1| = 3$ that are consistent with B. In this case, S_1 is the only subset that satisfies the condition, so

we set $\mathcal{W}$ to $\{S_1\}$ and break out of the repeat loop since $\mathcal{W} \neq \emptyset$. So $\mathcal{B}_1$ is then set to $\{[C \sqsubseteq \neg D, C \sqsubseteq \neg E, D \sqsubseteq \neg E]\}$. When i is 2, $\mathcal{B}_2$ is set to an empty set and we consider each DKB in $\mathcal{B}_1$. $\mathcal{B}_1$ has only one element, so we set B to $\{S_1\}$ and set j to 0. Now, we consider the most inner repeat loop. Since j is 0, so we look at subsets of S_2 with size $|S_2| = 3$ that are consistent with B.

However, $S_1 \cup S_2$ is *not* consistent, so $\mathcal{W}$ is set to $\emptyset$ and j is set to 1. Since $\mathcal{W} = \emptyset$, we cannot break of the repeat loop. Instead, we consider subsets of S_2 with size $|S_2| - j = 2$, and they are $\{[C(a), D(a)], [C(a), E(a)], [D(a), E(a)]\}$. However, none of these three are consistent with S_1, so we increment j to 2 and continue with the repeat loop. This time we consider subsets of S_2 with size $|S_2| - 2 = 1$ and we have $\{[C(a)], [D(a)], [E(a)]\}$.

Since each of these are consistent with S_1 so $\mathcal{W}$ is non-empty and we break out of the repeat loop. Finally, $\mathcal{B}_2$ is set to the following:

$$\begin{aligned}\mathcal{B}_2 = \{&[C \sqsubseteq \neg D, C \sqsubseteq \neg E, D \sqsubseteq \neg E, C(a)],\\ &[C \sqsubseteq \neg D, C \sqsubseteq \neg E, D \sqsubseteq \neg E, D(a)],\\ &[C \sqsubseteq \neg D, C \sqsubseteq \neg E, D \sqsubseteq \neg E, E(a)]\}\end{aligned}$$

The algorithm breaks out of all loops and returns $\mathcal{B}_2$.

As with classical CMA, CMA-DL (and also RCMA-DL) are methods that compute all possible alternatives instead of a single one. There is no established way to select between these alternatives, however, this selection is likely to be domain specific. One possible solution is to select alternatives based on further refinement of the ordering between the input sentences. For example, in the above example, we can select between the three alternatives by refining the ordering between the three concept assertions (e.g., $S_1 = [C \sqsubseteq \neg D, C \sqsubseteq \neg E, D \sqsubseteq \neg E]$, $S_2 = [C(a)]$, $S_3 = [D(a)]$ and $S_4 = [E(a)]$.

Definition 4.13 (Ordering of DL Interpretations)
Let $\pi \in \Pi$, $\mathcal{I} \in \mathcal{I}^\pi$, ϕ be a DL statement, and Φ be a multiset of DL statements. The number of ϕ-exceptions $e^\phi(\mathcal{I})$ for $\mathcal{I}$ is 0 if $\mathcal{I}$ satisfies ϕ and 1 otherwise. The number of

Φ-exceptions for $\mathcal{I}$ is: $e^{\Phi}(\mathcal{I}) = \sum_{\phi \in \Phi} e^{\phi}(\mathcal{I})$. The ordering $\preceq^{\phi}_{\Phi}$ on $\mathcal{I}^{\pi}$ is defined as:

$$\mathcal{I} \preceq^{\pi}_{\Phi} \mathcal{J} \text{ if and only if } e^{\Phi}(\mathcal{I}) \leq e^{\Phi}(\mathcal{J})$$

To show the following lemma, we introduce two notations $\delta^{i}_{cma-dl}(\Phi)$ and $\delta_{cma-dl}(\Phi)$. The former corresponds to the i-weakening of the set of sentences Φ. Formally, $\delta^{i}_{cma-dl}(\Phi) = \{\Phi' | \Phi' \subseteq \Phi, |\Phi'| = i, \Phi' \text{ is consistent}\}$. The latter corresponds to the i-weakening of Φ such that it is non-empty for some value i, but the j-weakening of Φ is empty for all $j < i$.

Lemma 4.14

Let Φ be a multiset of DL sentences $[\phi_1, \ldots, \phi_n]$, where $|\Phi| = n$. A DL interpretation $\mathcal{I}$ is a $(\preceq^{\pi}_{\Phi})$-minimal model if and only if $\mathcal{I}$ satisfies $\delta_{cma-dl}(\Phi)$.

Proof. Similar to Proposition 4.8. We establish a connection between the $(\preceq_{\Phi})$-minimal models and the weakened DL sentences.

1. ($\Longrightarrow$) Suppose $\mathcal{I}$ is a $(\preceq^{\pi}_{\Phi})$-minimal model. We need to show that $\mathcal{I}$ satisfies $\delta_{cma-dl}(\Phi)$. This amounts to showing that there is some i, such that: (1) $\delta^{j}_{cma-dl}(\Phi)$ is inconsistent for all $j < i$. That is, there does not exist an interpretation $\mathcal{J}$ such that $\mathcal{J}$ satisfies $\delta^{j}_{cma-dl}(\Phi)$ for any $j < i$, and (2) $\mathcal{I}$ satisfies $\delta^{i}_{cma-dl}(\Phi)$.

   Consider case(1). To show (1), suppose there is an interpretation $\mathcal{J}$ that satisfies $\delta^{j}_{cma-dl}(\Phi)$ for some $j < e^{\Phi}(\mathcal{I})$. By definition, $\delta^{j}_{cma-dl}(\Phi)$ is a DKB that contains all sets of size $|\Phi| - j$ of the elements in Φ. Since $\mathcal{J}$ satisfies $\delta^{j}_{cma-dl}(\Phi)$, by definition 4.13 it follows that $\mathcal{J}$ satisfies a subset Φ' of Φ where $|\Phi'| = |\Phi| - j$. By Def. 4.13, we know that $\mathcal{J}$ satisfies at least $|\Phi'|$ DL sentences of Φ, so $e^{\Phi}(\mathcal{J}) \leq j$. Since $j < e^{\Phi}(\mathcal{I})$, so $e^{\Phi}(\mathcal{J}) < i$. This implies that $\mathcal{I}$ is not a $\preceq^{\pi}_{\Phi}$-minimal model, so we have a contradiction. Therefore, (1) holds.

   Consider case (2). By Def. 4.13, we know that $e^{\Phi}(\mathcal{I})$ is the number of DL sentences in Φ that $\mathcal{I}$ does not satisfy, where $0 \leq e^{\Phi}(\mathcal{I}) \leq |\Phi|$. This means that there is some subset Φ' of Φ, such that $|\Phi'| = |\Phi| - e^{\Phi}(\mathcal{I})$, and $\mathcal{I}$ satisfies ϕ for all $\phi \in \Phi'$. That

is, $\mathcal{I}$ satisfies Φ'. By definition, we know that $\delta^{e^{\Phi}(\mathcal{I})}_{cma-dl}(\Phi)$ is a DKB that contains *all* sets of size $|\Phi'|$ of the elements in Φ and Φ' is one of these sets. Since $\mathcal{I}$ satisfies Φ', it follows that $\mathcal{I}$ also satisfies $\delta^{e^{\Phi}(\mathcal{I})}_{cma-dl}(\Phi)$. Therefore (2) holds.

2. ($\Longleftarrow$) Suppose $\mathcal{I}$ satisfies $\delta_{cma-dl}(\Phi)$, and $\mathcal{I}$ is *not* a $(\preceq^{\pi}_{\Phi})$-minimal model. We know that $\delta_{cma-dl}(\Phi) = \delta^{i}_{cma-dl}(\Phi)$ for some i, and $\delta^{j}_{cma-dl}(\Phi)$ is inconsistent for all $j < i$. Since $\mathcal{I}$ satisfies $\delta_{cma-dl}(\Phi)$, it also satisfies $\delta^{i}_{cma-dl}(\Phi)$ for some i. By definition, $\delta^{i}_{cma-dl}(\Phi)$ is a DKB that contains all sets of size $|\Phi| - i$ of the elements of Φ, so $\mathcal{I}$ satisfies a subset Φ' of Φ where $|\Phi'| = |\Phi| - i$. It implies that $\mathcal{I}$ satisfies at least $|\Phi| - i$ DL sentences of Φ, so by definition it means that $e^{\Phi}(\mathcal{I}) \leq i$. However, $\mathcal{I}$ does not satisfy $\delta^{j}_{cma-dl}(\Phi)$ for all $j < i$, so it follows that $e^{\Phi}(\mathcal{I})$ is *not* less than i. Therefore $e^{\Phi}(\mathcal{I}) = i$. If $\mathcal{I}$ is *not* a $(\preceq^{\pi}_{\Phi})$-minimal model then there exists an interpretation $\mathcal{J}$ such that $e^{\Phi}(\mathcal{J}) < e^{\Phi}(\mathcal{I})$. By Def. 4.13, $\mathcal{J}$ does not satisfy

$e^{\Phi}(\mathcal{J})$ DL sentences of Φ, so $\mathcal{J}$ satisfies $|\Phi| - e^{\Phi}(\mathcal{J})$ DL sentences of Φ. Therefore, there exists a subset Φ' of Φ where $|\Phi'| = |\Phi| - e^{\Phi}(\mathcal{J})$, and $\mathcal{J}$ satisfies ϕ for all $\phi \in \Phi'$. That is, $\mathcal{J}$ satisfies Φ'. By definition, we know that $\delta^{e^{\Phi}(\mathcal{J})}_{cma}(\Phi)$ is a DKB that contains all sets of size $|\Phi| - e^{\Phi}(\mathcal{J})$ of the elements of Φ, and Φ' is an element of this DKB. Since $\mathcal{J}$ satisfies Φ', so by definition it also satisfies $\delta^{e^{\Phi}(\mathcal{J})}_{cma}(\Phi)$. But $e^{\Phi}(\mathcal{J}) < e^{\Phi}(\mathcal{I})$ and $e^{\Phi}(\mathcal{J}) = i$, so $\mathcal{J}$ satisfies $\delta^{e^{\Phi}(\mathcal{J})}_{cma}(\Phi)$, where $e^{\Phi}(\mathcal{J}) < i$. This is a contradiction.

□

Lemma 4.15

Let K be a stratified DL knowledge base. An interpretation $\mathcal{I}$ is a $(\preceq^{\pi}_{lex})$-minimal model of K if and only if $\mathcal{I}$ satisfies $\delta_{cma-dl}(K)$.

Proof. Let $K = K_n$, $K_i = \{S_1, \ldots, S_i\}$ for $i > 0$, and $K_0 = \emptyset$. Also, let $(\preceq^{K_i}_{lex})$ be the lexicographic ordering of K_i, and $\delta_{cma-dl}(K_i)$ be the output of K_i from Algorithm 3.

Consider $n = 1$. By Def. 4.3, we can derive that $(\preceq^{K_1}_{lex}) \equiv (\preceq_{K_1})$. Also, we can derive that $\delta_{cma-dl}(K_1) \equiv \delta_{cma-dl}(S_1)$. It then follows from Lemma 4.14 that $\mathcal{I}$ is a $(\preceq_{lex})$-minimal model of K_1 if and only if $\delta_{cma-dl}(K_1)$.

Consider $n = k$. Assume that $\mathcal{I}$ is a $(\preceq_{lex}^{K_k})$-minimal model if and only if $\mathcal{I}$ satisfies $\delta_{cma-dl}(K_k)$. Consider $n = k + 1$. Suppose $\mathcal{I}$ is a $(\preceq_{lex}^{K_{k+1}})$-minimal model. It implies that $\mathcal{I}$ is a $(\preceq_{lex}^{K_k})$-minimal model and a $(\preceq_{S_{k+1}})$-minimal model. From our assumption, we know that $\mathcal{I}$ satisfies $\delta_{cma-dl}(K_k)$, since $\mathcal{I}$ is a $(\preceq_{lex}^{K_k})$-minimal model. Also, since $\mathcal{I}$ is a $(\preceq_{S_{k+1}})$-minimal model, by Proposition 4.14 we know that $\mathcal{I}$ satisfies $\delta_{cma-dl}(S_{k+1})$. It then follows that $\mathcal{I}$ satisfies $\delta_{cma-dl}(K_k) \cup \delta_{cma-dl}(S_{k+1})$ which means that $\mathcal{I}$ satisfies $\delta_{cma-dl}(K_{k+1})$. □

Deficiency of CMA

Lexicographic entailment for DLs and the CMA-DL strategy are both faithful translations of their propositional counterparts. It is precisely because of this that they do not take the structure of DL statements into account. The following example illustrates this deficiency.

Example 4.16

This example highlights the deficiency of the propositional CMA algorithm applied to DL (i.e., the CMA-DL algorithm).

$$S_1 = [Bird(tweety), \neg Flies(tweety), Bird(chirpy)]$$
$$S_2 = [Bird \sqsubseteq Flies]$$

Let $K = (S_1, S_2)$, where S_1 and S_2 are shown above. Again, we set $\mathcal{B}_0$ to $\{\emptyset\}$ and consider i from 1 to n. When i is 1, we set $\mathcal{B}_1$ to $\emptyset$ and we look at each element of $\mathcal{B}_{i-1} = \mathcal{B}_0$. Since $\emptyset$ is the only element in $\mathcal{B}_0$, we set B to $\emptyset$. Also, we set j to 0. Next, we consider the most inner repeat loop. The subsets Ψ of $S_i = S_1$ with with size $|\Psi| = |S_i - j| = 3$ are $\{S_1\}$. Since S_1 is consistent with $B = \emptyset$, so we set $\mathcal{W}$ to $\{[Bird(tweety), \neg Flies(tweety), Bird(chirpy)]\}$. We break out of the repeat loop since $\mathcal{W} \neq \emptyset$ and set $\mathcal{B}_i = \mathcal{B}_1$ to $\mathcal{W}$. When i is 2, we set $\mathcal{B}_2$ to $\emptyset$ and we look at each element of $\mathcal{B}_{i-1} = \mathcal{B}_1$. There is only one element in $\mathcal{B}_1$, so we set B to $[Bird(tweety), \neg Flies(tweety), Bird(chirpy)]$. Again, we consider the most inner repeat loop. At the first iteration, j is set to 0, so we consider subsets Ψ of $S_i = S_2$ where

$|\Psi| = |S_2| - j = 1$.

There is only one such subset which is $[Bird \sqsubseteq Flies]$, but it is inconsistent with B so $\mathcal{W} = \emptyset$. We cannot break out of the repeat loop because $\mathcal{W} = \emptyset$. For the second iteration, j is set 1 and we consider subsets Ψ of $S_i = S_2$ where $|\Psi| = |S_2| - j = 0$. Obviously, this is the $\emptyset$ and it is consistent with B. Therefore, we set $\mathcal{W}$ to B and break out of the repeat loop. We then set $\mathcal{B}_i = \mathcal{B}_2 = \{[Bird(tweety), \neg Flies(tweety), Bird(chirpy)]\}$. Finally, we break out of all loops and return $\mathcal{B}_2$.

When Algorithm 3 (CMA-DL) is applied to K from Example 4.16, it concludes correctly that Tweety is a non-flying bird and that Chirpy is a bird. But it does not conclude that Chirpy flies since it has discarded the statement $bird \sqsubseteq flies$. It therefore does not exploit the structure of $bird \sqsubseteq flies$ appropriately. Ideally, we should be able to conclude that Tweety is an exception and that all birds other than Tweety (including Chirpy) can fly.

For this to be possible we need to weaken TBox statements such as $bird \sqsubseteq flies$, something that is not possible in the propositional approach (including the CMA approach) where we treat a TBox sentence as a single propositional sentence (e.g., $bird \sqsubseteq flies$ is treated as p).

Corollary 4.17

Let K be a stratified DL knowledge base, $\mathcal{B}$ the DKB obtained from K by CMA-DL in Algorithm 3, let lexicographic entailment for DLs be defined in terms of Definition 4.13, and let Φ be a DKB. Then $K \models_{lex} \Phi$ if and only if $K' \models_{lex} \Phi$.

Proof. This result follows immediately from Proposition 4.15. □

4.5.2 Refined Conjunctive Maxi-adjustment

An element in the domain of an interpretation $\mathcal{I}$ violates a statement of the form $C \sqsubseteq D$ if it is $C^{\mathcal{I}}$ but not in $D^{\mathcal{I}}$, i.e. if it is in $C^{\mathcal{I}} \cap (\neg D)^{\mathcal{I}}$.

Definition 4.18

Let $\pi \in \Phi$, $\mathcal{I} \in \mathcal{I}^{\pi}$, ϕ a DL statement, and Φ a multiset of DL statements. If ϕ is an ABox statement, the number of ϕ-exceptions $e^{\phi}(\mathcal{I})$ for an interpretation $\mathcal{I}$ is 0 if $\mathcal{I}$

satisfies ϕ and 1 otherwise. If ϕ is a TBox statement of the form $C \sqsubseteq D$, the number of ϕ-exceptions for $\mathcal{I}$ is:

$$e^{\phi}(\mathcal{I}) = \begin{cases} |C^{\mathcal{I}} \cap \neg D^{\mathcal{I}}| & \text{if } C^{\mathcal{I}} \cap \neg D^{\mathcal{I}} \text{ is finite;} \\ \infty & \text{otherwise.} \end{cases}$$

The number of Φ-exceptions for $\mathcal{I}$ is $e^{\Phi}(\mathcal{I}) = \sum_{\phi \in \Phi} e^{\phi}(\mathcal{I})$. The ordering $\preceq_{\Phi}^{\pi}$ on $\mathcal{I}^{\pi}$ is: $\mathcal{I} \preceq_{\Phi}^{\pi,*} \mathcal{J}$ if and only if $e^{\Phi}(\mathcal{I}) \leq e^{\Phi}(\mathcal{J})$.

So $\preceq_{X}^{\pi}$ is a version of cardinality-based circumscription [LS95]: the more exceptions, the less preferred an interpretation, while interpretations with an infinite number of exceptions are all equally bad.

Using Definition 4.3 in our construction of lexicographic entailment will ensure that we will be able to conclude, in Example 4.16, that Chirpy can fly. However, we are still unable to express the conclusion that all birds, except for Tweety, can fly. The problem is that the notion of an exception is not expressible in a DL. We cannot state that all birds, with the exception of one, can fly. It is necessary to extend the level of expressivity of the DL languages we are interested in.

An appropriate extension, adding cardinality restrictions on concepts, was proposed in [BBH96]. There, its introduction was motivated by the use of DL systems for solving configuration tasks. These restrictions are placed on statements in the TBox, allowing one to express restrictions on the number of elements a concept may have: $(\geq mC)$ and $(\leq nC)$ respectively express that the concept C has at least m elements and at most n elements. For our purposes it is sufficient to consider cardinality restrictions of the form $(\leq nC)$. An alternative approach is to make use of nominals, which is a common construct in more recently developed description logics such as $\mathcal{SHOIQ}$, $\mathcal{SROIQ}$ and $\mathcal{EL}^{++}$. In contrast to cardinality restrictions on concepts, expressing exceptions with nominals require explicitly stating the individuals that violates the constraints. For example, $C \sqcap \neg\{a\} \sqsubseteq D$ expresses that the individual a is an exception to the TBox axiom $C \sqsubseteq D$. It should be noted that nominal does incur additional costs in terms of reasoning. However, it is known that satisfiability checking in both $\mathcal{SHOIQ}$ and $\mathcal{SROIQ}$ are decidable but not tractable, and subsumption checking in $\mathcal{EL}^{++}$ is tractable.

In the following section, we will elaborate on the semantics of cardinality restrictions on concepts.

An interpretation $\mathcal{I}$ is said to satisfy a restriction of the form $(\leq nC)$ if and only if $|C^{\mathcal{I}}| \leq n$. The statement $C \sqsubseteq D$ is equivalent to stating that the concept $C \sqcap \neg D$ is empty, i.e. that $(\leq 0(C \sqcap \neg D))$. This demonstrates that the TBox statements we have considered thus far can all be expressed as cardinality restrictions. Therefore, a TBox will from now on be a finite multiset of cardinality restrictions. An interpretation $\mathcal{I}$ is a model of such a TBox if and only if it satisfies each of its restrictions. Other semantic notions such as entailment are extended in the obvious way. With the inclusion of cardinality restrictions we can now rephrase S_2 in Example 4.16 as $\{(\leq 0(bird \sqcap \neg flies))\}$. Additionally, using Definition 4.18, K now lexicographically entails that Tweety is a non-flying bird, that Chirpy is a flying bird, and that there is at most one non-flying bird, $(\leq 1(bird \sqcap \neg flies))$, which is a weakening of S_2. So it follows that, barring Tweety, all birds can fly.

The next step is to refine the CMA-DL strategy to coincide with the modified version of lexicographic entailment for DLs. This strategy, referred to as refined CMA-DL (or RCMA-DL for short), is described in Algorithm 4. The main difference between the two algorithms is in the construction of $\delta_{rcma-dl}(\Phi)$. ABox sentences are treated exactly as in Algorithm 3.

The i-weakening $\delta^i_{rcma-dl}(\Phi_{\mathcal{A}})$ of the ABox $\Phi_{\mathcal{A}}$ of a set of DL sentences Φ (where $i \leq |\mathcal{A}|$), contains all those sub-multisets of $\Phi_{\mathcal{A}}$ where the size of $\Phi_{\mathcal{A}}$ is $|\Phi_{\mathcal{A}}| - i$. Equivalently, $\delta^i_{rcma-dl}(\Phi_{\mathcal{A}}) = \{\Phi'_{\mathcal{A}} \mid \Phi'_{\mathcal{A}} \subseteq \Phi_{\mathcal{A}}, |\Phi'_{\mathcal{A}}| = |\Phi_{\mathcal{A}}| - i\}$. For example, for $\Phi_{\mathcal{A}} = [C(a), D(a), E(a)]$ we have the following:

$$\delta^0_{rcma-dl}(\Phi_{\mathcal{A}}) = \{[C(a), D(a), E(a)]\}$$
$$\delta^1_{rcma-dl}(\Phi_{\mathcal{A}}) = \{[C(a), D(a)], [C(a), E(a)], [D(a), E(a)]\}$$
$$\delta^2_{rcma-dl}(\Phi_{\mathcal{A}}) = \{[C(a)], [D(a)], [E(a)]\}$$
$$\delta^3_{rcma-dl}(\Phi_{\mathcal{A}}) = \top$$

The i-weakening $\delta^i_{rcma-dl}(\Phi_{\mathcal{T}})$ of the TBox $\Phi_{\mathcal{T}}$ of a set of DL sentences Φ (where

$i \leq |\mathcal{T}|$), contains all those sets of TBox sentences that have been weakened $|\Phi_\mathcal{T}| - i$ times.

More specifically, for a TBox sentence of the form $(\leq nC)$ (where C can be a complicated concept[1]), $(\leq (n+m)C)$ is the weakened version of $(\leq nC)$ where m indicates that it has been weakened m times. In other words, m is the extent in which the original TBox sentence is being weakened. The i-weakening $\delta^i_{rcma-dl}(\Phi_\mathcal{T})$ of the TBox $\Phi_\mathcal{T}$ is defined as:

$$\begin{aligned}\delta^i_{rcma-dl}(\Phi_\mathcal{T}) = \{\Phi'_\mathcal{T} \mid\ & \Phi_\mathcal{T} = [(\leq n_1 C_1), \ldots, (\leq n_n C_n)], \\ & \Phi'_\mathcal{T} = [(\leq (n_1+m_1)C_1), \ldots, (\leq (n_n+m_n)C_n)], \\ & \sum_{j=1}^{n} m_j = i\}\end{aligned}$$

For example, for $\Phi_\mathcal{T} = [(\leq 0C), (\leq 0D)]$, we have:

$$\begin{aligned}\delta^0_{rcma-dl}(\Phi_\mathcal{T}) &= \{[(\leq 0C), (\leq 0D)]\} \\ \delta^1_{rcma-dl}(\Phi_\mathcal{T}) &= \{[(\leq 0C), (\leq 1D)], [(\leq 1C), (\leq 0D)]\} \\ \delta^2_{rcma-dl}(\Phi_\mathcal{T}) &= \{[(\leq 0C), (\leq 2D)], [(\leq 1C), (\leq 1D)], [(\leq 2C), (\leq 0D)]\} \\ &\ldots\end{aligned}$$

The i-weakening $\delta^i_{rcma-dl}(\Phi)$ of a set of DL sentences Φ is defined as follows:

$$\begin{aligned}\delta^i_{rcma-dl}(\Phi) = \{\Phi'_\mathcal{A} \cup \Phi'_\mathcal{T} \mid\ & \Phi'_\mathcal{A} \in \delta^j_{rcma-dl}(\Phi_\mathcal{A}), \\ & \Phi'_\mathcal{T} \in \delta^k_{rcma-dl}(\Phi_\mathcal{T}), \\ & j < |\Phi_\mathcal{A}|, i = j + k, \\ & \Phi = \Phi_\mathcal{A} \cup \Phi_\mathcal{T}\}\end{aligned}$$

[1] A complicated concept or complex concept are concepts that are not primitive concepts (or concept names). For example, conjunction, disjunction, existential or universal restrictions.

For example, for $\Phi = [(\leq 0C), C(a), C(b)]$, then

$$\delta^2(\Phi) = \{[(\leq 1C), C(a)], [(\leq 1C), C(b)], \\ [(\leq 0C)], [(\leq 2C), C(a), C(b)]\}$$

So $\delta^2(\Phi)$ contains those weakenings of Φ in which exactly two exceptions occur. The j-weakenings of a set of DL sentences are used in the repeat loop of Algorithm 4 where $\mathcal{W}$ is set to the j-weakening of S_i. As required, RCMA-DL is a compilation of lexicographic entailments using Definition 4.18.

input : $K = (S_1, \ldots, S_n)$
output: A consistent DKB
$\mathcal{B}_0 \leftarrow \{\emptyset\}$;
for $i \leftarrow 1$ **to** n **do**
  foreach $B \in \mathcal{B}_{i-1}$ **do**
    $j \leftarrow 0$;
    repeat
      $\mathcal{W} \leftarrow \{B \cup \Phi \mid \Phi \in \mathcal{W}^{|S_i|-j}(S_i)$ **and** $B \cup \Phi \not\models \bot\}$;
      $j \leftarrow j + 1$;
    until $\mathcal{W} \neq \emptyset$;
    $\mathcal{B}_i \leftarrow \mathcal{B}_i \cup \mathcal{W}$;
  end
end
return $\mathcal{B}_n$

Algorithm 4: The Refined CMA Algorithm for Description Logics (RCMA-DL).

The RCMA-DL algorithm in Algorithm 4 works similarly to the CMA-DL algorithm in Algorithm 3. The major difference is how sentences are weakened in the most inner repeat loop of the algorithm. This results in much more refined DKBs.

Example 4.19

This is an example demonstrating the RCMA-DL algorithm in Algorithm 4:

$$S_1 = [Bird(tweety), Bird(chirpy)]$$
$$S_2 = [\neg Fly(tweety), \neg Fly(chirpy), (\leq 0 \ \ (Bird \sqcap \neg Fly))]$$

Let $K = (S_1, S_2)$ where S_1 and S_2 are shown above. Initially, $\mathcal{B}_0$ is set to $\emptyset$. We

consider the most outer loop with i set from 1 to n. When i is 1, we look at each element of $\mathcal{B}_{i-1} = \mathcal{B}_0$. There is only element in $\mathcal{B}_0$ and that is the empty set, so we set B to $\emptyset$. We also set j to 0. Next, we consider the most inner repeat loop. At the first iteration, j is set to 0 and $\mathcal{W}^j(S_i) = \mathcal{W}^0(S_1) = \{S_1\}$. Since all elements of $S_1 = [Bird(tweety), Bird(chirpy)]$ are consistent with $B = \emptyset$, so we have $\mathcal{W} = \{S_1\}$. $\mathcal{W} \neq \emptyset$ so we break out of the repeat loop and set $\mathcal{B}_1$ to S_1. When i is 2, we look at each element of $\mathcal{B}_{i-1} = \mathcal{B}_1$. $\mathcal{B}_1$ contains only one element, so we only need to consider the case where B is S_1. Again, we set j to 0 and consider the most inner repeat loop. At the first iteration, j is 0 and $\mathcal{W}^j(S_i) = \mathcal{W}^0(S_2) = \{S_2\}$. However, S_2 is *not* consistent with $B = S_1$, so $\mathcal{W}$ is $\emptyset$ and we cannot break out of the repeat loop. At the second iteration, j is 1 and $\mathcal{W}^j(S_i) = \mathcal{W}^1(S_2) = \{[\neg Fly(tweety), \neg Fly(chipy), (\leq 1(Bird \sqcap \neg Fly))]$, $[\neg Fly(tweety), (\leq 0(Bird \sqcap \neg Fly))], [\neg Fly(chirpy), \leq 0(Bird \sqcap \neg Fly))]\}$. Again, none of the elements in $\mathcal{W}^1(S_2)$ are consistent with B, so $\mathcal{W}$ is $\emptyset$ and we cannot break out of the repeat loop. At the third iteration, j is 2 and $\mathcal{W}^j(S_i) = \mathcal{W}^2(S_2) = \{[\neg Fly(tweety)$, $\neg Fly(chirpy), (\leq 2(Bird \sqcap \neg Fly))], [\neg Fly(tweety), (\leq 1(Bird \sqcap \neg Fly))], [\neg Fly(chirpy), (\leq 1(Bird \sqcap \neg Fly))], [(\leq 0(Bird \sqcap \neg Fly))]\}$. Now, all the elements in $\mathcal{W}^2(S_2)$ are consistent with B so $\mathcal{W}$ contains $B \cup \Psi$ for every Ψ in $\mathcal{W}^2(S_2)$. Finally, the algorithm breaks out of all loops and returns $\mathcal{B}_2$. The elements of $\mathcal{B}_2$ indicates that exactly one of the following four cases hold: a) Tweety and Chirpy are the only two non-flying birds; b) Tweety is the only non-flying bird; c) Chirpy is the only non-flying bird; d) All birds fly, including Tweety and Chirpy.

Proposition 4.20 *Let $\Phi = \Phi_{\mathcal{A}} \cup \Phi_{\mathcal{T}}$ be a set of DL sentences, where $\Phi_{\mathcal{A}}$ is a set of ABox sentences and $\Phi_{\mathcal{T}}$ is a set of TBox sentences. Also, let the number of Φ-exception of an interpretation $\mathcal{I}$ be $e^{\Phi}(\mathcal{I})$. An interpretation $\mathcal{I}$ satisfies $\delta^{e^{\Phi}(\mathcal{I})}_{rcma-dl}(\Phi)$ and does not satisfy $\delta^{j}_{rcma-dl}(\Phi)$ for all $j < e^{\Phi}(\mathcal{I})$.*

Proof. By Def. 4.18, $e^{\Phi}(\mathcal{I})$ is defined as the sum of $e^{\Phi_{\mathcal{A}}}(\mathcal{I})$ and $e^{\Phi_{\mathcal{T}}}(\mathcal{I})$ where $e^{\Phi_{\mathcal{A}}}(\mathcal{I})$ is the number of $\Phi_{\mathcal{A}}$-exceptions of $\mathcal{I}$ and $e^{\Phi_{\mathcal{T}}}(\mathcal{I})$ is the number of $\Phi_{\mathcal{T}}$-exceptions of $\mathcal{I}$. We know that the number of $\Phi_{\mathcal{A}}$-exceptions of $\mathcal{I}$ is $e^{\Phi_{\mathcal{A}}}(\mathcal{I})$, it follows that $\mathcal{I}$ satisfies a subset $\Phi'_{\mathcal{A}}$ of $\Phi_{\mathcal{A}}$ which is of size $|\Phi_{\mathcal{A}}| - e^{\Phi_{\mathcal{A}}}(\mathcal{I})$. Hence, $\mathcal{I}$ satisfies the DKB $\delta^{e^{\Phi_{\mathcal{A}}}(\mathcal{I})}_{rcma-dl}(\Phi_{\mathcal{A}})$.

Similarly, we know that the number of $\Phi_{\mathcal{T}}$-exceptions of $\mathcal{I}$ is $e^{\Phi_{\mathcal{T}}}(\mathcal{I}) = \sum_{\phi \in \Phi_{\mathcal{T}}} e^{\phi}(\mathcal{I}) = \sum_{i=1}^{n} |C_i^{\mathcal{I}} \sqcap \neg D_i^{\mathcal{I}}|$, where $\Phi_{\mathcal{T}} = \{C_1 \sqsubseteq D_1, \ldots, C_n \sqsubseteq D_n\}$. It follows that $\mathcal{I}$ satisfies $(\leq |C_i^{\mathcal{I}} \sqcap \neg D_i^{\mathcal{I}}|(C_i \sqcap \neg D_i))$ for all $1 \leq i \leq n$. Hence, $\mathcal{I}$ satisfies $\delta^{e^{\Phi_{\mathcal{T}}}(\mathcal{I})}(\Phi_{\mathcal{T}})$. Therefore $\mathcal{I}$ satisfies $\delta^{e^{\Phi_{\mathcal{A}}}(\mathcal{I}) + e^{\Phi_{\mathcal{T}}}(\mathcal{I})}(\Phi) = \delta^{e^{\Phi}(\mathcal{I})}(\Phi)$.

□

Lemma 4.21

Let Φ be a multiset of DL sentences $[\phi_1, \ldots, \phi_n]$, where $|\Phi| = n$. A DL interpretation $\mathcal{I}$ is a $(\preceq_{\Phi}^{\pi,*})$-minimal model if and only if $\mathcal{I}$ satisfies $\delta_{rcma-dl}(\Phi)$.

Proof. Similar to Proposition 4.8. We establish a connection between the $(\preceq_{\Phi})$-minimal models and the weakened DL sentences.
$(\Longrightarrow)$
Suppose $\mathcal{I}$ is a $(\preceq_{\Phi}^{\pi,*})$-minimal model. We need to show that $\mathcal{I}$ satisfies $\delta_{rcma-dl}(\Phi)$. This amounts to showing that there is some i, such that: (1) $\delta_{cma-dl}^{j}(\Phi)$ is inconsistent for all $j < i$. That is, there does not exist an interpretation $\mathcal{J}$ such that $\mathcal{J}$ satisfies $\delta_{cma-dl}^{j}(\Phi)$ for any $j < i$, and (2) $\mathcal{I}$ satisfies $\delta_{cma-dl}^{i}(\Phi)$.

Consider case(1). To show (1), suppose there is an interpretation $\mathcal{J}$ that satisfies $\delta_{rcma-dl}^{j}(\Phi)$ for some $j < e^{\Phi_{\mathcal{A}}}(\mathcal{I}) + e^{\Phi_{\mathcal{T}}}(\mathcal{I})$. By definition, $\delta_{rcma-dl}^{j}(\Phi)$ is a DKB that contains all sets of DL sentences that corresponds to the j-weakening of Φ. $\delta_{rcma-dl}^{j}(\Phi)$ can be decomposed into $\delta_{rcma-dl}^{j_{\mathcal{A}}}(\Phi)$ and $\delta_{rcma-dl}^{j_{\mathcal{T}}}(\Phi)$ where $j = j_{\mathcal{A}} + j_{\mathcal{T}}$. Since $\mathcal{J}$ satisfies $\delta_{rcma-dl}^{j_{\mathcal{A}}}(\Phi)$, by definition it satisfies a subset $\Phi'_{\mathcal{A}}$ of $\Phi_{\mathcal{A}}$ where $|\Phi'_{\mathcal{A}}| \geq |\Phi_{\mathcal{A}}| - j_{\mathcal{A}}$. It follows that $e^{\Phi_{\mathcal{A}}}(\mathcal{J}) \leq j_{\mathcal{A}}$. By definition, $\delta_{rcma-dl}^{j_{\mathcal{T}}}(\Phi)$ is a DKB that contains all sets of DL sentences that correspond to the $j_{\mathcal{T}}$-weakening of $\Phi_{\mathcal{T}}$. That is, $\mathcal{J}$ satisfies a set of TBox sentences Φ'_T of the form $(\leq e_1(C_1 \sqcap \neg D_1)), \ldots, (\leq e_n(C_n \sqcap \neg D_n))$ where $\sum_{i=1}^{n} e_i \leq j_{\mathcal{T}}$. We can then derive that $|C_i^{\mathcal{J}} \sqcap \neg D_i^{\mathcal{J}}| \leq e_i$ for all $1 \leq i \leq n$, hence $\sum_{\phi \in \Phi_{\mathcal{T}}} e^{\phi}(\mathcal{J}) \leq \sum_{i=1}^{n} e^{i} \leq j_{\mathcal{T}}$. Therefore, $e^{\Phi_{\mathcal{T}}}(\mathcal{J}) \leq j_{\mathcal{T}}$. Hence, $e^{\Phi}(\mathcal{J}) \leq j < e^{\Phi_{\mathcal{A}}}(\mathcal{I}) + e^{\Phi_{\mathcal{T}}}(\mathcal{I})$. That is $\mathcal{J} \prec_{\Phi}^{\pi,*} \mathcal{I}$, so $\mathcal{I}$ is not a minimal model, which is a contradiction, therefore (1) holds.

Consider case (2). This result follows immediately from Proposition 4.20. $\mathcal{I}$ satisfies $\delta_{rcma-dl}^{e^{\Phi}(\mathcal{I})}(\Phi)$.

($\Longleftarrow$)

Suppose $\mathcal{I}$ satisfies $\delta_{rcma-dl}(\Phi)$, and $\mathcal{I}$ is *not* a $(\preceq^{\pi,*}_{\Phi})$-minimal model. We know that $\delta_{rcma-dl}(\Phi) = \delta^{i}_{rcma-dl}(\Phi)$ for some i, and $\delta^{j}_{rcma-dl}(\Phi)$ is not consistent for all $j < i$. Since $\mathcal{I}$ satisfies $\delta_{rcma-dl}(\Phi)$, it also satisfies $\delta^{i}_{rcma-dl}(\Phi)$ for some i.

By Def.4.18, $\delta^{i}_{rcma-dl}(\Phi) = \delta^{i_{\mathcal{A}}}_{rcma-dl}(\Phi_{\mathcal{A}}) \times \delta^{i_{\mathcal{T}}}_{rcma-dl}(\Phi_{\mathcal{T}})$, where $i = i_{\mathcal{A}} + i_{\mathcal{T}}$. Hence, $\mathcal{I}$ satisfies both $\delta^{i_{\mathcal{A}}}_{rcma-dl}(\Phi_{\mathcal{A}})$ and $\delta^{i_{\mathcal{T}}}_{rcma-dl}(\Phi_{\mathcal{T}})$. By definition $\delta^{i_{\mathcal{A}}}_{rcma-dl}(\Phi_{\mathcal{A}})$ contains a set $\Phi'_{\mathcal{A}}$, where $\Phi'_{\mathcal{A}}$ is a subset of of $\Phi_{\mathcal{A}}$ and $|\Phi'_{\mathcal{A}}| = i_{\mathcal{A}}$. Hence, $\mathcal{I}$ satisfies $\Phi'_{\mathcal{A}}$. Therefore $e^{\Phi}_{\mathcal{A}}(\mathcal{I}) \leq i_{\mathcal{A}}$. By definition $\delta^{i_{\mathcal{T}}}_{rcma-dl}(\Phi_{\mathcal{T}})$ contains the set of all $i_{\mathcal{T}}$-weakenings of $\Phi_{\mathcal{T}}$. It follows that $\mathcal{I}$ satisfies a set $\Phi'_{\mathcal{T}} = \{(\leq e_1(C_1 \sqcap \neg D_1)), \ldots, (\leq e_n(C_n \sqcap \neg D_n))\}$ where $\sum_{i=1}^{n} e^i = i_{\mathcal{T}}$. Therefore $e^{\Phi_{\mathcal{T}}}(\mathcal{I}) \leq \sum_{i=1}^{n} e^i = i_{\mathcal{T}}$. Since $e^{\Phi_{\mathcal{A}}}(\mathcal{I}) \leq i_{\mathcal{A}}$ and $e^{\Phi_{\mathcal{T}}}(\mathcal{I}) \leq i_{\mathcal{T}}$, so we have $e^{\Phi}(\mathcal{I}) \leq i$. But if $e^{\Phi}(\mathcal{I}) < i$, then by Proposition 4.20 $\mathcal{I}$ satisfies $\delta^{j}_{rcma-dl}(\Phi)$ for some $j < i$. It follows that $\delta^{j}_{rcma-dl}(\Phi)$ is consistent for some $j < i$ which is a contradiction. Hence $e^{\Phi}(\mathcal{I}) = i$.

Suppose $\mathcal{I}$ is *not* a $(\preceq^{\pi,*}_{\Phi})$-minimal model, then there is an interpretation $\mathcal{J}$ where $\mathcal{J} \prec^{\pi,*}_{\Phi} \mathcal{I}$, hence $e^{\Phi}(\mathcal{J}) < e^{\Phi}(\mathcal{I})$. By Def. 4.18, we know that the number of Φ-exceptions of $\mathcal{J}$ is $e^{\Phi}(\mathcal{J})$. Hence, by Proposition 4.20, it follows that $\mathcal{J}$ satisfies $\delta^{e^{\Phi}(\mathcal{J})}(\Phi)$. But $e^{\Phi}(\mathcal{J}) < i$, so we have a contradiction.

□

Corollary 4.22 *Let K be a stratified DL knowledge base, $\mathcal{B}$ the DKB obtained from K by RCMA-DL in Algorithm 4, let lexicographic entailment for DLs be defined in terms of Def. 4.18, and let Φ be a DKB. Then $K \models_{lex} \Phi$ if and only if $\mathcal{B} \models \Phi$.*

Proof. This corollary follows immediately from Lemma 4.21. □

4.6 Summary and Discussion

In this chapter, we presented knowledge integration strategies in propositional logic and recasted them into the DL setting. In particular, we explored the adjustment strategies that have been presented in [BKLBW04]. We showed that these adjustments strategies

can be directly used in the DL context. However, this would sometimes lead to counter-intuitive results as we have demonstrated with CMA-DL. We presented in this chapter an original refinement of the CMA-DL that takes advantage of the additional structure provided by DLs. We showed that this would lead to desirable results.

Chapter 5

Ontology Debugging

5.1 Chapter Introduction

In Chapter 3 and 4, the issues of semantic change and integration were discussed in the context of ontologies and we devised a number of original algorithms to resolve inconsistencies by removing and weakening sentences. In this chapter, we study a practical tableau algorithm for resolving inconsistencies in description logic knowledge bases. In particular, we delve into the classical tableau algorithm for checking satisfiability of a concept for given terminologies and incorporate propositional labels to establish a trace to the expansion rules. Such trace is used to compile a propositional formula that can be used to compute the maximally satisfiable subsets, as well as the minimally unsatisfiable subsets of the terminologies.

The main results we present in this chapter are extensions of the labelled consistency algorithm in [BH95]. We extend the labelled tableau-based algorithm for $\mathcal{ALC}$ to provide support for General Inclusion Axioms (GCIs). We show that classical subset blocking is inadequate for this purpose and does not guarantee completeness. We then present a revised blocking condition, known as subset label blocking, and we prove that it ensures both soundness and completeness. Moreover, we show that extending the labelled tableau algorithm to $\mathcal{ALCI}$ and $\mathcal{SI}$ with GCIs would also lead to a similar problem, and we present both a revised equality blocking and a pair-wise blocking for these algorithms. Furthermore, we show that these algorithms preserve both soundness and completeness.

Finally, we present a way to use the trace information constructed from the labelled consistency algorithm to construct Binary Decision Diagrams (BDDs) that can be used as a diagrammatic tool for debugging ontologies.

5.1.1 Motivation

Standard reasoning services in description logics, such as consistency and satisfiability checking, provide a means to detect logical errors in ontologies. These services are useful in many ways. However, there are fundamental limitations to what these services can do. To understand these limitations, consider a knowledge engineer constructing an ontology from scratch. The ontology engineer will start with an empty ontology that is consistent and add sentences into the ontology one at a time until eventually the ontology becomes inconsistent. That means the newly added sentence conflicts with some sentences in the existing ontology. At that point, the ontology engineer is faced with a challenge: whether to revise the newly added sentence, the existing sentences or both. To answer this question, it would be useful to know which of the sentences actually caused the logical errors and the available options to resolve them. However, standard reasoning services do not provide such support.

In light of this challenge, we consider a well-known notion in classical propositional and first-order logics, called maximal consistency. This notion is formally defined as follows: Let Φ be a set of sentences, a maximally consistent subset (of Φ) is any consistent subset Φ' such that if $\phi \in \Phi \setminus \Phi'$ then $\Phi' \cup \{\phi\}$ is inconsistent. Loosely speaking, assuming that Φ is inconsistent[1], a maximally consistent subset of Φ ensures minimal loss of information (in terms of number of sentences) from Φ in order to restore consistency. Another closely related notion is called minimal inconsistency and it is formally defined as follows: Let Φ be a set of sentences, a minimally inconsistent subset (of Φ) is any inconsistent subset Φ' such that if $\phi \in \Phi'$ then $\Phi' \setminus \{\phi\}$ is consistent. Both of these notions can be reformulated in terms of concept satisfiability as follows:

Let Φ be a set of sentences, a maximally satisfiable subset (of Φ) for the concept C (abbreviated as C-MSS or simply MSS) is any satisfiable subset Φ' for C such that

[1] Φ itself is the maximally consistent subset if it is consistent

if $\phi \in \Phi \setminus \Phi'$ then $\Phi' \cup \{\phi\}$ is unsatisfiable. Let Φ be a set of sentences, a minimally unsatisfiable subset (of Φ) for the concept C (abbreviated as C-MUS or simply MUS) is any unsatisfiable subset Φ' for C such that if $\phi \in \Phi'$ then $\Phi' \setminus \{\phi\}$ is satisfiable.

Returning to our earlier example, it would seem useful that the knowledge engineer considers C-MSSes and C-MUSes when trying to resolve logical errors of an ontology. A naive way to compute all C-MSSes of an ontology is by considering all its subsets and using a reasoner to check if each subset is satisfiable or not. However, this approach does not scale well especially when the size of an ontology is large. This is particularly inefficient in the case of DLs that are intractable because these languages are expressive and concept satisfiability is generally an expensive operation. Therefore, our goal in this chapter is to investigate an efficient approach to computing all MSSes (or MUSes) of an ontology without the need to use a reasoner as a black-box. Instead, we delve into the implementation of the classical tableau algorithm and propose to incorporate labels that will enable us to find all the MSSes (or MUSes).

In contrast to earlier chapters, the aim of this chapter is not to present another theoretical operation. Rather, we consider practical aspects of implementing two well-known operations (MSS and MUS). Both MSS and MUS are closely related to the operations we studied in earlier chapters, in particular Maxi-Adjustment proposed by Benferhat et al. [BKLBW04]. One main difference between ontology debugging and ontology revision is that we do not give preference to any sentence in ontology debugging. That is, all sentences are treated equally. On the other hand, classical revision gives preference to the newly added information.

In this section, we present a brief survey of the existing work.

5.1.2 Related Work

Approaches to resolving logical errors can roughly be divided into two categories. The first seeks to pinpoint possible problematic statements and leaves it up to the modeller to rectify them. Examples of this include the work of Schlobach et al. [SC03, Sch05a, SHCvH07], Kalyanpur et al. [KPS05, KPSH05, KPSCG06] and Lam et al. [LSPV08]. The second aims to resolve the problem directly by weakening the available information

to the extent that the errors are eliminated. Examples of this include the work of Baader et al. [BH95], Schlobach et al. [Sch05b], work based on model-based diagnosis [Rei87], and Meyer et al. [MLB05, MLBP06].

Baader et al. [BH95] considered the problem of finding maximally satisfiable and minimally unsatisfiable ABoxes in a slightly different context from ours. The techniques they used correspond closely to our approach if we assume that the terminologies are unfoldable. Schlobach et al. [SC03] provide an algorithm that is closely related to ours for finding Minimally Unsatisfiable Subsets (MUSes). Schlobach [Sch05b] considers the problem of finding Maximally Satisfiable Subsets (MSSes) assuming that terminologies unfoldable. His approach is to first find the MUSes of the terminologies and then to use the minimal hitting-set algorithm described by Reiter [Rei87] to calculate the A-MSSes.

Meyer et al. [MLBP06] also provide a method for finding MSSes if Γ is unfoldable. Their mechanism for labelling concept assertions is slightly different from ours (they use sets of indices). Instead, they employ a version of the $\bot$-rule to determine which TBox sentences to exclude. None of these techniques can deal with GCIs and cyclic terminologies and none of them, except for Meyer et al., consider approaches to provide incrementally better approximations of MSSes. Kalyanpur et al. [KPS05] provide an algorithm for finding a single MUS for a TBox expressed in the more expressive DL $\mathcal{SHIF}$.

In [KPGS05], Kalyanpur et al. described a tableau-based algorithm for finding MUSes for an unsatisfiable concept in $\mathcal{SHIQN}$. In contrast to our approach, they have used multisets as labels to keep track of dependencies while we use propositional formulas as shown in [BH95]. This has a number of advantages. Firstly, propositional formula is a more compact representation than multiset. It retains structural information generated during rule applications and consumes very little memory. Secondly, there is a close relationship between propositional formulas and binary decision diagrams (BDDs). There are established methods to convert propositional formulas to various types of BDDs. As we shall see in this chapter, BDD is a nice diagrammatic tool for debugging ontologies.

A recent line of research has examined the problem of computing justifications for entailments, including the work by Horridge et al. [HPS08, HBPS08] and Kalyanpur et

al. [KPHS07]. They considered the problem of identifying a minimal set of axioms that is sufficient for a given entailment to hold. This problem is similar to that of finding minimal set of axioms that explains an unsatisfiable concept in the sense that entailment can be reduced to satisfiability checking.

The problem of axiom pinpointing has also been explored in automata-based reasoning [BPn10], as well as for the description logic $\mathcal{EL}^{++}$ [BS08].

5.2 Labelled Consistency Algorithm

In this section we present a procedure known as labelled consistency algorithm that can be used to establish a trace to the classical tableau algorithm and representing this trace as a propositional formula. The labelled consistency algorithm can be seen as an extension to the classical tableau algorithm. It is composed of a number of expansion rules that look similar to the expansion rules in the classical (or unlabelled) tableau algorithm. As with the (unlabelled) tableau algorithm, each rule consists of a precondition and an action. The action is triggered if the precondition is satisfied. An action transforms a labelled ABox $\mathcal{A}$ into either one ABox $\mathcal{A}'$ or two ABoxes $\mathcal{A}'$ and $\mathcal{A}''$.

In contrast to the classical tableau algorithm, each sentence (axiom or assertion) in the labelled consistency algorithm is associated with a label. $C(x)^{\phi}$ and $R(x,y)^{\psi}$ denote assertions labelled with propositional formulas ϕ and ψ respectively. Similarly, $(C \sqsubseteq D)^{\phi}$ and $(C \doteq D)^{\psi}$ denotes axioms labelled with the propositional formulas ϕ and ψ respectively. We refer to an ABox with all its assertions labelled a labelled ABox, and similarly, a TBox with all its axioms labelled a labelled TBox. In general, the labelled consistency algorithm does not require labels to be in a particular representation.

As demonstrated in [Sch05b] and [MLBP06], labels can be represented as multisets of indices. However, we have adopted the representation used in [BH95] where labels are propositional formulas. We argue that this is a more desirable representation for labels than multisets for a number of reasons. Firstly, a propositional formula is a more compact way of representing information compared to multisets and its semantics and properties are well-understood. For example, a sentence with the label $p \vee (p \wedge q)$ can simply be

reduced to p since they are logically equivalent. In contrast, the formula $(p \wedge (q \vee r))$ would be represented as $\{\{p, q\}, \{p, r\}\}$ with multisets. This multisets have the implicit meaning that the outer set is a disjunction and the inner multisets are conjunctions of indices. Secondly, there is a large body of literatures relating to propositional logic. In particular, there exists highly optimised and efficient solver to check for satisfiability (hence also entailment) of propositional formulas, and there are efficient data structures for representing propositional (boolean) formulas and functions, such as the various types of Binary Decision Diagrams (BDDs). As we shall see in later sections, BDDs can be a useful tool to visualise ways to repair an ontology.

The labelled consistency algorithm uses labels as a means to keep track of relationships among sentences. More specifically, each expansion rule is triggered by some sentences (usually one), which in turns generates other new sentences. It is, therefore, useful to retain this information so that once an inconsistency is identified we could trace back to its sources and act on them to restore consistency. A sentence labelled with $p \wedge q$ is interpreted as the sentence that originates from sentences labelled with both p and q, which means if either p or q is disregarded then the sentence labelled with $p \wedge q$ will also disappear. We shall now describe how the labelled consistency algorithm operates.

The algorithm starts by labelling each axiom in the TBox with a unique propositional variable, hence this results in a labelled TBox $\mathcal{T}$ with elements taking the form of $(C \sqsubseteq D)^p$ and $(C \doteq D)^p$. These variables simply serve as unique identifiers so we could distinguish between the axioms. Next, we construct a labelled ABox $\mathcal{A}$ that contains only the labelled assertion $A(x)^\top$, where A is the concept that we want to check for satisfiability and x is an individual name. Labelling $A(x)$ with $\top$ simply indicates that this instance of $A(x)$ is not attributable to any of the TBox axioms. It also indicates that $A(x)$ cannot be disregarded and that its presence does not depend on any other sentences. Therefore, removing $A(x)$ as a way to restore consistency is not an option.

The next step is then to apply the labelled expansion rules in Figure 5.1. First, we check if any of the rules is applicable to $\mathcal{A}$. If so, we proceed to the ABoxes created subsequently and again check if they are applicable to any of the labelled rules and so on. We continue until there is an ABox to which none of the rules can be applied and

which does not contain any clashes. Then we have generated a clash-free model and so A is satisfiable. Otherwise, we have encountered a clash and we have to try other alternatives. But before we do so, we have to store information that leads to this clash. Note that for $\mathcal{ALC}$, a *clash* in $\mathcal{A}$ is defined as a subset $\{A(x)^{\phi}, \neg A(x)^{\psi}\}$ of $\mathcal{A}$. [2]

The culprit of the clash is therefore $\phi \wedge \psi$ and we store this globally for later use. It should be noted that the use of disjunction in the label is merely for the purpose of having a more compact representation. It is possible to restrict ourselves to use only conjunctions in our labels and allow concept assertions with the same concept description but different labelling to present in the ABox. For example, $C(s)^{p \vee q}$ can be represented as $C(s)^{p}$ and $C(s)^{q}$.

The algorithm presented in Figure 5.1 is identical to that in [BH95] with the exception that the $\exists$-rule is inserted with a condition to check if an individual is (label) blocked. This is necessary to ensure termination and completeness. We shall see in later parts that classical blocking is not adequate and a stronger blocking condition is necessary to ensure completeness in the case where cyclic definitions are permitted in the TBox. We have simplified the notations to make the expansion rules appear like the ones in the classical tableau algorithm. We introduce two additional operators $\tilde{\in}$ and $\tilde{\cup}$. The former is a variant of the set containment operator it considers not only set membership but also takes into account the label attached to the assertion.

Roughly speaking, a labelled concept assertion $C(s)^{\phi}$ is label contained in $\mathcal{A}$ if either $\mathcal{A}$ contain the concept assertion $C(s)$ and this concept assertion is attached with a label that is logically weaker than ϕ. The latter is a variant of the union operator. Again, this operator takes the labels of the assertions into account. It either adds the concept assertion together with the label if the concept assertion is not already in the labelled ABox or it updates the label of the concept assertion by taking disjunction of the new label with the existing label. These two operators are formally defined as follows:

Definition 5.1

[2] We assume that concept assertions added to these ABoxes have all been converted to the negation normal form.

A set of labelled assertions Φ label contains a labelled assertion α^ϕ, or $\alpha^\phi \in \Phi$, if and only if $\alpha^\psi \notin \Phi$ for any ψ or there exists $\alpha^\psi \in \Phi$ and ϕ implies ψ.

This definition applies to both concept assertions and role assertions. A labelled ABox $\mathcal{A}$ label contains a labelled concept assertion $C(x)^\phi$, or $C(x)^\phi \,\tilde{\in}\, \mathcal{A}$, if $\mathcal{A}$ contains a concept assertion $C(x)^\psi$ such that ϕ implies ψ. Similarly, a labelled ABox $\mathcal{A}$ label contains a labelled role assertion $R(s,t)^\phi$, or $R(s,t)^\phi \,\tilde{\in}\, \mathcal{A}$, if $\mathcal{A}$ contains a role assertion $R(s,t)^\psi$ such that ϕ implies ψ.

Definition 5.2

A function $\tilde{\cup}$ that extends a labelled ABox $\mathcal{A}$ by a set of labelled assertions is inductively defined as follows:

$$\mathcal{A} \,\tilde{\cup}\, \{\alpha_1, \ldots, \alpha_n\} = (\mathcal{A} \,\tilde{\cup}\, \{\alpha_1\}) \,\tilde{\cup}\, \{\alpha_2, \ldots, \alpha_n\}$$

$$\mathcal{A} \,\tilde{\cup}\, \{\alpha\} = \begin{cases} (\mathcal{A} \setminus \{\alpha\}) \cup \{C(x)^{\phi \vee \psi}\} & \text{if } \alpha = C(x)^\psi \text{ and } C(x)^\phi \in \mathcal{A}; \\ (\mathcal{A} \setminus \{\alpha\}) \cup \{R(x,y)^{\phi \vee \psi}\} & \text{if } \alpha = R(x,y)^\psi \text{ and } R(x,y)^\phi \in \mathcal{A}; \\ \mathcal{A} \cup \{\alpha\} & \text{otherwise.} \end{cases}$$

$\sqcap$-rule	**if** $(C_1 \sqcap C_2)(x)^\phi \in \mathcal{A}$, and either $C_1(x)^\phi \,\tilde{\notin}\, \mathcal{A}$ or $C_2(x)^\phi \,\tilde{\notin}\, \mathcal{A}$, **then** $\mathcal{A}' := \mathcal{A} \,\tilde{\cup}\, \{C_1(x)^\phi, C_2(x)^\phi\}$.
$\sqcup$-rule	**if** $(C_1 \sqcup C_2)(x)^\phi \in \mathcal{A}$, and both $C_1(x)^\phi \,\tilde{\notin}\, \mathcal{A}$ and $C_2(x)^\phi \,\tilde{\notin}\, \mathcal{A}$, **then** $\mathcal{A}' := \mathcal{A} \,\tilde{\cup}\, \{C_1(x)^\phi\}$, $\mathcal{A}'' := \mathcal{A} \,\tilde{\cup}\, \{C_2(x)^\phi\}$.
$\exists$-rule	**if** $(\exists R.C)(x)^\phi \in \mathcal{A}$, x is not label blocked, and there does not exist an individual y such that $R(x,y)^\phi \,\tilde{\in}\, \mathcal{A}$ and $C(y)^\phi \,\tilde{\in}\, \mathcal{A}$, **then** $\mathcal{A}' := \mathcal{A} \,\tilde{\cup}\, \{R(x,y)^\phi, C(y)^\phi\}$, where y is a new individual name and $y > y'$ for all individual names y' in $\mathcal{A}$.
$\forall$-rule	**if** $\{(\forall R.C)(x)^{\phi_1}, R(x,y)^{\phi_2}\} \subseteq \mathcal{A}$, and $C(y)^{\phi_1 \wedge \phi_2} \,\tilde{\notin}\, \mathcal{A}$, **then** $\mathcal{A}' := \mathcal{A} \,\tilde{\cup}\, \{C(y)^{\phi_1 \wedge \phi_2}\}$.

Figure 5.1: Labelled consistency algorithm for $\mathcal{ALC}$ with GCIs

To understand how the expansion rules work, a number of comments are in order. Firstly, observe that each assertion in $\mathcal{A}$, and in subsequent ABoxes, has attached to it a propositional formula. These formulas encode the conditions (or reasons) for the assertion appearing in the ABox. For example $B(x)^p$ means that $B(x)$ appears in the ABox because of the TBox sentence labelled with p. Disjunctions represent multiple occurrences of the same assertion, while conjunctions represent the dependence of the assertion on more than one TBox sentence. For example, $B(x)^{(p \vee q)}$ can be seen as shorthand for both $B(x)^p$ and $B(x)^q$ occurring in the ABox. On the other hand, $B(x)^{(p \wedge q)}$ means that $B(x)$ appears in the ABox because of the simultaneous presence of the sentences labelled with p and q in the TBox.

An mentioned earlier, a labelled rule is similar to that of the unlabelled version where each rule is composed of a precondition and an action. For example, the precondition of the labelled $\sqcap$-rule has two parts. The first part checks whether the labelled concept assertion $(C_1 \sqcap C_2)(x)^\phi$ is contained in the labelled ABox $\mathcal{A}$ where C_1 and C_2 can be any concept descriptions. The second part makes use of Def. 5.1. It checks whether $C_1(x)^\phi$ and $C_2(x)^\phi$ are label contained in $\mathcal{A}$. If the precondition is satisfied then the action is triggered. In this case we trigger the generation of a new labelled ABox $\mathcal{A}'$ where $\mathcal{A}'$ incorporates both $C_1(x)^\phi$ and $C_2(x)^\phi$ using the label union operator in Def. 5.2. The other labelled rules operate in a similar fashion as the $\sqcap$-rule. Next, we will introduce the notion of Maximally Satisfiable Subsets (MSS) and its dual Minimally Unsatisfiable Subsets (MUS).

5.2.1 Supplementary Labelled Consistency Rules

The classical tableau algorithm is often augmented with a number of additional expansion rules. These rules serve to provide a form of optimisation known as absorption to the reasoning algorithm. In particular, they are often used to avoid cases where non-deterministic branches are introduced. As we shall see below, the expansion rules presented in Figure 5.1 are sufficient to guarantee both soundness and completeness. However, we introduce here a number of supplementary labelled rules. Similarly to the classical tableau algorithm, these rules can be used to improve the performance of the

algorithm. Recall that, Γ is *unfoldable* if and only if the left-hand side of every $\gamma \in \Gamma$ contains a concept name A, that there are no other γs with A on the left-hand side, and that the right-hand side of γ contains no direct or indirect references to A (i.e. no cycles).

We divide Γ into an unfoldable part Γ_u and a general part Γ_g, such that $\Gamma_g = \Gamma \setminus \Gamma_u$.

$U^{+}_{\doteq}$-rule **if** $(A \doteq C)^{\phi} \in \Gamma_u$, $A(x)^{\psi} \in \mathcal{A}$, and $C(x)^{\phi \wedge \psi} \tilde{\notin} \mathcal{A}$,
then $\mathcal{A}' := \mathcal{A} \tilde{\cup} \{C(x)^{\phi \wedge \psi}\}$.

$U^{-}_{\doteq}$-rule **if** $(A \doteq C)^{\phi} \in \Gamma_u$, $\neg A(x)^{\psi} \in \mathcal{A}$ and $\neg C(x)^{\phi \wedge \psi} \tilde{\notin} \mathcal{A}$,
then $\mathcal{A}' := \mathcal{A} \tilde{\cup} \{\neg C(x)^{\phi \wedge \psi}\}$.

$U_{\sqsubseteq}$-rule **if** $(A \sqsubseteq C)^{\phi} \in \Gamma_u$, $A(x)^{\psi} \in \mathcal{A}$ and $C(x)^{\phi \wedge \psi} \tilde{\notin} \mathcal{A}$,
then $\mathcal{A}' := \mathcal{A} \tilde{\cup} \{C(x)^{\phi \wedge \psi}\}$.

$\sqsubseteq$-rule **if** $(C \sqsubseteq D)^{\phi} \in \Gamma_g$ and $(\neg C \sqcup D)(x)^{\phi} \tilde{\notin} \mathcal{A}$ for some individual name x,
then $\mathcal{A}' := \mathcal{A} \tilde{\cup} \{(\neg C \sqcup D)(x)^{\phi}\}$.

$\doteq$-rule **if** $(C \doteq D)^{\phi} \in \Gamma_g$ and $((\neg C \sqcup D) \sqcap (C \sqcup \neg D))(x)^{\phi} \tilde{\notin} \mathcal{A}$ for some individual name x,
then $\mathcal{A}' := \mathcal{A} \tilde{\cup} \{((\neg C \sqcup D) \sqcap (C \sqcup \neg D))(x)^{\phi}\}$.

Figure 5.2: Supplementary rules of the labelled consistency algorithm

The set of supplementary labelled rules composed of five rules. The first three rules are only applicable to unfoldable terminologies and they are targeted towards concept definitions, which often constitute a large portion of the terminologies. It should be noted that each of these rules transform one ABox $\mathcal{A}$ into another ABox $\mathcal{A}'$, and does not introduce any non-determinism. The last two rules are applied to general terminologies and they are necessary for terminologies that are not unfoldable to ensure completeness of the algorithm. Similar to the previous rules, these two rules also transform one ABox $\mathcal{A}$ into a new ABox $\mathcal{A}'$ but they do introduce non-determinism by incorporating disjunctions. This will eventually trigger the $\sqcup$-rule in Figure 5.1, which in turn creates two ABoxes. For convenience, we have adopted the supplementary labelled rules in our examples.

5.2.2 Maximally Satisfiable Subsets and Minimally Unsatisfiable Subsets

The notion of Maximally Satisfiable Subset (MSS) [3] has been introduced in various contexts, most notably in propositional logic. Here, we introduce MSS as a criterion for repairing ontologies. The idea is consistent to the principle of minimal change where information change is kept to a minimal. In the course of restoring consistency to an ontology, there are many ways in which this can be performed but not all ways are necessarily useful. For example, one could choose to remove every axiom in an ontology in order to restore its consistency. This would make it consistent but it is not useful at all. A maximally satisfiable subset guarantees that an axiom of the ontology is not to be removed unless this is necessary to restore consistency. The dual of MSS is that of a Minimally Unsatisfiable Subset (MUS). It is possible to compute all the MUSes given all the MSSes, and vice versa. The formal definition of MSSes and MUSes are stated below.

Definition 5.3

A subset $\mathcal{T}'$ of terminologies $\mathcal{T}$ is a C-MSS or a maximally satisfiable subset of $\mathcal{T}$ with respect to a concept C if and only if C is satisfiable, and every $\mathcal{T}''$ such that $\mathcal{T}' \subset \mathcal{T}'' \subseteq \mathcal{T}$ is unsatisfiable. A subset $\mathcal{T}'$ of terminologies $\mathcal{T}$ is a C-MUS or a minimally unsatisfiable subset of $\mathcal{T}$ with respect to a concept C if and only if C is unsatisfiable, and every $\mathcal{T}''$ such that $\mathcal{T}'' \subset \mathcal{T}'$ is satisfiable.

It is conventional to refer to A-MSS and A-MUS as MSS and MUS respectively. We will adopt this convention in situations where it is clear which concept we are referring to. Note that MUS is more commonly referred to as MUPS [4] in the literature [SC03].

Next, we describe how to make use of the information encoded in the propositional formulas associated with clashes to generate the MSSes or the MUSes of a TBox $\mathcal{T}$.

Consider a clash $\{A(x)^{\phi}, \neg A(x)^{\psi}\}$ occurring in an ABox $\mathcal{A}$ and recall that ϕ is an

[3] The concept of Maximally Satisfiable Subsets of terminologies is also known as Maximally Satisfiable Sub-terminologies.

[4] MUPS is used as an abbreviation for *minimal unsatisfiability-preserving sub-TBoxes*.

encoding of the reasons for $A(x)$ occurring in $\mathcal{A}$ (and similarly for ψ and $\neg A(x)$). So the formula $\phi \wedge \psi$ can be seen as an encoding of the reasons for both $A(x)$ and $\neg A(x)$ occurring in $\mathcal{A}$, and therefore $\neg(\phi \wedge \psi)$ encodes the reasons for $A(x)$ and $\neg A(x)$ not occurring simultaneously in $\mathcal{A}$. That is, $\neg(\phi \wedge \psi)$ expresses the different ways in which we can remove the clash. So, let $\mathcal{A}_1, \ldots, \mathcal{A}_n$ be the completely expanded ABoxes obtained from running the labelled consistency algorithm. Recall that these ABoxes represent the different paths generated by the expansion rules. Now suppose there are k_i clashes in $\mathcal{A}_i$ with ϕ_i^j and ψ_i^j the propositional formulas associated with the two concept assertions in clash j.

Now, recall that running a tableau algorithm on an unsatisfiable concept results in ABoxes where *each* ABox contains at least one clash. That is, for an ABox $\mathcal{A}_i$ that contains clashes $C_1^{\phi_1}$,$\neg C_1^{\psi_1}$,...,$C_n^{\phi_n}$, $\neg C_n^{\psi_n}$, at least one of $\phi_j \wedge \psi_j$ evaluates to true, or equivalently $\bigvee_{j=1}^{m_i}(\phi_i^j \wedge \psi_i^j)$ evaluates to true. Since each of the ABoxes is required to have at least one clash for a concept to be unsatisfiable, we have the clash formula as stated in [BH95]. The clash formula is $\bigwedge_{i=1}^{n} \bigvee_{j=1}^{m_i}(\phi_i^j \wedge \psi_i^j)$ where n is the number of ABoxes and m_i is the number of clashes in the i-th ABox. Furthermore, primes implicates and prime implicants of the clash formula correspond to the MSSes and MUSes of the terminologies respectively. The following example illustrates a simple application of the algorithm. We note that, in this case, classical blocking happens to be sufficient.

Example 5.4

This example demonstrates a straight application of the labelled tableau algorithm (with the supplementary expansion rules). Consider a TBox $\mathcal{T}$ with the following axioms:

$$\mathcal{T}_u \begin{cases} (A \doteq (\forall R.\neg C) \sqcap D)^p \\ (D \doteq \neg E)^q \end{cases}$$

$$\mathcal{T}_g \begin{cases} (\forall R.\neg C \sqsubseteq \neg D \sqcap \neg E)^r \end{cases}$$

The axioms are labelled with the propositional variables p, q and r. The axioms

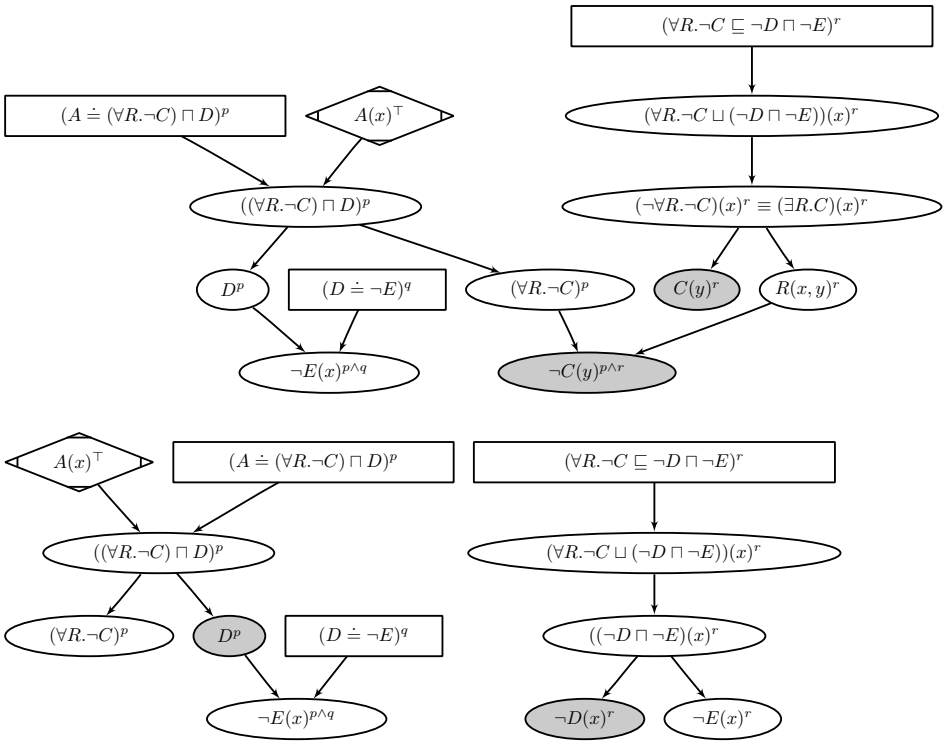

Figure 5.3: The two implication graphs (top and bottom) correspond to the labelled ABoxes generated in Example 5.4. Diamond is the starting point, rectangles are the axioms and filled ellipses are the clashes.

labelled with p and q are in $\mathcal{T}_u$ and the axiom labelled with r is in $\mathcal{T}_g$. Recall that $\mathcal{T}_u$ is the unfoldable part of $\mathcal{T}$, with which the supplementary expansion rules can be applied. Our goal is to compute the MSSes and MUSes for $\mathcal{T}$.

Suppose A is the concept (or concept name) that we wish to check for satisfiability and also give "reasons" if it is unsatisfiable. That is, A is the input to our algorithm. We begin by initialising $\mathcal{A}$ to $\{A(x)^\top\}$. Applying the $U^+_{\doteq}$-rule to $A^\top$ followed by the $\sqcap$-rule gives the new ABox $\{A(x)^\top, ((\forall R.\neg C) \sqcap D)^p, (\forall R.\neg C)(x)^p, D(x)^p\}$. Applying the $U^+_{\doteq}$-rule to $D(x)^p$ adds $\neg E(x)^{p\wedge q}$ to the ABox. Now, apply the $\sqsubseteq$-rule to $(\forall R.\neg C \sqsubseteq (\neg D \sqcap \neg E))^r$ for x to add $(\neg\forall R.\neg C \sqcup (\neg D \sqcap \neg E))(x)^r$ to the ABox. The $\sqcup$-rule now becomes applicable. Applying it creates two new ABoxes. One branch adds $(\neg\forall R.\neg C)(x)^r$ to $\mathcal{A}'$, but this is

not in the negation normal form, so we add $(\exists R.C)(x)^r$ instead to obtain the new ABox $\mathcal{A}'$. Applying the $\exists$-rule to $(\exists R.C)(x)^r$ adds $C(y)^r$ and $R(x,y)^r$ to $\mathcal{A}'$. Now, applying the $\forall$-rule to $(\forall R.\neg C)(x)^p$ adds $\neg C(y)^{p\wedge r}$ to $\mathcal{A}'$. The other branch adds $(\neg D \sqcap \neg E)(x)^r$ to obtain the new ABox $\mathcal{A}''$. Applying the $\sqcap$-rule adds $\neg D(x)^r$ and $\neg E(x)^{(p\wedge q)\vee r}$ to $\mathcal{A}''$. None of the expansion rules are now applicable and we are left with the two ABoxes: $\{A(x)^\top$, $((\forall R.\neg C) \sqcap D)(x)^p$, $(\forall R.\neg C)^p$, $D(x)^p$, $(\neg\forall R.\neg C \sqcup (\neg D \sqcap \neg E))(x)^r$, $\exists R.C(x)^r$, $C(y)^r$, $R(x,y)^r$, $\neg C(y)^{p\wedge r}\}$ and $\{A(x)^\top$, $((\forall R.\neg C) \sqcap D)(x)^p$, $(\forall R.\neg C)^p$, $D(x)^p$, $(\neg\forall R.\neg C \sqcup (\neg D \sqcap \neg E))(x)^r$, $(\neg D \sqcap \neg E)(x)^r$, $\neg D(x)^r$, $\neg E(x)^{(p\wedge q)\vee r}\}$. The set of clashes in $\mathcal{A}'$ is $\{C(y)^r, \neg C(y)^{p\wedge r}\}$, and the set of clashes in $\mathcal{A}''$ is $\{D(x)^p, \neg D(x)^r\}$. The clash formula for these ABoxes is $(r \wedge (p \wedge r)) \wedge (p \wedge r)$, which is logically equivalent to $p \wedge r$ and so the prime implicates are p and r. Hence, there are two MSSes for $\mathcal{T}$. One in which the sentence labelled with p is removed, and one in which the sentence labelled with r is removed. Similarly, $p \wedge r$ has only one prime implicate which is $p \wedge r$ itself. So $\mathcal{T}$ has one MUS: the set containing the two sentences labelled with p and r.

5.3 Refined Blocking

As pointed out in Chapter 2, blocking in classical $\mathcal{ALC}$ tableau reasoning ensures correctness and termination. In the case of our algorithm, classical blocking still ensures termination, but it does not always block correctly at the right point, as the following example shows.

$$\mathcal{T}_u \begin{cases} (A \doteq \neg C \sqcap D \sqcap E \sqcap F \sqcap \exists R.A)^p \\ (D \doteq C)^q \\ (E \doteq \forall R.C)^r \\ (F \doteq \forall R.\forall R.C)^s \end{cases}$$

Example 5.5

Let $\mathcal{T} = \mathcal{T}_u$. To compute the MSSes and MUSes for $\mathcal{T}$ we initialise the ABox $\mathcal{A}$ to $\{A(x)^\top\}$. An application of the $U^+_{\doteq}$-rule followed by the $\sqcap$-rule adds $(C \sqcap D \sqcap E \sqcap F \sqcap \exists R.A)^p$, $\neg C(x)^p$, $D(x)^p$, $E(x)^p$, $F(x)^p$, and $(\exists R.A)(x)^p$. Applying the $U^+_{\doteq}$-rule to $D(x)^p, E(x)^p$ and $F(x)^p$ adds $C(x)^{p\wedge q}$, $(\forall R.C)(x)^{p\wedge r}$, and $(\forall R.\forall R.C)(x)^{p\wedge s}$. Applying the $\exists$-rule to $\exists R.A(x)^p$ will then add $A(y)^p$ and $R(x,y)^p$, where y is a new individual. Applications of the $\forall$-rule then adds $C(y)^{p\wedge r}$, $(\forall R.C)(y)^{p\wedge s}$. An application of the $U^+_{\doteq}$-rule to $A(y)^p$ followed by the $\sqcap$-rule adds $(C \sqcap D \sqcap E \sqcap F \sqcap \exists R.A)^p$, $\neg C(y)^p$, $D(y)^p$, $E(y)^p$, $F(y)^p$, and $(\exists R.A)(y)^p$. Applying the $U^+_{\doteq}$-rule to $D(y)^p, E(y)^p$ and $F(y)^p$ adds $C(y)^{(p\wedge q)\vee(p\wedge r)}$, $(\forall R.C)(y)^{(p\wedge r)\vee(p\wedge s)}$, and $(\forall R.\forall R.C)(y)^{p\wedge s}$. If we now attempt to apply the $\exists$-rule to $(\exists R.A)(y)^p$, using the classical blocking condition for $\mathcal{ALC}$, then y will be blocked by x since $\{C \mid C(y)^\phi \in \mathcal{A}\} \subseteq \{C' \mid C'(x)^\psi \in \mathcal{A}\}$, and the algorithm will terminate. The set of clashes in this ABox is $\{\neg C(x)^p, C(x)^{p\wedge q}\}$. The clash formula is therefore $p \wedge (p \wedge q)$ and so the prime implicates of the formula are p and q. According to this, the two MSSes of $\mathcal{T}$ are obtained by (i) removing the sentence indexed by p, and (ii) removing the sentence indexed by q. But while the first of these is indeed an MSS, it is easy to verify that the second one isn't. Similarly, the sole prime implicant of the formula $p \wedge (p \wedge q)$ is $p \wedge q$ and according to this $\mathcal{T}$ thus has one MUS: the set $\{(A \doteq \neg C \sqcap D \sqcap E \sqcap F \sqcap \exists R.A)^p, (D \doteq C)^q\}$. But in fact $\mathcal{T}$ has two other MUSes as well.

5.3.1 Subset Label Blocking

We now introduce the notion of label subset blocking. For convenience, we define $\mathcal{L}_\mathcal{A}(s) = \{C^\phi | C(x)^\phi \in \mathcal{A}\}$, or we simply use $\mathcal{L}(s)$ instead of $\mathcal{L}_\mathcal{A}(s)$ where it is clear which ABox we are referring to.

The reason that classical blocking does not work in the above example, is that the labels (the propositional formulas associated with sentences) are not taken into account when blocking is performed. So, in the example above, blocking occurs despite the fact that the label of $C(y)^{(p\wedge q)\vee(p\wedge r)}$ differs from that of $C(x)^{p\wedge q}$.

What we need to do is take the label into account in an appropriate way. This is

achieved by replacing the existential rule in Figure 5.4 by our subset label blocking. Subset label blocking is formally defined as follows:

Definition 5.6

Let Φ and Ψ be sets of labelled assertions. $\Phi \tilde{\subseteq} \Psi$ if and only if for each labelled assertion $\tilde{\alpha} \in \Phi$ implies $\tilde{\alpha} \tilde{\in} \Psi$. Equivalently, $\Phi \tilde{\subseteq} \Psi$ if and only if it satisfies:

1. *if $\alpha^{\phi} \in \Phi$ then $\alpha^{\psi} \in \Psi$ (ϕ and ψ may be different)*
2. *For each assertion α such that $\alpha^{\phi} \in \Phi$ and $\alpha^{\psi} \in \Psi$, ϕ implies ψ*

Definition 5.7 (Subset Label Blocking)

Let $\pi(x) = \{C^{\phi}(x) | C^{\phi}(x) \in \mathcal{A}\}$. An individual x subset label blocks an individual y if and only if $\pi(x) \tilde{\subseteq} \pi(y)$ and x is an ancestor of y.

An individual x is an ancestor of y if x is created before y by the existential rule given in Figure 5.4.

Informally, the justification for this refinement is the following: the blocking of an individual y is intended to capture the case where every possible expansion of a concept assertion involving y will mirror an expansion applied to concept assertions involving its parent individual x. Now, suppose an ABox $\mathcal{A}$ contains the labelled concept assertions $C(x)^{\phi}$ and $C(y)^{\psi}$. If ψ does not imply ϕ, it means that the reasons for $C(y)$ occurring in $\mathcal{A}$ are not contained in the reasons for $C(x)$ occurring in $\mathcal{A}$, therefore the expansion of $C(y)$ might yield results that will not be mirrored by an expansion of $C(x)$. On the other hand, if ψ implies ϕ it means that every reason for $C(y)$ occurring in $\mathcal{A}$ will also be a reason for $C(x)$ occurring in $\mathcal{A}$, and so every expansion of $C(y)$ would be mirrored by an expansion of $C(y)$. We return to the previous example.

Example 5.8 (Example 5.5 revisited)
Since $C(y)^{(p \wedge q) \vee (p \wedge r)}$ and $C(x)^{p \wedge q}$ are in $\mathcal{A}$, but $(p \wedge q) \vee (p \wedge r) \not\models p \wedge q$, y will not be blocked by x when we apply the $\exists$-rule to $(\exists R.A)(y)^p$ using subset label blocking. Applying the $\exists$-rule will add $A(z)^p$ and $R(y,z)^p$, where z is a new individual. Applications of the $\forall$-rule then adds $C(z)^{(p \wedge r) \vee (p \wedge s)}$, $(\forall R.C)(z)^{p \wedge s}$. An application of the $U^{+}_{\doteq}$-rule to $A(z)^p$

followed by the $\sqcap$-rule adds $\neg C(z)^p$, $D(z)^p$, $E(z)^p$, $F(z)^p$, and $(\exists R.A)(z)^p$. Applying the $U^+_{\dot{=}}$-rule to $D(z)^p, E(z)^p$ and $F(z)^p$ adds $C(z)^{(p\wedge q)\vee(p\wedge r)\vee(p\wedge s)}$, $(\forall R.C)(z)^{(p\wedge r)\vee(p\wedge s)}$, and $(\forall R.\forall R.C)(z)^{p\wedge s}$. Since $C(z)^{(p\wedge q)\vee(p\wedge r)\vee(p\wedge s)}$ and $C(y)^{(p\wedge q)\vee(p\wedge r)}$ are in $\mathcal{A}$, but $(p \wedge q) \vee (p \wedge r) \vee (p \wedge s)$ does not imply $(p \wedge q) \vee (p \wedge r)$, z will not be blocked by y when we apply the $\exists$-rule to $(\exists R.A)(z)^p$ using subset label blocking. It can then be readily verified that continued expansion will result in the following being added to the ABox (with z' being a new individual): $A(z')^p$, $R(z, z')^p$, $C(z')^{(p\wedge q)\vee(p\wedge r)\vee(p\wedge s)}$, $(\forall R.C)(z')^{(p\wedge r)\vee(p\wedge s)}$, $\neg C(z')^p$, $D(z')^p$, $E(z')^p$, $F(z')^p$, $(\exists R.A)(z')^p$, and $(\forall R.\forall R.C)(z')^{p\wedge s}$. When attempting to apply the $\exists$-rule to $(\exists R.A)(z')^p$, the subset label blocking ensures that z' is blocked by z, and so the algorithm terminates. The set of ABox clashes is $\{\neg C(x)^p, C(x)^{p\wedge q}, \neg C(y)^p, C(y)^{(p\wedge q)\vee(p\wedge r)}, \neg C(z)^p, C(z)^{(p\wedge q)\vee(p\wedge r)\vee(p\wedge s)}\}$. Therefore, the clash formula is $(p \wedge (p \wedge q)) \vee (p \wedge ((p \wedge q) \vee (p \wedge r))) \vee (p \wedge ((p \wedge q) \vee (p \wedge r) \vee (p \wedge s)))$ which is logically equivalent to $(p \wedge q) \vee (p \wedge r) \vee (p \wedge s)$. The prime implicates of this formula are p and $q \vee r \vee s$. Therefore, the two MSSes of $\mathcal{T}$ are obtained by (i) removing the sentence indexed by p, and (ii) removing the sentences indexed by q, r, and s. Similarly, the prime implicants of the clash formula are $p \wedge q$, $p \wedge r$ and $p \wedge s$, which correspond to the MUSes of $\mathcal{T}$.

5.3.2 Soundness, Completeness and Termination

We study in this section a number of important properties of the labelled consistency algorithm for $\mathcal{ALC}$ with GCIs. Results of soundness and completeness for the $\mathcal{ALC}$ case without GCIs have been proven in [BH95]. The soundness proof is very similar between the non-GCI and GCI cases and the completeness proof differs mainly in the labelled $\exists$-rule. We present results on soundness, completeness and termination below. We introduce some notations used in [BH95]. Let w be a valuation[5]. The unlabelled ABox $w(\mathcal{A})$ is a w-projection of the labelled ABox $\mathcal{A}$, such that if $C(s)^\phi \in \mathcal{A}$ and $w(\phi)$ evaluates to true, then $C(s) \in w(\mathcal{A})$. Details of the notation can be found in [BH95].

Lemma 5.9 (Soundness $\sqcap$-, $\exists$-, $\forall$-rule [BH95])

[5] w is the classical valuation as in the propositional setting.

Let $\mathcal{A}'$ be the labelled ABox generated by applying the $\sqcap$-rule (resp. $\exists$-rule, $\forall$-rule) to a labelled ABox $\mathcal{A}$. Then we either have $w(\mathcal{A}) = w(\mathcal{A}')$, or $w(\mathcal{A}')$ is obtained from $w(\mathcal{A})$ by application of the (unlabelled) $\sqcap$-rule (resp. modified $\exists$-rule, $\forall$-rule).

Lemma 5.10 (Soundness $\sqcup$-rule [BH95])
Let $\mathcal{A}'$ and $\mathcal{A}''$ be the labelled ABox generated by applying the $\sqcup$-rule to a labelled ABox $\mathcal{A}$. Then we either have $w(\mathcal{A}) = w(\mathcal{A}') = w(\mathcal{A}'')$, or $w(\mathcal{A}')$ and $w(\mathcal{A}'')$ are obtained from $w(\mathcal{A})$ by application of the (unlabelled) modified $\sqcup$-rule.

Lemma 5.11 (Soundness $\exists$-rule)
Let $\mathcal{A}'$ be the labelled ABox generated by applying the $\exists$-rule to a labelled ABox $\mathcal{A}$. Then we either have $w(\mathcal{A}) = w(\mathcal{A}')$, or $w(\mathcal{A}')$ is obtained from $w(\mathcal{A})$ by application of the (unlabelled) $\exists$-rule.

Proof. Suppose that the labelled $\exists$-rule is applied to the assertion $\exists R.C(a)$ and its label in $\mathcal{A}_0$ is ϕ. The case where $w(\phi) = false$ is trivial. We consider $w(\phi) = true$. Since $w(\phi) = true$, so $\exists R.C(s)$ is in $w(\mathcal{A}_0)$. Application of the $\exists$-rule to $\exists R.C(a)$ creates a new individual b and adds $R(a, b)$ and $C(b)$ to $\mathcal{A}$ where these assertions have label ϕ. Hence, both $R(a, b)$ and $C(b)$ are in $w(\mathcal{A}_1)$. We can obtain $\mathcal{A}_1$ by application of the modified (unlabelled) $\exists$-rule from $w(\mathcal{A}_0)$. Notice that blocking has to be removed from the modified $\exists$-rule, because otherwise a might be blocked and so we will not be able to obtain $w(\mathcal{A}_1)$ from $w(\mathcal{A}_0)$. This proof is identical to that presented in [BH95], we present this proof to show that the addition of blocking in the $\exists$-rule does not destroy soundness. □

Lemma 5.12 (Completeness)
If none of the rules of the labelled consistency algorithm for $\mathcal{ALC}$ with GCIs are applicable to $\mathcal{A}$ then none of the unlabelled rules for $\mathcal{ALC}$ with GCIs are applicable to $w(\mathcal{A})$.

Proof. Elements of $\mathcal{A}$ are of the form $C(s)$ or $R(s, t)$ where C is a conjunction, disjunction, existential quantifier, universal quantifier or primitive. Therefore it suffices to show for each of the concept types mentioned that if $C(s)$ does not satisfy the precondition(s)

of the labelled rules then $C(s)$ also does not satisfy the precondition(s) of unlabelled $\sqcap$-rule (resp. $\sqcup$-, $\exists$- and $\forall$-rule). The cases where C (or R) is a primitive does not have to be considered because they do not trigger any of the rules.

1. Suppose $(C \sqcap D)(s)$ is in $\mathcal{A}$ and its label ϕ satisfies $w(\phi) = true$. Since this case does not satisfy the labelled $\sqcap$-rule, so $\mathcal{A}$ contains $C(s)$ and $D(s)$, and their labels, ϕ_1 and ϕ_2 respectively, are implied by ϕ. So $w(\psi_1) = w(\psi_2) = true$, thus $C(s)$ and $D(s)$ are in $w(\mathcal{A})$. Therefore, the (unlabelled) $\sqcap$-rule is not applicable.

2. Suppose $(C \sqcup D)(s)$ is in $\mathcal{A}$ and its label ϕ satisfies $w(\phi) = true$. Since this case does not satisfy the labelled $\sqcup$-rule, so $\mathcal{A}$ contains at least one of $C(s)$ or $D(s)$ and its label ψ is implied by ϕ. So $w(\psi) = true$, thus at least one of $C(s)$ or $D(s)$ is in $w(\mathcal{A})$. Therefore, the (unlabelled) $\sqcap$-rule is not applicable.

3. Suppose $\forall R.C(s)$ and $R(s,t)$ are in $\mathcal{A}$ for some individual t, and their labels, ϕ_1 and ϕ_2 respectively, satisfy $w(\phi_1) = w(\phi_2) = true$. Since this case does not satisfy the labelled $\forall$-rule, so $\mathcal{A}$ contains $C(t)$ and its label ψ is implied by $\phi_1 \wedge \phi_2$. So $w(\psi) = true$, thus $C(t)$ is in $w(\mathcal{A})$. Therefore, the (unlabelled) $\forall$-rule is not applicable.

4. Suppose $\exists R.C(s)$ is in $\mathcal{A}$ and its label ϕ satisfies $w(\phi) = true$. Now, assume that $\exists R.C(s)$ is not applicable to $\mathcal{A}$, then it follows from the labelled $\exists$-rule that either s is label blocked or $\mathcal{A}$ contains $R(s,t)$ and $C(t)$ such that their labels, ψ_1 and ψ_2 respectively, are implied by ϕ. There are two cases to consider. In the first case, s is label blocked by some individual s'. Suppose $C_1(s), \ldots, C_n(s)$ are all the concept assertions of s in $\mathcal{A}$ such that their labels, $\phi_1, \ldots, \phi_n$ respectively, satisfy $w(\phi_1) = \cdots = w(\phi_n) = true$. That is, $C_1(s), \ldots, C_n(s)$ are in $w(\mathcal{A})$. By definition of subset label blocking, it follows that the corresponding concept assertions $C_1(s'), \ldots, C_n(s')$ of s', with labels $\psi_1, \ldots, \psi_n$ respectively, satisfy $w(\psi_1) = \cdots = w(\psi_n) = true$. Note that, s' may contain other labelled concept assertions but this doesn't affect the fact that s' is subset label blocked. That is $C_1(s'), \ldots, C_n(s')$ are in $w(\mathcal{A})$. Therefore, for any concept D if $D(s)$ is in $w(\mathcal{A})$ then $D(s')$ is in $w(\mathcal{A})$. So s is (unlabelled subset) blocked by s'. It follows that $\exists R.C(s)$

is not applicable to the unlabelled $\exists$-rule. In the second case, since $w(\phi) = true$, and ϕ imples both ψ_1 and ψ_2, so $w(\psi_1) = w(\psi_2) = true$. Thus, both $R(s,t)$ and $C(t)$ are in $w(\mathcal{A})$.

For each of the cases above, we assume that the label ϕ of $C(s)$ satisfies $w(\phi) = true$. In the cases where $w(\phi) = false$, $C(s)$ is simply not in $w(\mathcal{A})$, so none of the unlabelled rules are applicable to it.

□

Lemma 5.13

The labelled consistency algorithm for $\mathcal{ALC}$ with GCIs (as shown in Figure 5.1) terminates in a finite number of steps.

To show that the algorithm in Figure 5.1 ensures termination. We consider a number of factors. Firstly, each labelled rule can only be applied to the same fact only once. For example, the labelled $\sqcap$-rule can be applied to the same label concept assertion $(C \sqcap D)(s)^{\phi}$ only once and this can happen only if $C_1^{\phi} \,\tilde{\notin}\, \mathcal{A}$ or $C_2^{\phi} \,\tilde{\notin}\, \mathcal{A}$. After applying the labelled $\sqcap$-rule, a new ABox $\mathcal{A}'$ that incorporates both C_1^{ϕ} and C_2^{ϕ} is generated, resulting in $C_1^{\phi} \,\tilde{\in}\, \mathcal{A}$ and $C_2^{\phi} \,\tilde{\in}\, \mathcal{A}$ and falsifying the precondition of the labelled $\sqcap$-rule. Secondly, the description of the concept assertions being generated by applying a rule is shorter than or the same as the description of the concept assertion that triggers the rule. This means concept assertions will eventually break down into atomic concept assertions. Moreover, the $\tilde{\cup}$ operator ensures that only concept assertions with strictly weaker labels are incorporated into the ABoxes.

5.4 More Expressive Description Logics

So far, we have only considered ontology debugging for $\mathcal{ALC}$ with GCIs. In the following section, we consider extensions of the labelled consistency algorithms for more expressive description logics $\mathcal{ALCI}$ and $\mathcal{SI}$. The former extends $\mathcal{ALC}$ with inverse roles and the latter further extends it with transitive roles. It is important to note that (unlabelled) tableau algorithms for $\mathcal{ALCI}$ and $\mathcal{SI}$ with GCIs employ different blocking conditions

$\sqcap$-rule	**if** $(C_1 \sqcap C_2)(x)^\phi \in \mathcal{A}$, x is not indirectly label blocked, and either $C_1(x)^\phi \,\tilde{\notin}\, \mathcal{A}$ or $C_2(x)^\phi \,\tilde{\notin}\, \mathcal{A}$, **then** $\mathcal{A}' := \mathcal{A} \,\tilde{\cup}\, \{C_1(x)^\phi, C_2(x)^\phi\}$.
$\sqcup$-rule	**if** $(C_1 \sqcup C_2)(x)^\phi \in \mathcal{A}$, x is not indirectly label blocked, and both $C_1(x)^\phi \,\tilde{\notin}\, \mathcal{A}$ and $C_2(x)^\phi \,\tilde{\notin}\, \mathcal{A}$, **then** $\mathcal{A}' := \mathcal{A} \,\tilde{\cup}\, \{C_1(x)^\phi\}$, $\mathcal{A}'' := \mathcal{A} \,\tilde{\cup}\, \{C_2(x)^\phi\}$.
$\exists$-rule	**if** $(\exists R.C)(x)^\phi \in \mathcal{A}$, x is not directly label blocked, and there does not exist an individual y such that $R(x,y)^\phi \,\tilde{\in}\, \mathcal{A}$ and $C(y)^\phi \,\tilde{\in}\, \mathcal{A}$, **then** $\mathcal{A}' := \mathcal{A} \,\tilde{\cup}\, \{R(x,y)^\phi, C(y)^\phi\}$, where y is a new individual name and $y > y'$ for all individual names y' in $\mathcal{A}$.
$\forall$-rule	**if** $\{(\forall R.C)(x)^\phi, R(x,y)^\psi\} \subseteq \mathcal{A}$, x is not indirectly label blocked and $C(y)^{\phi \wedge \psi} \,\tilde{\notin}\, \mathcal{A}$, **then** $\mathcal{A}' := \mathcal{A} \,\tilde{\cup}\, \{C(y)^{\phi \wedge \psi}\}$.
$\forall'_+$-rule	**if** $\{(\forall R.C)(x)^\phi, R(x,y)^\psi\} \subseteq \mathcal{A}$, x is not indirectly label blocked, $\texttt{Trans}(\mathrm{R})$ and $(\forall R.C)(y)^{\phi \wedge \psi} \,\tilde{\notin}\, \mathcal{A}$, **then** $\mathcal{A}' := \mathcal{A} \,\tilde{\cup}\, \{(\forall R.C)(y)^{\phi \wedge \psi}\}$.

Figure 5.4: Labelled consistency algorithm for $\mathcal{SI}$ with cyclic definitions

than the one used in $\mathcal{ALC}$. The former requires equivalence blocking and the latter requires pair-wise blocking.

5.4.1 Equivalence Label Blocking and Pair-wise Label Blocking

Definition 5.14 (label equivalence)

Let Φ and Ψ be sets of labelled assertions. Φ is label equivalent to Ψ or $\Phi \,\tilde{=}\, \Psi$ if and only if $\Phi \,\tilde{\subseteq}\, \Psi$ and $\Psi \,\tilde{\subseteq}\, \Phi$. Equivalently, it satisfies:

1. *$\alpha^\phi \in \Phi$ if and only if $\alpha^\psi \in \Psi$ (ϕ and ψ may be different)*
2. *For each assertion α such that $\alpha^\phi \in \Phi$ and $\alpha^\psi \in \Psi$, ϕ is logically equivalent to ψ*

Definition 5.15 (equivalence label blocking)

An individual x equivalent label blocks y if and only if $\mathcal{L}(x) \,\tilde{=}\, \mathcal{L}(y)$ and x is an ancestor of y.

We propose below that the pair-wise label blocking is a revised version of the pair-wise blocking in [HS99]. This is almost identical to pair-wise blocking except that set equivalence is replaced with label equivalence defined in Def. 5.14.

Definition 5.16 (pair-wise label blocking)

An individual is label blocked if and only if it is directly or indirectly label blocked[6]*. A node x is directly label blocked if and only if none of its ancestors are label blocked, and it has ancestors x', y and y' such that:*

1. *x is a successor of x' and y is a successor of y' and*
2. *$\mathcal{L}(x) \tilde{=} \mathcal{L}(y)$ and $\mathcal{L}(x') \tilde{=} \mathcal{L}(y')$ and*
3. *$\mathcal{L}(\langle x', x\rangle) \tilde{=} \mathcal{L}(\langle y', y\rangle)$*

5.4.2 Soundness, Completeness and Termination

Similar to the results we have obtained for $\mathcal{ALC}$, we prove soundness, completeness and termination for $\mathcal{SI}$. The proofs provided are extensions of the proofs for Theorem 5.11.

Lemma 5.17 (Soundness)
Let $\mathcal{A}'$ be the labelled ABox generated by applying the $\forall'_+$-rule to a labelled ABox $\mathcal{A}$. Then we either have $w(\mathcal{A}) = w(\mathcal{A}')$, or $w(\mathcal{A}')$ is obtained from $w(\mathcal{A})$ by application of the (unlabelled) $\forall'_+$-rule.

Proof. Suppose the labelled $\forall'_+$-rule is applicable, then there exists $\{\forall R.C(x)^{\phi_1}, R(x,y)^{\phi_2}\} \subseteq \mathcal{A}$. If $w(\phi_1) = false$ or $w(\phi_2) = false$ (that is, $w(\phi_1 \wedge \phi_2) = false$), then either $\forall R.C(x) \notin w(\mathcal{A})$ or $R(x,y) \notin w(\mathcal{A})$ so the unlabelled $\forall'_+$-rule is not applicable. Hence, we have $w(\mathcal{A}') = w(\mathcal{A})$. Let $w(\phi_1) = true$ and $w(\phi_2) = true$ (i.e. $w(\phi_1 \wedge \phi_2) = true$), so $\forall R.C \in w(\mathcal{A})$ and $R(x,y) \in w(\mathcal{A})$. Therefore the unlabelled $\forall$-rule is applicable.

□

[6] A node is indirectly label blocked if at least one of its ancestors is directly label blocked.

The two theorems below are for equivalent label blocking and pair-wise label blocking.

Lemma 5.18 (Completeness)
If none of the rules of the labelled consistency algorithm for $\mathcal{ALCI}$ with GCIs are applicable to $\mathcal{A}$ then none of the unlabelled rules for $\mathcal{ALCI}$ with GCIs are applicable to $w(\mathcal{A})$.

Proof. To show that $\mathcal{ALCI}$ labelled rules are complete, it suffices to redo the existential case with the stronger equality label blocking (as defined in 5.15). Proofs for the other cases are the same as that of $\mathcal{ALC}$. The proof for the existential case is adapted from the earlier proof for $\mathcal{ALC}$. Suppose $\exists R.C(s)$ is in $\mathcal{A}$ and its label ϕ satisfies $w(\phi) = true$. Now, assume that $\exists R.C(s)$ is not applicable to $\mathcal{A}$, then it follows from the labelled $\exists$-rule that either s is label blocked or $\mathcal{A}$ contains $R(s,t)$ and $C(t)$ such that their labels, ψ_1 and ψ_2 respectively, are implied by ϕ. There are two cases to consider. In the first case, s is label blocked by some individual s'. Suppose $C_1(s), \ldots, C_n(s)$ are all the concept assertions of s in $\mathcal{A}$ such that their labels, $\phi_1, \ldots, \phi_n$ respectively, satisfy $w(\phi_1) = \cdots = w(\phi_n) = true$. That is, $C_1(s), \ldots, C_n(s)$ are in $w(\mathcal{A})$. By definition of equality label blocking, it follows that the corresponding concept assertions $C_1(s'), \ldots, C_n(s')$ of s', with labels $\psi_1, \ldots, \psi_n$ respectively, satisfy $w(\psi_1) = \cdots = w(\psi_n) = true$ and s' has no other labelled concept assertions. That is $C_1(s'), \ldots, C_n(s')$ are in $w(\mathcal{A})$. Therefore, for any concept D, $D(s)$ is in $w(\mathcal{A})$ if and only if $D(s')$ is in $w(\mathcal{A})$. So s is (unlabelled equality) blocked by s'. It follows that $\exists R.C(s)$ is not applicable to the unlabelled $\exists$-rule. In the second case, since $w(\phi) = true$, and ϕ imples both ψ_1 and ψ_2, so $w(\psi_1) = w(\psi_2) = true$. Thus, both $R(s,t)$ and $C(t)$ are in $w(\mathcal{A})$. □

Lemma 5.19 (Completeness)
If none of the rules of the labelled consistency algorithm for $\mathcal{SI}$ with GCIs are applicable to $\mathcal{A}$ then none of the unlabelled rules for $\mathcal{SI}$ with GCIs are applicable to $w(\mathcal{A})$.

Proof. Analagous to the proof in Lemma 5.12, we show that the labelled consistency algorithm is complete for $\mathcal{SI}$ by proving that if an assertion $C(s)$ does not satisfy the precondition(s) of an labelled rule then the corresponding assertion will also not satisfy

the unlabelled rule. Again, we leave out the cases where the assertions are primitives. We begin by considering a concept assertion of the form $(C \sqcap D)(s)$ with its label ϕ satisfying $w(\phi) = true$. Since the precondition of the labelled $\sqcap$-rule is not satisfied so either: (1) s is indirectly label blocked; or (2) $C_1(x)^{\phi} \tilde{\in} \mathcal{A}$ and $C_2(x)^{\phi} \tilde{\in} \mathcal{A}$. For (2), this case is identical to that of Lemma 5.12 and we have $w(\phi_1) = w(\phi_2) = true$, and so $C(s)$ and $D(s)$ are in $w(\mathcal{A})$. For (1), s is indirectly label blocked. By Definition 5.16, it implies that there is an ancestor s' such that s' is directly label blocked. Hence, by Definition 5.16, s' has ancestors s'', t' and t'' where s' is a successor of s'' and t' and is a successor of t''. It is then easy to see that s' is (directly) pair-wise blocked by s''. This means the unlabelled $\exists$-rule is not applicable to $w(\mathcal{A})$. The other cases including $\sqcup$-rule, $\exists$-rule, $\forall$-rule and $\forall'_{+}$-rule can be shown in a similar manner. □

Proposition 5.20 (Termination) *The labelled consistency algorithm (in Figure 5.1) for $\mathcal{ALCI}$ with GCIs using equality label blocking terminates in finite number of steps.*

Proof. (sketch) This proof is very similar to that of Proposition 5.13 since both $\mathcal{ALCI}$ and $\mathcal{ALC}$ employ the same set of labelled consistency rules in Figure 5.1. The only difference is that $\mathcal{ALCI}$ adopts the equality label blocking, whereas $\mathcal{ALC}$ adopts the subset label blocking. Therefore, it suffices it show that the use of equality label blocking does not destroy termination. The labelled algorithm will terminate in finite number of steps if it does not generate an infinitely long chain. We know that it is not possible to generate an infinitely long chain because: (1) For the same concept assertion with the same label, it can be applied to each labelled rule only once. In fact, for the same concept assertion only the ones with a weaker label (than previous rule application) will trigger a rule. (2) A node will eventually be blocked because the number of propositional variables or labels is fixed (same size as the number of axioms). □

Proposition 5.21 (Termination) *The labelled consistency algorithm (in Figure 5.4) for $\mathcal{SI}$ with GCIs using pair-wise label blocking terminates in finite number of steps.*

Proof. (sketch) Again, this is similar to the earlier proofs on termination for $\mathcal{ALC}$ and $\mathcal{ALCI}$ with the exception that $\mathcal{SI}$ has a slightly different set of labelled rules as well as a different label blocking condition. The key of this proof is to show that there is no path of infinite length. From Horrocks et al.'s paper [HS99], we know that the classical (unlabelled) rules can create a path with at most 2^{mn} nodes in length where m is the number of concept and n is the number of roles. Since our labelled algorithm attaches a propositional label to each of the concept and role assertions, hence for each concept assertion we can attach 2^p labels to it, where p is the number of propositional variables (or number of axioms). Therefore, the maximum length of a path is 2^{mnp}. A path with this length will necessarily have a pair of individuals that are pair-wise label blocked. □

5.5 Binary Decision Diagram (BDD)

The labelled consistency algorithm constructs trace information and allows such information to be used in ontology debugging. As we have seen in the previous section, one way to make use of this trace information is to compute the MSSes and MUSes. However, there are at least two major drawbacks to this approach. Firstly, finding prime implicates (or prime implicants) of a propositional formula is an NP-hard problem. That means finding them is time-consuming, especially when the original formula is complicated and involves a large number of variables. Secondly, the number of prime implicates (or prime implicants) can be exponential to the length of the original formula, which means that there can potentially be a large number of MSSes or MUSes presenting to the user and this is certainly not desirable. In [KPSCG06], the authors proposed a way to rank the variables appearing in the MUSes based on the frequency of each variable appearing in the MUSes. A variable with higher frequency is given a higher rank over a variable with lower frequency. The rationale behind this is that a repair corresponds to picking at least one variable from each of the MUSes, hence a variable with higher frequency is one that makes more of the MUSes satisfiable. However, this solution still requires finding the MUSes first and so it still suffers from the first drawback mentioned above. It should be

noted that multisets of indices were used in [Sch05b] to store trace information instead of propositional formulas, but finding MUSes for them is still an NP-hard problem.

As we have mentioned in earlier sections, adopting propositional formulas as labels in the labelled consistency algorithm has a number of advantages. There are efficient algorithms to check for satisfiability and entailment of propositional formulas, and there are efficient data structures that represent propositional formulas (or functions). A data structure that is of particular interest to us is the Binary Decision Diagram (BDD) [Bry86, Bry92], and we propose to use BDD as a means for ontology debugging. BDD is most well known for its application in fault tree analysis and in the area of formal verification. The main advantage of using BDD for ontology debugging is that it presents a compact representation for indicating logical errors and repairs of an ontology, thus it is a user friendly approach for ontology debugging especially when the ontology is large and complex.

A binary decision diagram is a directed acyclic graph that has a root. It has two terminal nodes with out-degree 0 that are labelled with 0 or 1, and it contains non-terminal nodes with out-degree 2. In the literature, outgoing edges of a non-terminal node are commonly referred to as $low(n)$ and $high(n)$, and the propositional variable associated with the node as $var(n)$. In what follows, we study a special type of BDD known as Reduced Ordered Binary Decision Diagram (ROBDD), which we propose to use in ontology debugging. A ROBDD assumes an ordering on the variables over the formula. However, computing the best ordering is an NP-Complete problem [BW96]. An important property of ROBDD is that formulas that are logically equivalent would result in the same ROBDD. This is a useful property because it indicates that ROBDD is a representation that captures the semantics of a formula and not just the syntax. Note that, a (normal) BDD does not have this property.

5.5.1 Constructing ROBDDs from Propositional Formulas

There are two established approaches to construct an ROBDD [Bry86, Bry92] from a propositional formula. One approach is to build a Binary Decision Tree (BDT) from a propositional formula and then apply reduction rules exhaustively to transform the

BDT into a BDD. The other approach [BRB90] is to build the ROBDD directly from the propositional formula. The best known algorithm is a bottom-up construction that applies a number of operations recursively to sub-BDDs of a formula.

We describe below an example that makes use of the first approach to construct an ROBDD for the propositional formula in Example 5.8. We begin by building a binary decision tree using the well-known Shannon expansions.

Example 5.22

In Example 5.8, we have a propositional (clash) formula $(p\wedge(p\wedge q))\vee(p\wedge((p\wedge q)\vee(p\wedge r)))$ $\vee(p\wedge((p\wedge q)\vee(p\wedge r)\vee(p\wedge s)))$. This formula consists of 4 variables $\{p,q,r,s\}$ and we assume an ordering on these variables: $p<q<r<s$. The ordering is not necessary for constructing a BDT but it is necessary to construct an *Ordered* Binary Decision Diagram. The notation $p<q$ denotes that the binary variable p is assigned with values (0 or 1) before an assignment is made to q. By selecting variables based on the defined ordering, we derive the following expressions using Shannon expansions:

$$t = p \rightarrow t_1, 0$$

$$t_1 = q \rightarrow 1, t_{10}$$

$$t_{10} = r \rightarrow 1, t_{100}$$

$$t_{100} = s \rightarrow 1, 0$$

These expressions can be diagrammatically shown as a binary decision tree and we can further develop it into a binary decision diagram by merging nodes (both branch nodes and terminal nodes) that are identical. To do that, we create two terminal nodes for the values $0, 1$ (typically shown as boxes with the corresponding values written on them). For each expression above, in a bottom-up manner (i.e., starting with the last expression), we create a node for each variable and draw a directed arrow (broken arrow for assigning a value of 0 and solid arrow for 1) from this node to its successor nodes (i.e., those that corresponds to the right-hand side of the $\rightarrow$ in each expression above).

We show below a construction of ROBDD based on the expressions above:

An interesting point to note is that the formula $(p\wedge(p\wedge q))\vee(p\wedge((p\wedge q)\vee(p\wedge r)))$

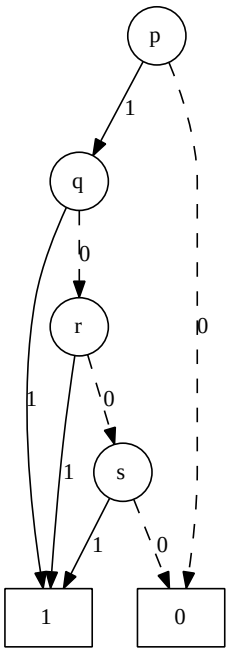

Figure 5.5: Binary Decision Diagram for $(p \wedge q) \vee (p \wedge r) \vee (p \wedge s)$ with $p < q < r < s$

$\vee(p \wedge ((p \wedge q) \vee (p \wedge r) \vee (p \wedge s)))$ and $(p \wedge q) \vee (p \wedge r) \vee (p \wedge s)$ are equivalent to each other, and they produce ROBDDs that are isomorphic to each other.

5.5.2 Debugging with ROBDD

Here we describe how ROBDD can be used for ontology debugging. Any path from the root node to terminal node 1 is a repair to the ontology. Paths starting at the root node and leading to the terminal node 0 corresponds to a non-repair and can be omitted. In fact, nodes that do not have any paths that leads to terminal node 1 can be safely disregarded. Similarly, any edges that do not lie on paths leading to terminal node 1 can also be removed.

By following the assignments along such a path we can restore the consistency of the ontology. Hence the task of debugging an ontology becomes simply a navigation through the paths and selecting the one that suits the user's need.

Figure 5.5 describes the propositional formula $(p \wedge q) \vee (p \wedge r) \vee (p \wedge s)$ displayed as a ROBDD. The diagram is rooted at p (the least variable in the ordering), has two

terminal nodes 0 and 1, and four non-terminal nodes p,q,r and s. It has two paths leading to terminal 0 and three paths leading to terminal 1. This indicates that there are two ways to make assignments to the variables such that the formula evaluates to false, and three ways evaluates to true. Note that, all paths in the diagram adheres to the variable ordering $p < q < r < s$. Now, suppose $(p \wedge (p \wedge q)) \vee (p \wedge ((p \wedge q) \vee (p \wedge r)))$ $\vee(p\wedge((p\wedge q)\vee(p\wedge r)\vee(p\wedge s)))$ is the clash formula as shown in Example 5.8. It may not be immediately clear, but this formula is logically equivalent to $(p\wedge q)\vee(p\wedge r)\vee(p\wedge s)$. That is, the ROBDD of the clash formula is isomorphic to the one in Figure 5.5. Now, finding a repair of the ontology amounts to finding a valuation that evaluates the formula to false. This corresponds to selecting (at least) a path from the root node to the terminal node 0. As shown clearly in the diagram, there are two alternatives. One is to remove p, the other is to keep p and remove p, q and r.

Both of these solutions turn out to be prime implicants, but they do not necessarily have to be so. However, it is possible to generate the prime implicants (hence prime implicates) for a propositional formula given its ROBDD through a series of reductions.

Though computing an ROBDD is computationally as expensive as finding prime implicates and implicants, ROBDD presents a much more user-friendly interface for debugging ontologies. Moreover, the presentation of an ROBDD is usually much more compact than that of prime implicates and implicants that often contain a large amount of redundant information.

5.6 Summary and Discussion

In this chapter, we studied the problem of ontology debugging by presenting algorithms with desirable properties that guide ontology engineers to construct and repair ontologies. We presented extensions to a well known tableau-based algorithm known as labelled consistency algorithm that builds traces as propositional formulas to obtain the necessary information to resolve logical inconsistencies by computing the Maximally Satisfiable Subsets (MSSes) (and Minimally Unsatisfiable Subsets (MUSes) of terminologies). In particular, we extended the labelled consistency algorithm for $\mathcal{ALC}$ to handle General

Inclusion Axioms (GCIs). We showed that classical blocking is not sufficient to guarantee completeness in the presence of cyclic definitions and we proposed a refined blocking condition called subset label blocking for circumventing this problem. In addition, we proved that the labelled consistency algorithm for GCIs preserve a number of desirable properties, including soundness, completeness and termination. Furthermore, we presented the labelled consistency algorithm for both $\mathcal{ALCI}$ and $\mathcal{SI}$ with GCIs, and we showed that these algorithms also ensures soundness, completeness and termination. Finally, we argued that building traces as propositional formulas has a number of advantages over building traces as multisets of indices. We described how these propositional formulas can be represented as Reduced Ordered Binary Decision Diagrams (ROBDDs) and that ROBDDs are suitable diagrammatic tools for repairing ontologies. In the following chapter, we will present implementation details and some experimental results of the labelled consistency algorithms described in this chapter.

Chapter 6

Conclusion

In this final chapter, we provide a summary of the research results we have presented in the earlier chapters.

6.1 Ontology Contraction

In Chapter 3 we considered the problem of ontology contraction in a setting where there is a finite set of description logic sentences A and we are to contract a DL sentence α from it in a logically consistent manner. This problem is similar to the problem of belief contraction in the AGM framework, but AGM theory is defined under propositional assumptions for belief sets, whereas we are considering the problem in description logic and adopting the belief base approach. It is well-known [MLB05, FHP$^+$06] that DL lacks the expressive power to apply some of the AGM theory directly. In particular, the well known Levi identity cannot be used without negation being defined for all sentences in the language, while negation of axioms cannot be expressed in DL. We showed that classical remainder set, as used in the belief contraction framework to construct partial meet construction, has limitations when applied in the description logic setting.

Though we showed through example that partial meet contraction can be directly used by treating DL sentences as propositional sentences (hence satisfying the partial meet contraction postulates for belief bases), we also demonstrated that this would lead to counter-intuitive results.

6.1.1 Remainder Set for Description Logics

Realising this limitation, we introduced the notion of exceptions first published in our work [MLB05] and later extended by Qi et al. [QLB06]. We presented a construction of the remainder set for DL $A\perp_{dl}\alpha$ using the notion of exceptions and we showed that it is a refinement of the classical remainder set $A\perp\alpha$, in the sense that our construction would lead to more refined solutions than that of the classical remainder set when being applied to DL sentences. We argued that this is a desirable result because it is consistent with the Principle of Minimal Change, which is widely adopted in the Belief Change community.

6.1.2 Revised Partial Meet Contraction Postulates for Ontology Bases

In addition, we extended the construction of the remainder set of DL to partial meet contraction for belief bases similar to that of classical remainder set, and we observed that our contraction operator no longer satisfies the classical partial meet contraction postulates for belief bases. We realised that this is due to the fact that we are no longer building remainder set based on subsets, so we presented weakened versions of the contraction postulates. More specifically, we revised the inclusion and relevance postulates and we showed that our construction satisfies all of the revised postulates.

(Success) If $\alpha \notin Cn(\emptyset)$, then $\alpha \notin Cn(A \div \alpha)$

(Inclusion') If $\beta' \in A \div \alpha$ then $\beta' \in Cn(\{\beta\})$ for some $\beta \in A$

(Relevance') If $\beta \in A$ and $\beta \notin A \div \alpha$, then there is a set A' such that $A \div \alpha \sqsubseteq A' \sqsubseteq A$ and that $\alpha \notin Cn(A')$ but $\alpha \in Cn(A' \cup \{\beta\})$

(Uniformity) If it holds for all subsets A' of A that $\alpha \in Cn(A')$ if and only if $\beta \in Cn(A')$, then $A \div \alpha = A \div \beta$.

(Success') ensures that the contraction operation does in fact have the sentence removed, except in the case where the sentence a tautology. **(Inclusion')** captures the

notion that every weakened sentence must originate from the original set of sentences. It is a weakened version of the classical inclusion postulate in the sense that every contraction operator that satisfies the classical inclusion postulate must also satisfy **(Relevance')** ensures that every weakened sentence must have a good reason, i.e., a sentence is not to be weakened if it is not necessary to ensure consistency. **(Uniformity')** imposes a view that sentences which are the same from the perspective of A are treated equally and produce the same results.

Moreover, we introduced the notion of multiple contraction for DL. We presented both the notion of partial meet package contraction and partial meet choice contraction, as well as a revised set of multiple contraction postulates for each of them.

6.2 Ontology Integration

In Chapter 4, we extended our framework to knowledge integration where we considered integration of stratified DL knowledge bases. Our work was motivated by Benferhat et al. [BKLBW04] where they presented the notion of adjustment in the propositional setting. In the propositional setting, the problem of knowledge integration concerns with the integration of a stratified knowledge base where each stratum is a set of propositional sentences.

6.2.1 Conjunctive Max-Adjustment (CMA)

We introduced the notion Conjunctive Maxi-Adjustment (CMA) in the propositional setting, similar to (whole) Disjunctive Maxi-Adjustment (DMA) in [BKLBW04]. The idea is to traverse through the stratified knowledge base one sentence by one sentence to accumulate a set of consistent sentences, from the stratum with the highest preference to the stratum with the lowest preference. At each stratum, we collect sentences that do not conflict with the accumulated set of sentences. If a sentence conflicts then we weaken it until it is consistent and then collects it into the accumulated set.

In addition, we showed that the lexicographic entailment relation constructed from the lexicographic ordering of the set of propositional interpretations corresponds exactly

to the CMA construction.

$$K \models_{lex} \phi \text{ if and only if } \delta_{cma} \models \phi$$

We recasted the notion of adjustment into the DL setting. For this reason, we introduced the notion of disjunctive knowledge bases that allows disjunction of DL sentences to be expressed adequately. We did this because disjunction of sentences generally cannot be expressed in description logics. In particular, disjunctions of combinations of concept assertions and role assertions are not syntactically valid DL sentences and therefore they have no proper semantics. We introduced this notion so that we could express the results of adjustment adequately in DL.

6.2.2 Knowledge Integration in Description Logics

We addressed the problem of knowledge integration in the DL setting where DL sentences are presented as a stratified knowledge base, with each stratum being a multiset of DL sentences. We made use of the notion of disjunctive knowledge base to express the result of integrating a stratified knowledge base. We presented the construction for CMA-DL based on the idea of CMA but adapted into DL. We made use of the notion of exceptions again to construct weakened DL sentences as we did in Chapter 3. In addition, we introduced the semantics of exceptions by defining the number of exceptions of an interpretation $\mathcal{I}$ for a TBox sentence of the form $C \sqsubseteq D$.

$$e^{\mathcal{I}}(C \sqsubseteq D) = |C^{\mathcal{I}} \cap \neg D^{\mathcal{I}}|$$

Similarly, we extended this notion to sets of TBox sentences and then to sets of ABox sentences. These notions were refined by [QLB06] based on our work in [MLB05]. We presented the notion of lexicographic ordering of DL interpretations using the notion of exceptions and based on the minimal models of the lexicographic ordering we introduced lexicographic entailment for DL. Moreover, we proved that CMA-DL corresponds exactly

to lexicographic entailment for DL and we obtained the following property:

$$K \models_{lex} \phi \text{ if and only if } \delta_{cma-dl} \models \phi$$

We then showed on examples that there are limitations to CMA-DL, in particular it does not exploit the structure of DL sentences properly. In response to this, we made use of a construct (invented in [BBH96]) called cardinality restriction on concepts, to produce a refinement of CMA-DL called RCMA-DL. Similarly, we established the correspondence of RCMA-DL with lexicographic entailment:

$$K \models_{lex*} \phi \text{ if and only if } \delta_{rcma-dl} \models \phi$$

It is important to note that there is a close connection between nominals and cardinality restriction on concepts. It is possible to express one in terms of the other and vice versa.

6.3 Ontology Debugging

In Chapter 5 we investigated on a tableau-based algorithm for computing Maximally Satisfiable Subsets (MSSes) of a TBox. It should be noted that there is a close relationship between MSSes and Minimally Unsatisfiable Subsets (MUSes) and it has been showed in the literature [Sch05b] that one could convert from one to another using the hitting set algorithm. This problem was first explored in the DL context by Baader et al. [BH95] and later in Schlobach et al. [Sch05b]. There is also a large body of work in this area using various strategies, including automata based methods and structural subsumption. Baader et al. used the name labelled consistency algorithm in their work [BH95], so we followed this convention.

6.3.1 Labelled Consistency Algorithm for Cyclic Definitions

We discovered a problem with the labelled consistency algorithm in the presence of cyclic definitions. In particular, we showed that the labelled consistency algorithm with

classical subset blocking will not guarantee completeness. However, the procedure will still terminate and the results it produces will still be sound. We produced examples that demonstrated this behaviour. In addition, we showed that a more refined blocking is needed for the labelled consistency algorithm in the presence of cyclic definitions. We named this blocking condition labelled subset blocking and we showed that it is suitable for description logics that use subset blocking.

Let Φ and Ψ be sets of labelled assertions. $\Phi \tilde{\subseteq} \Psi$ if and only if for each labelled assertion $\tilde{\alpha} \in \Phi$ implies $\tilde{\alpha} \tilde{\in} \Psi$. Equivalently, $\Phi \tilde{\subseteq} \Psi$ if and only if it satisfies:

1. *if $\alpha^{\phi} \in \Phi$ then $\alpha^{\psi} \in \Psi$ (ϕ and ψ may be different)*

2. *For each assertion α such that $\alpha^{\phi} \in \Phi$ and $\alpha^{\psi} \in \Psi$, ϕ implies ψ*

We extended the proof by Baader et al. [BH95] and showed that under the labelled subset blocking, the labelled consistency algorithm is both sound and complete.

6.3.2 Reduced Ordered Binary Decision Diagram (ROBDD)

We argued that the labelled consistency algorithm has an advantage over other existing approaches. In particular, the labelled consistency algorithms keeps track of the trace as a propositional formula. Propositional formula is *not* just a more compact representation to keep track of the trace but also has the advantage of being able to convert to a Reduced Ordered Binary Decision Diagram (ROBDD) that can be used directly as a diagrammatic tool to debug an ontology. We showed that a repair to an ontology corresponds to choosing one path in the ROBDD. In addition, we argued that conversion to ROBDD is desirable because there are efficient algorithms to convert a propositional formula to ROBDD and from ROBDD to other compact and useful representations. For example, the problem of ranking axioms mentioned in [KPSCG06] relates closely to optimising ordering of variables in ROBDDs [BW96].

6.4 Future work

6.4.1 On the Relation of Partial Meet Contraction and Kernel Contraction for DL

We presented in Chapter 3 the notion of remainder set for DL, which takes advantage of the structure of DL sentences and produces more refined solutions than that of classical remainder set. We showed also that the notion of remainder set for DL allows us to produce the partial meet contraction for DL that satisfies a set of refined contraction postulates. In classical belief contraction, it is possible to establish a relation between partial meet contraction and kernel contraction. In fact, it has been shown that kernel contraction is more general than partial meet contraction [RW08]. It has also been shown that remainder sets can be obtained from incision functions by finding the minimum hitting sets [Rei87, FFI06]. However, we argued also in Chapter 3 that it is difficult to apply the same technique to produce kernel contraction for DL. In particular, it is unclear how one could perform set subtraction on a set $A \setminus \sigma(B \perp\!\!\!\perp \alpha)$, where $\sigma(B \perp\!\!\!\perp \alpha)$ is a weakened set of DL sentences. We believe it is possible to make use of the work on concept description subtraction [Tee94]. However, this work only talks about concept description subtraction, whereas we need subtraction of DL sentences in our case. We will explore this issue in our future work.

6.4.2 Performance Evaluation of the Labelled Consistency Algorithm

We studied in Chapter 5 the labelled consistency algorithm, which was first introduced by [BH95] and we defined new notion of blocking that ensures completeness of the algorithm in the presence of cyclic definition. A major criticism of this work, however, is that the labelled consistency requires that the labelled algorithm is completely saturated, and therefore it is believed to be computationally expensive. It was also argued in [Kal06] that computing a single solution and then using the hitting set algorithm can be more efficient. We believe it is possible to obtain results that are comparable to existing approaches. We argue that most experimental results on ontology debugging were obtained from ontologies with artificially created inconsistencies, and these ontologies contain relatively

small number of inconsistencies compared to the size of the whole ontology. We believe that this is not sufficient to show the actual performance of the existing approaches. We argue that the labelled consistency algorithm could produce equally good results.

In response to this, we have implemented a prototype for the labelled consistency algorithm described in Chapter 5 and have run our implementation against a number of well-known ontologies including: (1) the Camera ontology with 12 axioms and 14 concepts; (2) the Koala ontology with 29 concepts and 19 axioms; and (3) a simplified version of the DICE terminology with 527 concepts and 536 axioms, and with the 21 disjointness axioms disabled.

In each case, we execute our labelled consistency algorithm to find the A-MSSes for every concept name occurring in the ontology and our algorithm would return a propositional formula for each input concept indicating what is needed to make the concept satisfiable. If a concept is already satisfiable then our algorithm would return $\top$, meaning that every valuation would result in a satisfiable concept. In other words, no repair is required. On the other hand, if a concept is not satisfiable then each evaluation of the output formula corresponds to a repair of the intput concept. For example, an output of $\neg p \wedge \neg q$ means that we need to remove both sentences tagged with the propositional variables p and q.

We conducted our experiments on a standard Linux (Debian) machine with a 2.53GHz Intel Pentium 4 processor, 512KB cache and 512MB of physical memory. Our implementation was developed in Java (JDK 1.5.0) without using any of the existing reasoners. That is, we developed our own tableau algorithm, as well as the modification needed to turn it into a labelled algorithm. However, this also means that we did not inherit any implementation of optimisations from existing reasoners. Instead, we implemented our own ordering heuristics which allows any ordering of expansion rules to be specified, and such optimisation can have a significant impact on performance. In our evaluation, we performed all experiments using the following fixed ordering of expansion: (1) $\sqcap$-rule (2) D^+- and D^--rules; (3) $\sqcup$-rule; (4) $\exists$-rule; (5) $\bot$-rule.

In comparison to using FACT++ as a black box for finding the A-MSSes of DICE, it takes 527 satisfiability checks to determine that 109 concepts are satisfiable, then

Ontology Name	Parsing	Finding A-MSSes	No. of Unsatisfiable Concepts
Camera	83ms	83ms	3/14
Koala	86ms	131ms	3/29
DICE	282ms	7017ms	109/527

Figure 6.1: Experimental Results of Finding A-MSSes with the Labelled Consistency Algorithm

additional 109×515 checks to determine all A-MSSes for all concepts in which exactly one axiom is excluded, and then $2 \times (515 \times 514/2)$ to find the A-MSSes in which two axioms are excluded. In total, this required $321,372$ satisfiability checks. The algorithm takes about 0.2ms for FACT++, implemented on the same machine as our algorithm, to perform a satisfiability check for one of the DICE concepts. This means it takes FACT++ 64.274 seconds to find all A-MSSes, but without a guarantee that all A-MSSes have been found. At this stage it is too early to draw any meaningful conclusions, but the results obtained so far merit a more detailed investigation.

Our next step is to create artificial ontologies with a large number of inconsistencies, which would allow us to make comparisons with existing approaches.

Bibliography

[AGM85] Carlos E. Alchourrón, Peter Gärdenfors, and David Makinson. On the logic of theory change: Partial meet contraction and revision functions. *Symbolic Logic*, 50(2):510–530, 1985.

[AM81] Carlos E. Alchourrón and David Makinson. Hierarchies of regulation and their logic. *New studies in Deontic Logic: norms, actions, and the foundations of ethics*, pages 125–148, 1981.

[BBB+09] Franz Baader, Andreas Bauer, Peter Baumgartner, Anne Cregan, Alfredo Gabaldon, Krystian Ji, Kevin Lee, David Rajaratnam, and Rolf Schwitter. A novel architecture for situation awareness systems. In *Proceedings of the 18th International Conference on Automated Reasoning with Analytic Tableaux and Related Methods*, pages 77–92. Springer, 2009.

[BBH96] Franz Baader, Martin Buchheit, and Bernhard Hollunder. Cardinality restrictions on concepts. *Artificial Intelligence*, 88(1-2):195–213, December 1996.

[BBL05] Franz Baader, Sebastian Brandt, and Carsten Lutz. Pushing the $\mathcal{EL}$ envelope. In *Proceedings of the* 19^{th} *International Joint Conference on Artificial Intelligence*, pages 364–369. Morgan Kaufmann, 2005.

[BCD+93] Salem Benferhat, Claudette Cayrol, Didier Dubois, Jerome Lang, and Henri Prade. Inconsistency management and prioritized syntax-based entailment. In *Proceedings of the* 13^{th} *International Joint Conference on Artifical Intelligence*, pages 640–645. Morgan Kaufmann., 1993.

[BCM+03] Franz Baader, Diego Calvanese, Deborah L. McGuinness, Daniele Nardi, and Peter F. Patel-Schneider, editors. *The description logic handbook: theory, implementation, and applications*. Cambridge University Press, 2003.

[BH95] Franz Baader and Bernhard Hollunder. Embedding defaults into terminological knowledge representation formalisms. In *Journal of Automated Reasoning*, volume 14, pages 306–317. Morgan Kaufmann, 1995.

[BHGS01] Sean Bechhofer, Ian Horrocks, Carole A. Goble, and Robert Stevens. Oiled: A reason-able ontology editor for the semantic web. In *Proceedings of the Joint German/Austrian Conference on AI: Advances in Artificial Intelligence*, pages 396–408. Springer, 2001.

[BHJV08] Jürgen Bock, Peter Haase, Qiu Ji, and Raphael Volz. Benchmarking OWL Reasoners. In *Proceedings of the ARea2008 Workshop*, volume 350. CEUR, June 2008.

[BKLBW04] Salem Benferhat, Souhila Kaci, Daniel Le Berre, and Mary-Anne Williams. Weakening conflicting information for iterated revision and knowledge integration. *Artificial Intelligence*, 153(1-2):339–371, March 2004.

[BKW03] Franz Baader, Ralf Küsters, and Frank Wolter. Extensions to description logics. In Franz Baader, Diego Calvanese, Deborah L. McGuinness, Daniele Nardi, and Peter F. Patel-Schneider, editors, *The description logic handbook*, pages 219–261. Cambridge University Press, 2003.

[BL98] Tim Berners-Lee. Semantic web road map. *Design Issues for the World Wide Web*, pages 1–5, September 1998.

[BLHL01] Tim Berners-Lee, James Hendler, and Ora Lassila. The semantic web. *Scientific American*, 284(5):34–43, May 2001.

[BLS06] Franz Baader, Carsten Lutz, and Boontawee Suntisrivaraporn. Efficient reasoning in $\mathcal{EL}^+$. In *Proceedings of the* 27^{th} *International Workshop on Description Logics*, volume 189. CEUR, 2006.

[BPn10] Franz Baader and Rafael Peñaloza. Automata-based axiom pinpointing. *Journal of Automated Reasoning*, 45(2):91–129, August 2010.

[BRB90] Karl S. Brace, Richard L. Rudell, and Randal E. Bryant. Efficient implementation of a BDD package. In *Proceedings of the 27th ACM/IEEE Design Automation Conference*, pages 40–45. ACM, 1990.

[Bry86] Randal E. Bryant. Graph-based algorithms for boolean function manipulation. *IEEE Transactions on Computers*, 35(8):677–691, August 1986.

[Bry92] Randal E. Bryant. Symbolic boolean manipulation with ordered binary-decision diagrams. *ACM Computing Surveys*, 24(3):293–318, September 1992.

[BS08] Franz Baader and Boontawee Suntisrivaraporn. Debugging SNOMED CT using axiom pinpointing in the description logic $\mathcal{EL}^+$. In *KR-MED*, volume 410 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2008.

[BvHH+04] Sean Bechhofer, Frank van Harmelen, Jim Hendler, Ian Horrocks, Deborah L. McGuinness, Peter F. Patel-Schneider, and Lynn Andrea Stein. OWL web ontology language reference. Technical report, W3C, February 2004.

[BW96] Beate Bollig and Ingo Wegener. Improving the variable ordering of OBDDs is NP-complete. *IEEE Transactions on Computers*, 45(9):993–1002, 1996.

[FFI06] Marcelo A. Falappa, Eduardo L. Fermé, and Gabriele K. Isberner. On the logic of theory change: Relations between incision and selection functions. In *Proceeding of the* 17^{th} *European Conference on Artificial Intelligence*, pages 402–406. IOS Press, 2006.

[FH94] André Fuhrmann and Sven O. Hansson. A survey of multiple contractions. *Journal of Logic, Language and Information*, 3:39–76, 1994.

[FHP+06] Giorgos Flouris, Zhisheng Huang, Jeff Z. Pan, Dimitris Plexousakis, and Holger Wache. Inconsistencies, negations and changes in ontologies. In

Proceedings of the 21^{st} *National Conference on Artificial intelligence*, pages 1295–1300. AAAI Press, 2006.

[FPA04] Giorgos Flouris, Dimitris Plexousakis, and Grigoris Antoniou. Generalizing the agm postulates: Preliminary results and applications. In *Proceedings of the* 10^{th} *International Workshop on Non-Monotonic Reasoning*, pages 171–179, 2004.

[FPA05a] Giorgos Flouris, Dimitris Plexousakis, and Grigoris Antoniou. On applying the AGM theory to DLs and OWL. In *Proceedings of the* 4^{th} *International Semantic Web Conference*, pages 216–231. Springer, 2005.

[FPA05b] Giorgos Flouris, Dimitris Plexousakis, and Grigoris Antoniou. Updating description logic using the AGM theory. In *Proceedings of the* 7^{th} *International Symposium on Logical Formalizations of Commonsense Reasoning*, pages 69–76, 2005.

[FSS03] Eduardo L. Fermé, Karina Saez, and Pablo Sanz. Multiple kernel contraction. *Studia Logica: An International Journal for Symbolic Logic*, 73(2):183–195, 2003.

[FT88] U. Faigle and G. Turán. Sorting and recognition problems for ordered sets. *SIAM Journal on Computing*, 17(1):100–113, 1988.

[Gar88] Peter Gardenfors. *Knowledge in Flux: Modeling the Dynamics of Epistemic States*. MIT Press, 1988.

[GHM+08] 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, November 2008.

[GLRV04] Diego C. Giuseppe, Maurizio Lenzerini, Riccardo Rosati, and Guido Vetere. DL-Lite: Practical reasoning for rich DLs. In *Proceedings of the* 25^{th} *International Workshop on Description Logics*, volume 104, 2004.

[Gru93] Thomas R. Gruber. A translation approach to portable ontology specifications. *Knowledge Acquisition*, 5(2):199–220, June 1993.

[Gru95] Thomas R. Gruber. Toward principles for the design of ontologies used for knowledge sharing. *International Journal Humuman-Computer Studies*, 43(5-6):907–928, December 1995.

[GSK00] Dov M. Gabbay, Philippe Smets, and Rudolf Kruse. *Handbook of Defeasible Reasoning and Uncertainty Management Systems: Volume 4: Abductive Reasoning and Learning*. Handbook of Defeasible Reasoning and Uncertainty Management Systems. Kluwer, 2000.

[Han94] Sven O. Hansson. Kernel contraction. *Journal of Symbolic Logic*, 59(3):845–859, 1994.

[Han99] Sven O. Hansson. *A Textbook of Belief Dynamics: Theory Change and Database Updating*. Applied Logic Series. Kluwer Academic Publishers, 1999.

[Har75] William L. Harper. Rational conceptual change. In *Proceedings of the Meeting of the Philosophy of Science Association*, volume 2, pages 462–494. Philosophy of Science Association, 1975.

[HBPS08] Matthew Horridge, Johannes Bauer, Bijan Parsia, and Ulrike Sattler. Understanding entailments in OWL. In *Proceedings of the* 9^{th} *International Workshop in OWL: Experience and Directions*, volume 432. CEUR-WS, 2008.

[HKS06] Ian Horrocks, Oliver Kutz, and Ulrike Sattler. The even more irresistible $\mathcal{SROIQ}$. In *Proceedings of the* 10^{th} *International Conference on Principles of Knowledge Representation and Reasoning*, pages 57–67. AAAI Press, June 2006.

[HM01a] Volker Haarslev and Ralf Möller. Description of the racer system and its applications. In *Proceedings of the 2001 International Workshop on Description Logics (DL-2001)*, pages 132–141. CEUR-WS, 2001.

[HM01b] Volker Haarslev and Ralf Möller. Racer system description. In *IJCAR '01: Proceedings of the First International Joint Conference on Automated Reasoning*, pages 701–706, London, UK, 2001. Springer.

[HMW03] Volker Haarslev, Ralf Moller, and Michael Wessel. Racer: Renamed ABox and concept expression reasoner. available: http://www.sts.tu-harburg.de/~r.f.moeller/racer/, 2003.

[Hor98] Ian Horrocks. The $fact$ system. In *Proceedings of the International Conference on Automated Reasoning with Analytic Tableaux and Related Methods*, pages 307–312, London, UK, 1998. Springer.

[HPS03] Ian Horrocks and Peter F. Patel-Schneider. Reducing OWL entailment to description logic satisfiability. In *Journal of Web Semantics*, volume 1, pages 17–29, 2003.

[HPS08] Matthew Horridge, Bijan Parsia, and Ulrike Sattler. Explanation of OWL entailments in Protégé 4. In *Proceedings of the* 7^{th} *International Semantic Web Conference (Posters & Demos)*, 2008.

[HS99] Ian Horrocks and Ulrike Sattler. A description logic with transitive and inverse roles and role hierarchies. *Journal of Logic and Computation*, 9(3):385–410, June 1999.

[HS07] Ian Horrocks and Ulrike Sattler. A tableau decision procedure for $\mathcal{SHOIQ}$. *Journal of Automated Reasoning*, 39(3):249–276, October 2007.

[HST99] Ian Horrocks, Ulrike Sattler, and Stephan Tobies. Practical reasoning for expressive description logics. In *Proceedings of the 6th International Conference on Logic Programming and Automated Reasoning*, pages 161–180. Springer, 1999.

[HST00] Ian Horrocks, Ulrike Sattler, and Stephan Tobies. Practical reasoning for very expressive description logics. *Logic Journal of the IGPL*, 8, 2000.

[HVHT05] Zhisheng Huang, Frank Van Harmelen, and Annette Ten Teije. Reasoning with inconsistent ontologies. In *Proceedings of the 19th international joint conference on Artificial intelligence*, IJCAI'05, pages 454–459. Morgan Kaufmann Publishers Inc., 2005.

[HWK06] Christian Halaschek-Wiener and Yarden Katz. Belief base revision for expressive description logics. In *Proceedings of The International Workshop on OWL: Experience and Directions*, volume 216 of *CEUR Workshop Proceedings*. CEUR-WS, 2006.

[Kal06] Aditya Kalyanpur. *Debugging and repair of owl ontologies*. PhD thesis, University of Maryland at College Park, 2006. AAI3222483.

[KH08] Uwe Keller and Stijn Heymans. The SAT-tableau calculus. In *Proceedings of the* 21^{st} *International Workshop on Description Logics*, volume 353. CEUR-WS, May 2008.

[KKIM02] Kouji Kozaki, Yoshinobu Kitamura, Mitsuru Ikeda, and Riichiro Mizoguchi. Hozo: An environment for building/using ontologies based on a fundamental consideration of "role" and "relationship". *Knowledge Engineering and Knowledge Management: Ontologies and the Semantic Web*, pages 155–163, 2002.

[Kli08] Pavel Klinov. Pronto: a non-monotonic probabilistic description logic reasoner. In *Proceedings of the 5th European semantic web conference on The semantic web: research and applications*, ESWC'08, pages 822–826. Springer, 2008.

[KPGS05] Aditya Kalyanpur, Bijan Parsia, Bernardo C. Grau, and Evren Sirin. Tableaux tracing in $\mathcal{SHOIN}$. Technical Report UMIACS-TR 2005-66, University of Maryland at College Park, 2005.

[KPHS07] Aditya Kalyanpur, Bijan Parsia, Matthew Horridge, and Evren Sirin. Finding all justifications of OWL DL entailments. In *Proceedings of the* 6^{th} *International Semantic Web Conference*, pages 267–280. Springer, 2007.

[KPP98] Sébastien Konieczny and Ramón Pino-Pérez. On the logic of merging. In *Proceedings of the International Conference on Principles of Knowledge Representation and Reasoning*, pages 488–498, 1998.

[KPS05] Aditya Kalyanpur, Bijan Parsia, and Evren Sirin. Black box techniques for debugging unsatisfiable concepts. In *Proceedings of the International Workshop on Description Logics*, page 147. CEUR-WS, 2005.

[KPS+06] Aditya Kalyanpur, Bijan Parsia, Evren Sirin, Bernardo C. Grau, and James Hendler. Swoop: A web ontology editing browser. *Web Semantics: Science, Services and Agents on the World Wide Web*, 4(2):144–153, June 2006.

[KPSCG06] Aditya Kalyanpur, Bijan Parsia, Evren Sirin, and Bernardo Cuenca-Grau. Repairing unsatisfiable concepts in OWL ontologies. In *Proceedings of the 3rd European conference on The Semantic Web: research and applications*, ESWC'06, pages 170–184. Springer, 2006.

[KPSH05] Aditya Kalyanpur, Bijan Parsia, Evren Sirin, and James Hendler. Debugging unsatisfiable classes in owl ontologies. *Journal of Web Semantics*, 3(4):268–293, December 2005.

[Lev77] Isaac Levi. Subjunctives, dispositions and chances. *Synthese*, 34:423–455, 1977. 10.1007/BF00485649.

[LLMW06] Hongkai Liu, Carsten Lutz, Maja Milicic, and Frank Wolter. Updating description logic ABoxes. In *Proceedings of the Tenth International Conference on Principles of Knowledge Representation and Reasoning*, pages 46–56. AAAI Press, 2006.

[LM04] Kevin Lee and Thomas Meyer. A classification of ontology modification. In *Proceedings of the 17th Australian joint conference on Advances in Artificial Intelligence*, pages 248–258. Springer, 2004.

[LMPB06] Kevin Lee, Thomas Meyer, Jeff Z. Pan, and Richard Booth. Computing maximally satisfiable terminologies for the description logic $\mathcal{ALC}$ with cyclic definitions. In *Description Logics*, volume 189. CEUR-WS, 2006.

[LS95] Paolo Liberatore and Marco Schaerf. Relating belief revision and circumscription. In *In Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence (IJCAI-95*, pages 1557–1563, 1995.

[LSPV08] Joey Sik Chun Lam, Derek Sleeman, Jeff Z. Pan, and Wamberto Vasconcelos. A fine-grained approach to resolving unsatisfiable ontologies. *Journal on Data Semantics*, 10:62–95, 2008.

[MLB05] Thomas Meyer, Kevin Lee, and Richard Booth. Knowledge integration for description logics. In *Proceedings of the* 20^{th} *National Conference on Artificial intelligence - Volume 2*, pages 645–650. AAAI Press, 2005.

[MLBP06] Thomas Meyer, Kevin Lee, Richard Booth, and Jeff Z. Pan. Finding maximally satisfiable terminologies for the description logic $\mathcal{ALC}$. In *Proceedings of the 21st national conference on Artificial intelligence - Volume 1*, pages 269–274. AAAI Press, 2006.

[Mot08] Boris Motik. KAON2 - scalable reasoning over ontologies with large data sets. *ERCIM News*, 2008(72), 2008.

[MPSH07] Boris Motik, Peter F. Patel-Schneider, and Ian Horrocks. OWL 1.1 Web Ontology Language – structural specification and functional-style syntax. Editors draft, The University of Manchester, May 2007.

[MPSP+08] Boris Motik, Peter F. Patel-Schneider, Bijan Parsia, Conrad Bock, Achille Fokoue, Peter Haase, Rinke Hoekstra, Ian Horrocks, Alan Ruttenberg, Uli

Sattler, and Mike Smith. OWL 2 web ontology language: Structural specification and functional-style syntax. Technical report, W3C, 2008.

[Neb90] Bernhard Nebel. *Reasoning and revision in hybrid representation systems*. Springer, 1990.

[NFM00] Natalya Fridman Noy, Ray W. Fergerson, and Mark A. Musen. The knowledge model of Protege-2000: Combining interoperability and flexibility. In *Proceedings of the 12th European Workshop on Knowledge Acquisition, Modeling and Management*, EKAW '00, pages 17–32. Springer, 2000.

[Pep07] Pavlos Peppas. Belief revision. In Frank van Harmelen, Frank van Harmelen, Vladimir Lifschitz, and Bruce Porter, editors, *Handbook of Knowledge Representation*. Elsevier, 2007.

[QHH+08] Guilin Qi, Peter Haase, Zhisheng Huang, Qiu Ji, Jeff Z. Pan, and Johanna Völker. A kernel revision operator for terminologies – algorithms and evaluation. In *Proceedings of the 7th International Conference on The Semantic Web*, ISWC '08, pages 419–434. Springer, 2008.

[QLB06] Guilin Qi, Weiru Liu, and David A. Bell. Knowledge base revision in description logics. In *Proceedings of the 10th European conference on Logics in Artificial Intelligence*, JELIA'06, pages 386–398. Springer, 2006.

[QP07] Guilin Qi and Jeff Z. Pan. An analysis of approaches to resolving inconsistencies in DL-based ontologies. In *Proceedings of International Workshop on Ontology Dynamics*, June 2007.

[QR92] Joachim Quantz and Véronique Royer. A preference semantics for defaults in terminological logics. In *Proceedings of the International Conference of Principles of Knowledge Representation and Reasoning*, pages 294–305. Morgan Kaufmann, 1992.

[Rei87] R. Reiter. A theory of diagnosis from first principles. *Artificial Intelligence*, 32(1):57–95, April 1987.

[RW06] Marcio M. Ribeiro and Renata Wassermann. Base revision in description logics - preliminary results. In *Proceedings of the International Workshop on Ontology Dynamics*, 2006.

[RW08] Marcio M. Ribeiro and Renata Wassermann. On the relation between remainder sets and kernels. In *Encontro Brasileiro de Logica*, 2008.

[RW09] Márcio M. Ribeiro and Renata Wassermann. Base revision for ontology debugging. *Journal of Logic and Computation*, 19(5):721–743, October 2009.

[SC03] Stefan Schlobach and Ronald Cornet. Non-standard reasoning services for the debugging of description logic terminologies. In *Proceedings of the 18th international joint conference on Artificial intelligence*, IJCAI'03, pages 355–360. Morgan Kaufmann Publishers Inc., 2003.

[Sch05a] Stefan Schlobach. Debugging and semantic clarification by pinpointing. In *Proceedings of the Second European conference on The Semantic Web: research and Applications*, ESWC'05, pages 226–240. Springer, 2005.

[Sch05b] Stefan Schlobach. Diagnosing terminologies. In *Proceedings of the 20th national conference on Artificial intelligence - Volume 2*, AAAI'05, pages 670–675. AAAI Press, 2005.

[SHCvH07] Stefan Schlobach, Zhisheng Huang, Ronald Cornet, and Frank van Harmelen. Debugging incoherent terminologies. *Journal of Automated Reasoning*, 39(3):317–349, October 2007.

[SMH08] Rob Shearer, Boris Motik, and Ian Horrocks. Hermit: A highly-efficient owl reasoner. In Catherine Dolbear, Alan Ruttenberg, and Ulrike Sattler, editors, *Proceedings of the International Workshop on OWL: Experience and Directions*, volume 432. CEUR-WS, 2008.

[SPG+07] Evren Sirin, Bijan Parsia, Bernardo Cuenca Grau, Aditya Kalyanpur, and Yarden Katz. Pellet: A practical OWL-DL reasoner. *Journal of Web Semantics*, 5(2):51–53, June 2007.

[SSS91] Manfred Schmidt-Schaubßand Gert Smolka. Attributive concept descriptions with complements. *Artificial Intelligence*, 48(1):1–26, February 1991.

[Tee94] Gunnar Teege. Making the difference: A subtraction operation for description logics. In *Proceedings of the International Conference on Principles of Knowledge Representation and Reasoning*, 1994.

[TH06] Dmitry Tsarkov and Ian Horrocks. FaCT++ description logic reasoner: system description. In *Proceedings of the Third international joint conference on Automated Reasoning*, IJCAR'06, pages 292–297. Springer, 2006.

[Tob01] Stephan Tobies. Complexity results and practical algorithms for logics in knowledge representation. *CoRR*, cs.LO/0106031, 2001.

[YK03] Marco S. Yannis and Yannis Kalfoglou. Using information-flow theory to enable semantic interoperability. In *Artificial Intelligence Research and Development*. IOS Press, 2003.