5-10-2017

Handling Inconsistency in Knowledge Bases

Badrinath Jayakumar

HANDLING INCONSISTENCY IN KNOWLEDGE BASES

by

BADRINATH JAYAKUMAR

Under the Direction of Rajshekhar Sunderraman, PhD

ABSTRACT

Real-world automated reasoning systems, based on classical logic, face logically inconsistent information, and they must cope with it. It is onerous to develop such systems because classical logic is explosive. Recently, progress has been made towards semantics that deal with logical inconsistency. However, such semantics was never analyzed in the aspect of inconsistency tolerant relational model.

In our research work, we use an inconsistency and incompleteness tolerant relational model called ”Paraconsistent Relational Model.” The paraconsistent relational model is an extension of the ordinary relational model that can store, not only positive information but also negative information. Therefore, a piece of information in the paraconsistent relational model has four truth values: true, false, both, and unknown.

However, the paraconsistent relational model cannot represent disjunctive information (disjunctive tuples). We then introduce an extended paraconsistent relational model called disjunctive paraconsistent relational model. By using both the models, we handle inconsistency - similar to the notion of quasi-classic logic or four-valued logic – in deductive databases (logic programs with no functional symbols).

In addition to handling inconsistencies in extended databases, we also apply inconsistent tolerant reasoning technique in semantic web knowledge bases. Specifically, we handle inconsistency assosciated with closed predicates in semantic web. We use again the paraconsistent approach to handle inconsistency.

We further extend the same idea to description logic programs (combination of semantic web and logic programs) and introduce dl-relation to represent inconsistency associated with description logic programs.

INDEX WORDS: Paraconsistent logics, Paraconsistent Relations, Description logic, Semantic Web, Description Logic Programs

TITLE: HANDLING INCONSISTENCY IN KNOWLEDGE BASES

by

BADRINATH JAYAKUMAR

A Dissertation Submitted in Partial Fulfillment of the Requirements for the Degree of

Doctor of Philosophy

in the College of Arts and Sciences

Georgia State University

2017

TITLE: HANDLING INCONSISTENCY IN KNOWLEDGE BASES

by

BADRINATH JAYAKUMAR

Committee Chair: Rajshekhar Sunderraman

Committee: Rafal A. Angryk

Florian Enescu

Yingshu Li

Electronic Version Approved:

Office of Graduate Studies

College of Arts and Sciences

Georgia State University

May 2017

DEDICATION

I like to dedicate this dissertation to my parents Jayakumar and Meera Bai who support me every day in my life. I also want to thank my brother Srinivasan who guided me to choose Ph.D. career.

ACKNOWLEDGEMENTS

This dissertation work would not have been possible without the support of many people. I want to express my gratitude to my advisor Rajshekhar Sunderraman for believing in me, providing me an opportunity and guiding me at every step in my PhD career. I also express my thanks to each of my committee members - Dr. Yingshu Li, Dr. Rafal A Angryk, and Dr. Florian Enescu, for their continuous support and encouragement. I would also like to thank many Georgia State University professors and students for helping through valuable knowledge sharing and contribution. Finally, I wish to thank all of my family members and all my friends for their unconditional support, love, patience and understanding.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF ABBREVIATIONS

- GSU - Georgia State University
- CS - Computer Science
- QC - Quasi-classic
- DL - Description Logic

CHAPTER 1

INTRODUCTION

The systems based on classical logic are purely deductive in nature. In other words, they embody monotonicity: if a statement is deducible from a set of statements, the statement is still deductible if we enlarge the set. But the real world reasoning is non-monotonic: a statement believed to be true in a set of statements which later turns out to be false. The classical logic reasoning fails to support non-monotonic reasoning. For example, a set of statements contains inconsistent statements. In this case, classical logic deduction becomes trivialized. Concretely, any statement can be driven from a set of statements.

To handle such inconsistencies, it is required to seek a different formalization of statements. We need to look for logics that are non-classical in nature. Thus, we need to go for paraconsistent logic. Paraconsistent logic [1–3] does not trivialize the result in the presence of inconsistent information. Four-valued logic [4], which is a type of paraconsistent logic, was introduced in logic programming by Blair and Subrahmanian [5].

Three prominent works have been done in positive extended disjunctive deductive databases with respect to inconsistencies: The first, answer set semantics, by Gelfond and Lifschitz [6], trivializes the results in the presence of inconsistencies. The second, p-minimal models, by Sakama and Inoue [7], which is based on four-valued logic [4], tolerates inconsistencies. In addition to that, for both logic programs and disjunctive logic programs, many works have been proposed [8–11], where all of the approaches are based on four-valued logic. The third, the QC models, by Zhang et al.. [12], has stronger inference power than p-minimal models because the QC models support disjunctive syllogism and disjunction introduction. Moreover, the QC models are based on QC logic [13].

In addition to that, other approaches [14–18] are available for non-monotonic reasoning, but we first consider only QC logic for inconsistency handling in our work. The reason we have chosen QC logic is if we find QC models, it is possible to find paraconsistent-models (p-models) in a similar fashion. In addition to that, it is easier to adapt the fixed-point semantics of QC logic to paraconsistent relational model.

The paraconsistent relational model moves a step forward and completes the relational model by representing both positive and negative information about any given relation. Bagai and Sunderraman [19] first proposed the model. The authors have given two applications for the paraconsistent relational model: weak well-founded semantics [19] and well-founded semantics [20] for general deductive databases. Bagai and Sunderraman find the models by constructing a system of algebraic equations for the clauses in the database.

In this thesis, we show disjunctive syllogism for positive extended disjunctive deductive databases, which are logic programs without functional symbols, in the paraconsistent relational model. Our solution is similar to QC models of QC logic programs. In addition to using the paraconsistent relational model to construct QC models, we also introduce the disjunctive paraconsistent relational model in our work.

The inconsistency-tolerant reasoning algorithm we designed for the QC models using paraconsistent relational databases serializes every clause in the disjunctive deductive database into a corresponding equation. During the serialization, we associate a relation for every predicate symbol in the clause. The serialized equation contains only set theoretic and relational theoretic operators. As an optimization, we unionize the right-hand side expression of the equation whose left-hand side expressions are the same. The second step is to solve the equations and incrementally find the minimal QC models.

Similarly, the disjunctive paraconsistent relational model that is employed to determine the QC models could be used for p-minimal models. The p-minimal models do not $focus$ the disjuncts during the construction of models, which make it different from the QC models. The expressive power of the p-minimal model is very much lesser than the expressive power of the QC models.

We also show handling inconsistencies in description logic that are carried over to local closed world reasoning, where the knowledge base consists of open world assumption (OWA) predicates (concepts or roles) and closed world assumption (CWA) predicates. When data from a relational database are migrated to a knowledge base (KB), the KB may become inconsistent. Consequently, querying becomes problematic in the KB as opposed to the database, where the database is consistent.

In this thesis, we present an approach to represent $\mathcal{ALC}$ with closed predicates in four-valued logic and show that four-valued $\mathcal{ALC}$ with closed predicates is sound with respect to two-valued $\mathcal{ALC}$ with closed predicates. Similar to [40, 43] and [41], we transform four-valued $\mathcal{ALC}$ with closed predicates to two valued $\mathcal{ALC}$ with closed predicates and reason two valued $\mathcal{ALC}$ with closed predicates over standard reasoners. We also introduce new inference rules to reason in the presence of closed predicates and prove the correctness of the tableau algorithm with the new inference rules.

We also show the method of using the paraconsistent relational model to description logic programs. Description logic programs provide a significant degree of expressiveness, substantially greater than the RDF-Schema fragment of description logic. The essential idea of the description logic program is the flow of information between description logic and logic programs. The flow of information happens with the help of description logic atoms. They are similar to regular atoms in the logic program, but they get the information from description logic knowledge base and use it with the clauses of the logic programs. Our approach starts with finding an equivalent relation (description logic relation) for the description logic atom and defining a proper domain for every attribute in the description logic relation. Then, using the description relation, we are working towards finding the fixed-point semantics of description logic programs.

The problem with existing methods (i.e. the methods that do not use paraconsistent relational model)– QC models, p-models, and description logic program – of finding the models is that they are too slow. In other words, it works one clause at a time while determining any model. There are two advantages of our approach: it operates on a set of tuples instead of a "tuple-at-a-time" basis and the algebraic expression in the algebraic equation can be optimized based on various laws of equality. The optimizations are similar to the ordinary relations case where selections and projections are pushed deeper into expressions whenever possible [19].

The rest of the thesis is structured as follows: In Chapter 2 we describe the previous works related to inconsistency in databases and description logic. We provide background information of deductive database, paraconsistent relation model, $\mathcal{ALC}$, and description logic programs in Chapter 3. In Chapter 4 we present the construction of quasi-classic model for positive extended disjunctive deductive databases. Next, in Chapter 5 we give the construction of p-minimal models for positive extended disjunctive deductive databases. We propose a technique to handle inconsistencies in closed world predicates in semantic web in Chapter 6. In Chapter 7 we propose a novel way of finding the fixed-point semantics of description logic programs. Finally, we conclude this thesis in Chapter 8.

CHAPTER 2

LITERATURE REVIEW

2.1 Inconsistency in Databases

Many research works are performed on representing negative information in the databases. Since this representation leads to inconsistencies, paraconsistent databases are required to handle it. In [19], Bagai and Sunderraman developed a framework to represent negative facts in relational database, which is based on four-valued logic. The four-valued relation represents both positive and negative information, and negative facts that are derived based on open world assumption. They also developed an application for it that finds the well founded semantics of general deductive databases. For general deductive databases and disjunctive deductive databases, various paraconsistent semantics have been proposed [20, 21], where all of them are based upon Belnap's four-valued model [4]. Even for logic program especially disjunctive logic program various paraconsistent semantics are proposed [8, 10], but all of those works come under Belnap's four-valued model.

However, the multi-valued logic doesn't support disjunctive syllogism [1]. For example, suppose a knowledge base contains {passed $\vee$ failed} about a student. When new information about the student comes to the knowledge base {$\neg$failed}, the four-valued logic gives two p-models [7] {{passed, $\neg$failed}, {failed, $\neg$failed}}. Hence, $\neg$failed is the only logical consequence of the two models. Whether the student is passed or not cannot be inferred with four-valued logic.

It's very clear from the above example that this four-valued logic doesn't behave as expected in such situations. In order to get accurate models, it is required to look for other paraconsistent logic to improve the ability of reasoning. Hence, we use QC logic [13] to address the issue.

2.2 Inconsistency in Description Logic

To handle inconsistencies in description logic, we focus on the paraconsistent method in this thesis. Particularly, we are very interested in the works [40] and [41], where the authors introduced four-valued description logic and transformed it to two-valued description logic. Then the transformed description logic is reasoned over standard reasoners. The advancement of this approach is called quasi-classic description logic [42], which has stronger inference power. But this thesis focuses on handling inconsistencies in closed predicates and not on the inference power. Specifically, we borrow some ideas from [43] to represent closed predicates in four-valued description logic.

Many formalisms have been proposed to integrate description logic and rules: SWRL [54–56], DL-Safe rules [57–59], DLP [60, 61], $\mathcal{AL}$-log [62, 63], CARIN [64, 65], $\mathcal{DL}$+*log* [66–70], Horn-$\mathcal{SHIQ}$ [57, 71, 72], Hybrid MKNF [73–76], *dl-programs* [29–33], disjunctive dl-programs [77], quantified equilibrium logic for hybrid knowledge bases [77], and description graphs [78–81]. We observed that no type of formalism employs the paraconsistent relational model [19] to provide the semantics for the integration of rules and description logic, which has the capbalities to handle incompleteness and inconsistencies.

CHAPTER 3

BACKGROUND

3.1 Background

The background for this thesis is divided into four parts: Paraconsistent Relational Model,Positive Extended Disjunctive Deductive Database, $\mathcal{ALC}$, and description logic programs. Background on these four topics are essential to understand the works presented in this thesis.

3.1.1 Paraconsistent Relational Model

Unlike normal relations where we only retain information believed to be true of a particular predicate, we also retain what is believed to be false of a particular predicate in the paraconsistent relational model. Let a relation scheme Σ be a finite set of attribute names, where for any attribute name $A \in \Sigma$, $dom(A)$ is a non-empty domain of values for A. A tuple on Σ is any map $t : \Sigma \rightarrow \bigcup_{A\in\Sigma}$ dom(A), such that $t(A) \in$ dom(A) for each $A \in \Sigma$. Let $\tau(\Sigma)$ denote the set of all tuples on Σ. An ordinary relation on scheme Σ is thus any subset of $\tau(\Sigma)$. The paraconsistent relation on a scheme Σ is a pair $< R^+, R^- >$ where R^+ and R^- are ordinary relations on Σ. Thus R^+ represents the set of tuples believed to be true of R, and R^- represents the set of tuples believed to be false.

Algebraic Operators. Two types of algebraic operators are defined here: i)Set Theoretic Operators, and ii) Relational Theoretic Operators.

Set Theoretic Operators. Let R and S be two paraconsistent relations on scheme Σ.

Union. The union of R and S, denoted $R\dot{\cup}$S, is a paraconsistent relation on scheme Σ, given that

$$(R\dot{\cup}S)^+ = R^+ \cup S^+, (R\dot{\cup}S)^- = R^- \cap S^-$$

Complement. The complement of R, denoted $\dot{-}R$, is a paraconsistent relation on scheme Σ, given that

$$\dot{-}R^+ = R^-, \dot{-}R^- = R^+$$

Intersection. The intersection of R and S, denoted $R \dot{\cap} S$, is a paraconsistent relation on scheme Σ, given that

$$(R \dot{\cap} S)^+ = R^+ \cap S^+, (R \dot{\cap} S)^- = R^- \cup S^-$$

Difference. The difference of R and S, denoted $R \dot{-} S$, is a paraconsistent relation on scheme Σ, given that

$$(R \dot{-} S)^+ = R^+ \cap S^-, (R \dot{-} S)^- = R^- \cup S^+$$

Example 1. *Let {a, b, c} be a common domain for all attribute names, and let R and S be the following paraconsistent relations on schemes {X} and {X} respectively:*

$$R^+ = \{(a), (b)\}, R^- = \{(c)\}$$

$$S^+ = \{(c), (b)\}, S^- = \{(a)\}$$

$R \dot{\cup} S$ is

$$(R \dot{\cup} S)^+ = \{(a), (b), (c)\}$$

$$(R \dot{\cup} S)^- = \{\}$$

$R \dot{\cap} S$ is

$$(R \dot{\cap} S)^+ = \{(b)\}$$

$$(R \dot{\cap} S)^- = \{(a), (c)\}$$

$\dot{-} R$ is

$$\dot{-} R^+ = \{(c)\}$$

$$\dot{-} R^- = \{(a), (b)\}$$

$R \dot{-} S$ is

$$(R \dot{-} S)^+ = \{(a)\}$$

$$(R \dot{-} S)^- = \{(b), (c)\}$$

Relation Theoretic Operators. Let Σ and Δ be relation schemes such that $\Sigma \subseteq \Delta$ and let R and S be paraconsistent relations on schemes Σ and Δ.

Join. The join of R and S, denoted $R \dot{\bowtie} S$, is a paraconsistent relation on scheme $\Sigma \cup \Delta$, given that

$$(R \dot{\bowtie} S)^+ = R^+ \bowtie S^+, (R \dot{\bowtie} S)^- = (R^-)^{\Sigma \cup \Delta} \cup (S^-)^{\Sigma \cup \Delta}$$

Projection. The projection of R onto Δ , denoted $\dot{\pi}_\Delta(R)$, is a paraconsistent relation on Δ, given that

$$\dot{\pi}_\Delta(R)^+ = \pi_\Delta(R^+)^{\Sigma \cup \Delta}, \dot{\pi}_\Delta(R)^- = \{t \in \tau(\Delta) \mid t^{\Sigma \cup \Delta} \subseteq (R^-)^{\Sigma \cup \Delta}\}$$

where π_Δ is the usual projection over Δ of ordinary relations.

Selection. Let F be any logic formula involving attribute names in Σ, constant symbols, and any of these symbols {==, ¬, ∧, ∨}. Then, the selection of R by F, denoted $\dot{\sigma}_F(R)$, is a paraconsistent relation on scheme Σ, given that

$$\dot{\sigma}_F(R)^+ = \sigma_F(R^+), \dot{\sigma}_F(R)^- = R^- \cup \sigma_{\neg F}(\tau(\Sigma))$$

where σ_F is a usual selection of tuples satisfying F from ordinary relations.

The following example is taken from Bagai and Sunderraman's paraconsistent relational data model [19].

Example 2. *Strictly speaking, relation schemes are sets of attribute names. However, in this example the authors [19] treat them as ordered sequence of attribute names, so tuples can be viewed as the usual lists of values. Let {a, b, c} be a common domain for all attribute names, and let R and S be the following paraconsistent relations on schemes $\langle X, Y\rangle$ and $\langle Y, Z\rangle$ respectively:*

$R^+ = \{(b, b), (b, c)\}, R^- = \{(a, a), (a, b), (a, c)\}$

$S^+ = \{(a, c), (c, a)\}, S^- = \{(c, b)\}$.

Then, $R \dot{\bowtie} S$ is the paraconsistent relation on scheme $\langle X, Y, Z\rangle$:

$(R \dot{\bowtie} S)^+ = \{(b, c, a)\}$

$(R \dot{\bowtie} S)^- = \{(a, a, a), (a, a, b), (a, a, c), (a, b, a), (a, b, b), (a, b, c), (a, c, a),$

$(a, c, b), (a, c, c), (b, c, b), (c, c, b)\}$

Now, $\dot{\pi}_{\langle X,Z\rangle}(R \dot{\bowtie} S)$ becomes the paraconsistent relation on scheme $\langle X, Z\rangle$:

$\dot{\pi}_{\langle X,Z\rangle}(R \dot{\bowtie} S)^+ = \{(b, a)\}$

$\dot{\pi}_{\langle X,Z\rangle}(R \dot{\bowtie} S)^- = \{(a, a), (a, b), (a, c)\}$

Finally, $\dot{\sigma}_{\neg X=Z}(\dot{\pi}_{\langle X,Z\rangle}(R \dot{\bowtie} S))$ becomes the paraconsistent relation on scheme $\langle X, Z\rangle$:

$\dot{\sigma}_{\neg X=Z}(\dot{\pi}_{\langle X,Z\rangle}(R \dot{\bowtie} S))^+ = \{(b, a)\}$

$\dot{\sigma}_{\neg X=Z}(\dot{\pi}_{\langle X,Z\rangle}(R \dot{\bowtie} S))^- = \{(a, a), (a, b), (a, c)(b, b), (c, c)\}$

Next, we discuss disjunctive deductive database in detail.

3.1.2 Positive Extended Disjunctive Deductive Database

Syntax. Given a first order language $\mathcal{L}$, a disjunctive deductive database P [22] consists of logical inference rules of the form

$$r \text{ (rule) } = l_0 \vee \cdots \vee l_n \leftarrow l_{n+1} \ldots l_m$$

$l_0 \ldots l_n$ is called head of the rule and $l_{n+1} \ldots l_m$ is called body of the rule. A rule is called fact if the rule has no body. A rule is called denial rule if the rule has only body and no head. A rule is called definite clause or horn clause, if the rule has only one literal in the head and has some literal in the body. A rule is called positive disjunctive rule if the rule has both body and head. Concretely, the rule r is called positive extended disjunctive rule, if $l_0, \ldots, l_n, l_{n+1}, \ldots, l_m$ are either positive or negative ($\neg$) literals.

For the given syntax of a positive extended disjunctive deductive database, we reproduce the fixed point semantics of P [12].

Fixed Point Semantics. Let P be a positive extended disjunctive deductive database and $\mathcal{I}$

be a set of interpretations, then $\mathcal{T}_P(\mathcal{I}) = \bigcup_{I \in \mathcal{I}} T_P(I)$

$$T_P(I) = \begin{cases} \emptyset, \text{ if } l_{n+1}, \ldots, l_m \subseteq I \text{ for some} \\ \text{ ground constraint } \leftarrow l_{n+1} \ldots l_m \text{from } P. \\ \{J \mid \text{for each ground rule} \\ \ r_i \colon l_0 \vee \cdots \vee l_n \leftarrow l_{n+1} \ldots l_m \text{ such that} \\ \ \{l_{n+1} \ldots l_m\} \subseteq I, J = I \cup \bigcup_{r_i} J' \text{ where} \\ J' \in Lits(focus(l_0 \vee \cdots \vee l_n, I))\}, \text{otherwise.} \end{cases}$$

In the definition of $T_P(I)$, $focus$ removes complementary literals from disjunction ($focus(l_0 \vee l_1, I) = l_0$ where $I = \{\neg l_1\}$). If all disjuncts ($l_0 \ldots l_n$) are available in I as complementary literals, then the disjunction of literals becomes the conjunction of literals. *Lits* of conjunction gives a set of conjuncts. On the other hand, *Lits* of disjunction is a collection of sets where every set in the collection contains a disjunct.

The T_P definition contains the constraint. we write it for the sake of completeness, but our contribution will not address the constraint.

The following two propositions are vital for our result.

Proposition 1. *For any positive extended disjunctive deductive database P,* $\mathcal{T}_P$ *is finite and* $\mathcal{T}_P \uparrow n = \mathcal{T}_P \uparrow \omega$ *where n is a successor ordinal and* ω *is a limit ordinal.*

Proposition 2. *For any positive extended disjunctive deductive database P, Minimal QC Model(P) = min(*$\mu(\mathcal{T}_P \uparrow \omega)$ [1]*) where min () stands for sets with a minimum number of literals.*

Next, we review without getting into too much detail of $\mathcal{ALC}$ and the mix of open and closed predicates. For comprehensive reading on all these topics, reader is requested to refer to [23–26].

Table (3.1) Syntax and Semantics of $\mathcal{ALC}$

Syntax	Name	Semantics
$\bot$	bottom	$\emptyset$
$\top$	top	$\Delta^{\mathcal{I}}$
$A \sqcap B$	conjunction	$A^{\mathcal{I}} \cap B^{\mathcal{I}}$
$A \sqcup B$	disjunction	$A^{\mathcal{I}} \cup B^{\mathcal{I}}$
$\neg A$	negation	$\Delta^{\mathcal{I}} \setminus A^{\mathcal{I}}$
$\exists r.A$	existential restrictions	$\{x \mid \exists y \in \Delta^{\mathcal{I}} : (x,y) \in r^{\mathcal{I}}\ and\ y \in A^{\mathcal{I}}\}$
$\forall r.A$	universal restrictions	$\{x \mid \forall y \in \Delta^{\mathcal{I}} : (x,y) \in r^{\mathcal{I}} \rightarrow y \in A^{\mathcal{I}}\}$
$A \sqsubseteq B$	general concept inclusion	$A^{\mathcal{I}} \subseteq B^{\mathcal{I}}$
$A(a)$	open concept assertion	$a^{\mathcal{I}} \in A^{\mathcal{I}}$
$C(a)$	closed concept assertion	$a^{\mathcal{I}} \in C^{\mathcal{I}}$
$r(a,b)$	(open or closed) role assertion	$(a^{\mathcal{I}}, b^{\mathcal{I}}) \in r^{\mathcal{I}}$

3.1.3 $\mathcal{ALC}$

Let N_C, N_R and $N_{\mathcal{I}}$ be a disjoint set of concepts, roles, and individuals. The syntax and semantics of $\mathcal{ALC}$ are shown in Table 3.1. In Table 3.1, $\{A, B\} \subseteq N_C$, $r \in N_R$, and $\{a, b\} \subseteq N_{\mathcal{I}}$. $\mathcal{ALC}$ KB (knowledge base) consists of a general TBox (terminological box) and an ABox (assertion box). A TBox has a set of axioms which are concept inclusions. An ABox has concept assertions and role assertions. In $\mathcal{ALC}$, a concept is

$$C = \top \mid \bot \mid A \sqcap B \mid A \sqcup B \mid \exists r.A \mid \forall r.A \mid \neg A \tag{3.1}$$

Concept descriptions are defined inductively from (1). The semantics of $\mathcal{ALC}$ is given by an interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ where $\Delta^{\mathcal{I}}$ is a finite non-empty set abstract domain and $\cdot^{\mathcal{I}}$ is a mapping where each concept is assigned to a subset of $\Delta^{\mathcal{I}}$ and each abstract role to a subset of $\Delta^{\mathcal{I}}$x $\Delta^{\mathcal{I}}$. A model of the KB is defined as an interpretation which satisfies every axiom in the TBox and every assertion in the ABox. A knowledge base is called satisfiable (unsatisfiable) iff there exists (does not exist) a model.

In this thesis, the entailment in $\mathcal{ALC}$ is denoted by $\models_2$. It represents entailment in two-valued logic. This thesis focuses on querying the potentially inconsistent KB. We consider the concept

[1] $\mu(\mathcal{T}_P \uparrow \omega) = \{I \mid I \in \mathcal{T}_P \uparrow \omega\ and\ I \in \mathcal{T}_{\mathcal{P}}(\{I\})\}$

descriptions as queries that can be derived from (1). Two types of queries can be given to the KB: i) boolean query (returns true/false) and ii) non-boolean query (returns answer tuples). Query entailment is deciding whether a boolean query is true/false and query answering is finding a set of answer tuples for a non-boolean query [25]. In this thesis, we work with an instance of concept descriptions i.e. boolean query (e.g. $(C \sqcap D)(a)$). Let KB be a knowledge base and Q be a boolean query. If, for every interpretation $\mathcal{I}$, $\mathcal{I} \models_2 KB$ and $\mathcal{I} \models_2 Q$, then $KB \models_2 Q$ [2].

Finally, we see the syntax and semantics of merging logic programs and semantic web, which is called description logic program.

3.1.4 Descripiton Logic Programs

Logic program P, which consists of a set of rules, and description logic L combine to form a description logic program. The rules in logic programs also contain queries to L. In the following, we briefly describe logic programs, description logic, and description logic programs. However, to get an in-depth understanding, we request the readers to read Fitting and Melvin's survery on fixed-point semantics of logic programming [27], $\mathcal{SHOIN}(\mathbf{D})$ [28], and Eiter et al.'s *dl-programs* [29–33].

Definite Logic Programs (*P*) In this subsection, we define the syntax and the fixed-point semantics of logic programs [27].

Syntax. Similar to Eiter et al.'s well-founded semantics of *dl-programs* [30, 31], we consider function free first-order vocabulary $\Phi = (\mathcal{P}, \mathcal{C})$, which consists of non-empty finite sets of constants $\mathcal{C}$ and predicate symbols $\mathcal{P}$. In addition to that, let $\mathcal{X}$ be a set of variables. A term is either a variable from $\mathcal{X}$ or a constant from $\mathcal{C}$. An atom is of the form $p(t_1, \ldots, t_n)$ where $p \in \mathcal{P}$ and $t_1, \ldots, t_n$ are terms. In this thesis, we consider only POSITIVE logic programs. Therefore, the rules are of the following form:

$$l_0 \leftarrow l_1, \ldots, l_z$$

[2]In this thesis, we refer boolean queries as queries

where $z \geq 1$.

In the above rule, the atom l_0 is the head of the rule and the conjunction of atoms $l_1, \ldots, l_z$ is called the body of the rule. The rule is called a positive rule because it does not have default negated (*not*) atoms. A definite logic program (or logic program) P is a finite set of rules.

In this thesis, we do not consider literals in rules. Such restriction is similar to Eiter et al.'s well-founded semantics of *dl-programs* [30, 31].

Fixed-point Semantics. A term, atom, or rule is called ground if it contains no variables. The *Herbrand Universe* of the underlying language is the set of all ground terms. The *Herbrand Base* of the language is the set of all ground atoms; a *Herbrand Interpretation* of the language is any subset of the Herbrand Base. Let I be a Herbrand Interpretation for the logic program P. Let P^* be the ground instances of rules in P. Since P does not have function symbols, P^* is always finite. Then, $T_P(I)$ (immediate consequence operator) is a Herbrand Interpretation, given by

$$T_P(I) = \{l_0 \mid \text{ for some rule } l_0 \leftarrow l_1, \ldots, l_z \text{ in } P^*, \{l_1, \ldots, l_z\} \subseteq I\}$$

It is well known that T_P always possesses a least fixed-point with respect to the partial order of set inclusion. The least fixed-point can be shown to be the minimal model for P. This model is also known to be $T_P \uparrow \omega$, where the ordinal power of T_P is given by:

Definition 1. *For any ordinal α,*

$$T_P \uparrow \alpha = \begin{cases} \emptyset \textit{ if } \alpha = 0, \\ T_P(T_P \uparrow (\alpha - 1)) \textit{ if } \alpha \textit{ is a successor ordinal,} \\ \bigcup_{\beta<\alpha}(T_P \uparrow \beta) \textit{if } \alpha \textit{ is a limit ordinal.} \end{cases}$$

The following observation for any logic program is relevant:

Proposition 3. *For any logic program P, the upward closure ordinal of T_P is finite, i.e. there is a number $n \geq 0$ such that $T_P \uparrow n = T_P \uparrow \omega$.*

Description Logic (L) In this subsection, we discuss $\mathcal{SHOIN}(\mathbf{D})$, which is the logical underpinning of OWL DL [28].

Syntax. Let E and V be a set of elementary datatypes and data values. A datatype theory $D = (\Delta^D, \cdot^D)$ consists of a datatype (or concrete) domain Δ^D and a mapping $\cdot^D$ that assigns to every elementary datatype a subset of Δ^D and to every elementary data value an element of Δ^D. The mapping $\cdot^D$ is extended to all datatypes by $\{v_1, \ldots\}^{\mathbf{D}} = \{v_1^{\mathbf{D}}, \ldots\}$. Let $\Psi = (A \cup R_A \cup R_D, I \cup V)$ be the vocabulary of the description logic, where A, R_A, R_D, and I are pairwise disjoint sets of atomic concepts, abstract roles, datatype (or concrete) roles and individuals. Table 1 describes the syntax and semantics of $\mathcal{SHOIN}(\mathbf{D})$. In Table 3.2, $\mathbf{R}_A^-$ is the set of inverses R^- of all $R \in \mathbf{R}_A$. A role is an element of $\mathbf{R}_A \cup \mathbf{R}_D \cup \mathbf{R}_A^-$. Complex concepts are defined inductively from the second part of Table 3.2. A description knowledge base is a finite set of axioms, where each axiom is one of the axiom from the third part of Table 3.2.

Semantics. We define the semantics of $\mathcal{SHOIN}(\mathbf{D})$ in terms of first-order interpretation.

An interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$, with respect to a datatype theory $\mathbf{D} = (\Delta^{D \cdot D})$, consists of a nonempty domain $\Delta^{\mathcal{I}}$ disjoint from Δ^D, and $\cdot^{\mathcal{I}}$ is a valuation function defined inductively as shown in the first and second parts of Table 1. The satisfaction of a DL axiom F in the interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ with respect to a datatype theory $\mathbf{D}= (\Delta^D, \cdot^D)$, denoted $\mathcal{I} \models F$, is given by the third part of Table 1. The interpretation satisfies an axiom F, or the interpretation is a model of F iff $\mathcal{I} \models F$. $\mathcal{I}$ is a model of knowledge base L ($\mathcal{I} \models L$) iff $\mathcal{I} \models F$ for all $F \in L$. L is satisfiable (unsatisfiable) iff L has a model (no model). An axiom $F(\neg F)$ is a logical consequence of L, denoted $L \models F(L \models \neg F)$, iff every model of L satisfies (does not satisfy) F.

Description Logic Programs (KB) In this subsection, we review Eiter et al.'s *dl-program* [29–33].

Syntax. The vocabularies of the logic program and description logic in any description logic program are defined in the previous two subsections. An important assumption is that $\mathbf{A} \cup \mathbf{R}_A \cup \mathbf{R}_D$ is disjoint from $\mathcal{P}$ where $\mathcal{P}$ is a set of predicate symbols, while $I_P \subseteq C \subseteq \mathbf{I} \cup \mathbf{V}$, where I_P is the set of all constant symbols appearing in P. As we said earlier, description logic programs contain

Table (3.2) Syntax and Semantics of $\mathcal{SHOIN}(\mathbf{D})$

Name	Syntax	Semantics
atomic concept	$C \in \mathbf{A}$	$C^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$
individual	$a \in \mathbf{I}$	$a^{\mathcal{I}} \in \Delta^{\mathcal{I}}$
abstract role	$R \in \mathbf{R}_A \cup \mathbf{R}_A^-$	$R^{\mathcal{I}} \in \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$
datatype	D	$D^{\mathbf{D}} \subseteq \Delta^{\mathbf{D}}$
concrete or datatype role	$U \in \mathbf{R}_D$	$U^{\mathcal{I}} \in \Delta^{\mathcal{I}} \times \Delta^{\mathbf{D}}$
data values	$v \in \mathbf{V}$	$v^{\mathcal{I}} = v^{\mathbf{D}}$
oneOf	$\{o_1, \ldots, o_n\}, o_i \in \mathbf{I}$	$\{o_1^{\mathcal{I}}, \ldots, o_n^{\mathcal{I}}\}$
top	$\top$	$\top^{\mathcal{I}} = \Delta^{\mathcal{I}}$
bottom	$\bot$	$\bot^{\mathcal{I}} = \emptyset$
negation	$\neg C$	$\Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$
conjunction	$C \sqcap E$ where $E \in \mathbf{A}$	$C^{\mathcal{I}} \cap E^{\mathcal{I}}$
disjunction	$C \sqcup E$ where $E \in \mathbf{A}$	$C^{\mathcal{I}} \cup E^{\mathcal{I}}$
exists restriction	$\exists R.C$	$\{x \mid (\exists y)[(x, y) \in R^{\mathcal{I}} \wedge y \in C^{\mathcal{I}}]\}$
value restriction	$\forall R.C$	$\{x \mid (\forall y)[(x, y) \in R^{\mathcal{I}} \rightarrow y \in C^{\mathcal{I}}]\}$
atleast restriction	$\geqslant nR$	$\{x \mid \#\{y \mid (x, y) \in R^{\mathcal{I}}\} \geqslant n\}$
atmost restriction	$\leqslant nR$	$\{x \mid \#\{y \mid (x, y) \in R^{\mathcal{I}}\} \leqslant n\}$
datatype exists restriction	$\exists U.D$	$\{x \mid (\exists y)[(x, y) \in U^{\mathcal{I}} \wedge y \in D^{\mathbf{D}}]\}$
datatype value restriction	$\forall U.D$	$\{x \mid (\forall y)[(x, y) \in U^{\mathcal{I}} \rightarrow y \in D^{\mathbf{D}}]\}$
datatype atleast restriction	$\geqslant nU$	$\{x \mid \#\{y \mid (x, y) \in U^{\mathcal{I}}\} \geqslant n\}$
datatype atmost restriction	$\leqslant nU$	$\{x \mid \#\{y \mid (x, y) \in U^{\mathcal{I}}\} \leqslant n\}$
Axiom	Syntax	Semantics
concept inclusion	$C \sqsubseteq E$	$C^{\mathcal{I}} \subseteq E^{\mathcal{I}}$
role inclusion	$R \sqsubseteq S$ where $R, S \in \mathbf{R}_A$ or $R, S \in \mathbf{R}_D$	$R^{\mathcal{I}} \subseteq S^{\mathcal{I}}$
transitivity	$trans(R)$	$R^{\mathcal{I}} = (R^{\mathcal{I}})^+$
concept membership	$C(a)$	$a^{\mathcal{I}} \in C^{\mathcal{I}}$
role membership	$R(a, b)$ where $b \in \mathbf{I}$ ($U(a, v)$ where v is a data value)	$(a^{\mathcal{I}}, b^{\mathcal{I}}) \in R^{\mathcal{I}}$ $((a^{\mathcal{I}}, v^{\mathbf{D}}) \in U^{\mathcal{I}})$
equality	$a = b (= (a, b))$	$a^{\mathcal{I}} = b^{\mathcal{I}}$
inequality	$a \neq b (\neq (a, b))$	$a^{\mathcal{I}} \neq b^{\mathcal{I}}$

dl-atoms, which helps to query the description logic knowledge base. A *dl-query* $Q(\mathbf{t})$ is either

1. an inclusion axiom F or its negation $\neg F$ ($\mathbf{t}$ is empty); or
2. a concept *C(t)* or its negation $\neg C(t)$ ($\mathbf{t}$ is t); or
3. a role $R(t_1, t_2)$ or its negation $\neg R(t_1, t_2)$ where $t_1$and t_2 are terms ($\mathbf{t}$ is (t_1, t_2)); or
4. an equality axiom $(= (t_1, t_2))$ or inequality axiom $(\neq (t_1, t_2))$ where $t_1$and t_2 are terms ($\mathbf{t}$ is (t_1, t_2)).

A *dl-atom* has the form,

$$DL[S_1 op_1 p_1, \ldots, S_m op_m p_m; Q](\mathbf{t}), m \geqslant 1$$

where each S_i is either a concept or role, $op_i = \{\uplus, \cup\!\!\!\!-\}$, and $p_1, \ldots, p_m$ are called input predicate symbols. If S_i is a concept, then p_i is a unary predicate symbol; if S_i is a role, then p_i is a binary predicate symbol. $Q(\mathbf{t})$ is called a *dl-query*. $op_i = \uplus$ ($op_i = \cup\!\!\!\!-$) increases S_i ($\neg S_i$) by the extension of p_i. A rule is called a *dl-rule* if one of the atoms in the rule $\{l_1, \ldots, l_z\}$ is a *dl-atom*. A *dl-program* $KB = (L, P)$ consists of a description logic knowledge base L and a finite set of *dl-rules* P. Since we considered only positive logic program P, KB is referred to positive KB. In this thesis, we call positive *dl-programs* (KB) as *dl-programs*.

Fixed-point Semantics. Let I be a Herbrand Interpretation for the dl-program $KB(KB = (L, P))$. Let P^* be the ground instances of rules in P. Since P does not have function symbols, P^* is always finite. Then, $T_{KB}(I)$ (immediate consequence operator) is a Herbrand Interpretation that is given by:

$$T_{KB}(I) = \{l_0 \mid l_0 \leftarrow l_1, \ldots, l_z \text{ in } P^*, \text{ for all } l_i \text{ where } 1 \leq i \leq z, I \models_L l_i\}$$

An important observation is that l_i is either a ground atom or ground *dl-atom*. I is a model of l_i under L, denoted $I \models_L l_i$:

- if l_i is a ground atom, then $I \models_L l_i$ iff $l_i \in I$

- if l_i is a ground *dl-atom* $DL[\lambda; Q](\mathbf{c})$, where $\lambda = S_1 op_1 p_1, \ldots, S_m op_m p_m$, then $I \models_L l_i$ iff $L(I; \lambda) \models Q(\mathbf{c})$ where $L(I; \lambda) = L \cup \bigcup_{i=1}^{m} A_i(I)$ and, for $1 \leqslant i \leqslant m$,

$$A_i(I) = \begin{cases} \{S_i(\mathbf{e}) \mid p_i(\mathbf{e}) \in I\}, \text{ if } op_i = \uplus; \\ \\ \{S_i(\mathbf{e}) \mid p_i(\mathbf{e}) \in I\}, \text{ if } op_i = \uplus. \end{cases}$$

We say I is a model of a *dl-program* $KB = (L, P)$, denoted $I \models KB$, iff $I \models_L r$ for all $r \in P^*$. We say the KB is satisfiable (unsatisfiable) iff it has some (no) models.

It is easy to show that T_{KB} always possesses a least fixed-point. The least fixed-point is a minimal model for KB ($KB = (L, P)$). This model can also shown to be $T_{KB} \uparrow \omega$, where the ordinal power of T_{KB} is given by:

Definition 2. *For any ordinal α,*

$$T_{KB} \uparrow \alpha = \begin{cases} \emptyset \text{ if } \alpha = 0, \\ \\ T_{KB}(T_{KB} \uparrow (\alpha - 1)) \text{ if } \alpha \text{ is a successor ordinal,} \\ \\ \bigcup_{\beta<\alpha}(T_{KB} \uparrow \beta) \text{if } \alpha \text{ is a limit ordinal.} \end{cases}$$

Similar to logic programs, the following proposition is true for any *dl-program* KB.

Proposition 4. *For any dl-program KB, the upward closure ordinal of T_{KB} is finite, i.e. there is a number $n \geq 0$ such that $T_{KB} \uparrow n = T_{KB} \uparrow \omega$.*

Proof. *The proof is immediate from the fact that the Herbrand Base is finite.*

The following example is taken from [30, 31], and it is modified to the positive *dl-program*.

Example 3. *Consider $KB = (L, P)$, where $L = \{S \sqsubseteq C\}$ and P is as follows:*

$r(a) \leftarrow DL[S \uplus q; C](a)$; $q(a) \leftarrow p(a)$; $p(a) \leftarrow$

Solution. For $I = \emptyset$, $T_{KB}(I) = \{p(a)\}$. For the second iteration, $I = \{p(a)\}$. Then, $T_{KB}(I) = \{p(a), q(a)\}$. For the third iteration, $T_{KB}(I) = \{p(a), q(a), r(a)\}$. In third iteration, the concept S was extended with a. Now L contains $S(a)$, by *modus ponens*, we say $C(a)$. Hence, the *dl-query* $(C(a))$ is true. Therefore, $r(a)$ is true.

CHAPTER 4

QC MODEL FOR POSITIVE EXTENDED DISJUNCTIVE DEDUCTIVE DATABASES

4.1 Introduction

The paraconsistent relation portrays a belief system rather than a knowledge system. The key idea of QC logic is given by the resolution rule of inference, which computes the focused belief. If the assumptions are considered as beliefs for the resolution, then the resolvent is called the focused belief. This ensures non-trivial reasoning in QC logic. As an individual can be both true and false for a given relation in the relational model, we decouple the link during the model construction. This is accomplished with the help of $FOCUS_D$ and $FOCUS_C$.

$FOCUS_D$**.** Let DR be a disjunctive relation on scheme 2^{Σ} and MR be a set of relations. Then

$FOCUS_D(DR, MR) = \{T \mid \forall T \in DISJ(DR) \wedge \exists t \in T \wedge \exists R \in MR \wedge Att(R) = Att(NRelation(t)) \wedge (NRelation(t) \text{ is positive } \wedge t \in R^- \rightarrow (T = T \setminus t)) \vee (NRelation(t) \text{ is negative } \wedge t \in R^+ \rightarrow (T = T \setminus t))))\}$

As a special case, for a given tuple T where $T \in DR^+$, if $FOCUS_D$ removes every element t in tuple T, then we convert the tuple T into a conjunction of the elements in the tuple. This is similar to $focus$ that we defined in the Preliminaries section.

CONJ. Let DR be a disjunctive relation that is mapped from $R_1 \dot{\cup} \ldots \dot{\cup} R_n$. For any $T \in DR$,

$CONJ(T) := \{\, t_1 \wedge \cdots \wedge t_n \mid \forall t_i \in T \wedge n \leq |T| \,\}$

Using **CONJ**, we define $FOCUS_C$.

$FOCUS_C$**.** Let DR be a disjunctive relation on scheme 2^{Σ} and MR be a set of relations. Then

$FOCUS_C(DR, MR) = \{CONJ(T) \mid \forall T \in DR^+ \wedge \forall t \in T \wedge \exists R \in MR \wedge Att(R) = Att(NRelation(t)) \wedge ((NRelation(t) \text{ is positive } \wedge t \in R^-) \vee (NRelation(t) \text{ is negative } \wedge t \in R^+))\}$

$FOCUS_D$ removes any element t , where $t \in T$ and $T \in DR$, that satisfies the predicate of $FOCUS_D$. Similarly, $FOCUS_C$ introduces conjunction among every $t \in T$, where $T \in DR$, that satisfies the predicate of $FOCUS_C$. In any *DR*, any tuple T that contains conjunction should never be affected by $FOCUS_D$.

To reiterate, DR^+ contains tuples which in turn can contain disjunction. From the base *DR*, multiple *DR* can be obtained by applying disjunction in tuples. Each newly created *DR* from the base *DR* should not lose any tuple set; otherwise, it leads to incorrect models. The following definition addresses the issue.

Proper Disjunctive Relation (PDR). Let *DR* be a base disjunctive relation. A proper disjunctive relation is a set, which contains all disjunctive relations that can be formed from *DR* by applying disjunction in tuples. Concretely, for every disjunctive relation (DR_i), which is obtained from *DR* by applying disjunction, $\tau(DR^+) = \tau(DR_i^+)$ where $1 \leq \mathrm{i} \leq (2^n - 1)^{\tau(DR^+)}$ such DR_i is a PDR^i.

To individualize the relation, we have the following definition.

Relationalize. Let $R_1 \dot{\cup} \ldots \dot{\cup} R_n$ and $R_1, \ldots, R_n$ be relations on scheme Σ.

$Relationalize(\dot{\pi}_{\{\Sigma\}}(R_1 \dot{\cup} \ldots \dot{\cup} R_n)[\Sigma]) :\ = \{R_1, \ldots, R_n\}$

The relationalize operator removes the unions among relations and the projection for it. By doing so, the operator produces a set of relations. If there is a select operation associated with the expression, then apply the operation before *Relationalize* is applied. *Relationalize* is in accordance to $Lits$, which is one of the key operators for finding the QC model [12].

During QC model construction, we encounter a set of redundant relation sets. In order to remove it, we define the following.

Minimize. Let $\{R1_1 \ldots R1_m\}$ and $\{R2_1 \ldots R2_n\}$ be two sets of relations where $m \leq n$.

$Minimize(\{\{R1_1 \ldots R1_n\}, \{R2_1 \ldots R2_m\}\}) :\ = \{\{R1_1 \ldots R1_m\} \mid R1_i = R2_j \wedge Att(R1_i) = Att(R2_j) \wedge \tau(R1_i) = \tau(R2_j)$ such that $\forall i, 1 \leq i \leq m \wedge \exists j, 1 \leq j \leq n\}$

4.2 Algorithm

By using the algebra of the relational model, we present a bottom up method for constructing the QC model for the positive extended disjunctive deductive database. The algorithm that we present in this section is an extension of the algorithm proposed by Bagai and Sunderraman [19]. The reader is requested to refer to QC logic programs [12] and QC logic [13]. The QC model's construction involves two steps. The first step is to convert P into a set of relation definitions for the predicate symbols occuring in P. These definitions are of the form

$$U_r = D_{U_r}$$

where U_r is the paraconsistent union of the disjunctive head predicate symbols of P, and D_{U_r} is an algebraic expression involving predicate symbols of P . Here r refers to the equation number, $1 \leq r \leq$ N, where N refers to a total number of equations. The second step is to iteratively evaluate the expressions in these definitions to incrementally construct the relations associated with the predicate symbols. The first step is called SERIALIZE and the second step is called Model Construction.

Algorithm. SERIALIZE

Input. A positive extended disjunctive deductive database clause $l_0 \vee \cdots \vee l_n \leftarrow l_{n+1} \ldots l_m$. For any i, $0 \leq i \leq m$, l_i is either of the form $p_i(A_{i1} \ldots A_{ik_i})$ or $\neg p_i(A_{i1} \ldots A_{ik_i})$. Let V_i be the set of all variables occurring in l_i

Output. An algebraic expression involving paraconsistent relations.

Method. The expression is constructed by the following steps:

1. For each argument A_{ij} of literal l_i, construct argument B_{ij} and condition C_{ij} as follows:

   (a) If A_{ij} is a constant a, then B_{ij} is any brand new variable and C_{ij} is $B_{ij}{=}a$.

   (b) If A_{ij} is a variable, such that for each k, $1 \leq k < j$, $A_{ik} \neq A_{ij}$, then B_{ij} is A_{ij} and C_{ij} is true.

   (c) If A_{ij} is a variable, such that for some k, $1 \leq k < j$, $A_{ik}{=}A_{ij}$, then B_{ij} is a brand new variable and C_{ij} is $A_{ij} = B_{ij}$.

2. Let $\hat{l}_i$ be the atom $p_i(B_{i1} \ldots B_{ik_i})$, and F_i be the conjunction $C_{i1} \wedge \cdots \wedge C_{ik_i}$. If l_i is a positive literal, then Q_i is the expression $\dot{\pi}_{V_i} \dot{\sigma}_{F_i}(\hat{l}_i))$. Otherwise, let Q_i be the expression $\dot{-}\dot{\pi}_{V_i}(\dot{\sigma}_{F_i}(\hat{l}_i))$.

   As a syntatic optimisation, if all conjuncts of F_i are true (i.e. all arguments of l_i are distinct variables), then both $\dot{\sigma}_{F_i}$ and $\dot{\pi}_{V_i}$ are reduced to identity operations, and are hence dropped from the expression.

3. Let U be the union ($\dot{\cup}$) of the Q_i's thus obtained, $0 \leq i \leq n$. The output expression is $(\dot{\sigma}_{F_1}(\dot{\pi}_{DV}(U)))\ [B_{01} \ldots B_{n_{kn}}]$ where DV is the set of distinct variables occurring in all l_i.

4. Let E be the natural join ($\dot{\bowtie}$) of the Q_i's thus obtained, $n+1 \leq i \leq m$. The output expression is $(\dot{\sigma}_{F_1}(\dot{\pi}_{DV}(E)))\ [B_{01} \ldots B_{n_{kn}}]$. As in step 2, if all conjuncts are true, then $\dot{\sigma}_{F1}$ is dropped from the output expression.

From the algebraic expression of the algorithm, we construct a system of equations.

For any positive extended disjunctive deductive database P, EQN (P) is a set of all equations of the form $U_r = D_{U_r}$, where U_r is a union of the head predicate symbols of P, and D_{U_r} is the union $\dot{\cup}$ of all expressions obtained by the algorithm *SERIALIZE* for clauses in P with the same U_r in their head. If all literals in the head are the same for any two rules, then U_r is the same for those two rules.

The final step is then to construct the model by incrementally constructing the relation values in P. For any positive extended disjunctive deductive database, P_E are the non disjunctive-facts (clauses in P without bodies), and P_B are the disjunctive rules (clauses in P with bodies). P_E^* refers to a set of all ground instances of clauses in P_E. Then, $P_I = P_E^* \cup P_B$.

The following algorithm finds the QC model for P.

ALGORITHM. Model Construction

Input. A positive extended disjunctive deductive database (P)

Output. Minimal QC Model for P.

Method: The values are computed by the following steps.

1. (Initialization)

   (a) Compute EQN(P_I) using the algorithm *SERIALIZE* for each clause in P_I.

   (b) SModel= $\emptyset$, For each predicate symbol p in P_E, set

   $p^+ = \{(a1 \ldots ak) \mid p(a1 \ldots, ak) \in P_E^*\}$, and $p^- = \emptyset$ or

   $p^- = \{(a1 \ldots ak) \mid \neg p(a1, \ldots ak) \in P_E^*\}$ and $p^+ = \emptyset$

SModel=p

End for.

2. (Rule Application)

   (a) DModel= $\emptyset$.

   For every SModel (SModel $\neq \emptyset$), create copies of the relations in SModel and replace the SModel with the copies.

   (b) For every equation r of the form $U_r = D_{U_r}$, create DR_r and insert the tuples from the copies in SModel into the corresponding exact relation in the equation r. Then map the definite tuples for the relations in U_r to DR_r. Compute the expression D_{U_r} and set the relations in U_r with $D^+_{U_r}$.

   (c) Map the newly added tuples of U_r to DR_r. Apply Θ and Ω to every relation in U_r. Also apply Θ to every relation in SModel. Then

$$DR_r = FOCUS_C(DR_r, SModel)$$

$$DR_r = FOCUS_D(DR_r, SModel)$$

   Repeat $FOCUS_D$ until there is no change in DR_r. When there is no change is DR_r, apply Θ to every relation in SModel and apply Ω and Θ to every relation in U_r.

   (d) Create a set of proper disjunctive relations (PDR_r) from the focused DR_r.

   (e) Delete all tuples for the relations in U_r and create multiple replicas of U_r, which is denoted by the set C_r, where $|C_r| = |PDR_r|$.

   (f) Re-map each p in PDR_r to C where $C \in C_r$.

   For every $C \in C_r$,

   C= *Relationalize*(C)

   /* C_r contains a collection of set of relations. */

DModel = DModel $\bigcup C_r$

/* Merging relations of every equation */

(g) Once all equations are evaluated for the current SModel, perform the following: i) for every $M \in$ DModel and for every exact relation for SModel that is not in M, create the exact relation in M, and ii) for every $M \in$ DModel and for every exact relation for SModel that is in M, insert the tuples from the copy relation in SModel into the exact relation. Then add DModel to TempModel.

(h) Once every SModel is applied, start from step 2 (a) with SModel=*Minimize* (TempModel) and stop when there is no change in SModel.

3. Minimal QC Model: Pick one (many) set (s) in SModel whose sum of the size of all relations in the set (s) is (are) minimal.

It is very intuitive from the algorithm that if the computation of D_{U_r} is empty for any SModel, then discard the SModel. We found that the algorithm should be extended a little to accommodate disjunctive facts, duplicate variables in disjunctive literals, and constants in disjunctive literals.

4.3 Example

Example 4. *Let P be a positive extended disjunctive deductive database. It has the following facts and rules*:

$r(a,c), p(a), p(c), \neg f(a,b), s(c)$

$w(X) \vee g(X) \vee \neg p(X) \leftarrow r(X,Y), s(Y)$

$w(X) \vee g(X) \vee \neg p(X) \leftarrow \neg f(X,Y)$

Solution. By step 1 (a) in initialization,

$w(X) \vee g(X) \vee \neg p(X) \leftarrow r(X,Y), s(Y)$ is serialized to

$(\dot{\pi}_{\{X\}}(w(X) \dot{\cup} g(X) \dot{\cup} \dot{-} p(X))[X] = (\dot{\pi}_{\{X\}}(r(X,Y) \dot{\bowtie} s(Y)))^{+}[X]$

and $w(X) \vee g(X) \vee \neg p(X) \leftarrow \neg f(X,Y)$ is serialized to

$(\dot{\pi}_{\{X\}}(w(X) \dot{\cup} g(X) \dot{\cup} \dot{-} p(X))[X] = (\dot{\pi}_{\{X\}}(\dot{-} f(X,Y)))^{+}[X]$

Both equations that are obtained after serialization have the same left-hand side expression. So, it is written as one equation (as show in (1)). EQN(P_I) returns:

1. $(\dot{\pi}_{\{X\}}(w(X)\dot{\cup}g(X)\dot{\cup}\dot{-}p(X))[X] = (\dot{\pi}_{\{X\}}(r(X,Y)\dot{\bowtie}s(Y)))^{+}[X]\dot{\cup}(\dot{\pi}_{\{X\}}(\dot{-}f(X,Y)))^{+}[X]$

After step 1 (b) in initialization, SModel = $\{r, p, s, f\}$ where

$r =$

$\{X, Y\}$
(a, c)

$p =$

$\{X\}$
(a)
(c)

$s =$

$\{Y\}$
(c)

$f =$

$\{X, Y\}$
(a, b)

After step 2(a), SModel = $\{r', p', s', f'\}$ (COPIES) where

$r' =$

$\{X, Y\}$
(a, c)

$p' =$

$\{X\}$
(a)
(c)

$s' =$

$\{Y\}$
(c)

$f' =$

$\{X, Y\}$
(a, b)

In step 2 (b), there is only one SModel and an equation. It is necessary to insert the tuples from the copies in SModel to the corresponding relations in the equation. DModel= $\emptyset$. Then map the definite tuples to DR_1 for the current SModel.

$DR_1 =$

$\{w.X\}$	$\{g.X\}$	$\{p.X\}$
		(a)
		(c)

Compute the equation and assign it to U_1. Map the newly added (disjunctive) tuples to DR_1.

$DR_1 =$

$\{w.X\}$	$\{g.X\}$	$\{p.X\}$
(a) $\vee$	(a) $\vee$	(a)
		(a)
		(c)

By step 2 (c), $DR_1 = FOCUS_D(DR_1, SModel)$

$DR_1 =$

$\{w.X\}$	$\{g.X\}$	$\{p.X\}$
$(a) \vee (a)$		
		(a)
		(c)

By step 2 (d), $PDR_1 = \{PDR_1^1, PDR_1^2, PDR_1^3\}$

$PDR_1^1 =$

$\{w.X\}$	$\{g.X\}$	$\{p.X\}$
(a)		
		(a)
		(c)

$PDR_1^2 =$

$\{w.X\}$	$\{g.X\}$	$\{p.X\}$
	(a)	
		(a)
		(c)

$PDR_1^3 =$

$\{w.X\}$	$\{g.X\}$	$\{p.X\}$
$(a) \vee (a)$		
		(a)
		(c)

Map every p in PDR_1 back to a set of base relations. We skip a step (2 (d)) here. After relationalizing the set of relations (step 2 (f)), we write:

$C_1 = \{\{w, p\}^1, \{g, p\}^2, \{w, g, p\}^3\}$

$\{w, p\}^1$

$w =$

$\{X\}$
(a)

$p =$

$\{X\}$
(a)
(c)

$\{g, p\}^2$

$g =$

$\{X\}$
(a)

$p =$

$\{X\}$
(a)
(c)

$\{\ w, g, p\ \}^3$

$$w = \begin{array}{|c|}\hline \{X\} \\ \hline (a) \\ \hline\hline \end{array}\ g = \begin{array}{|c|}\hline \{X\} \\ \hline (a) \\ \hline\hline \end{array}\ p = \begin{array}{|c|}\hline \{X\} \\ \hline (a) \\ \hline (c) \\ \hline\hline \end{array}$$

DModel= DModel $\bigcup C_1$

By step 2 (g),

DModel = $\{\{w, p, r, s, f\}^1, \{g, p, r, s, f\}^2, \{w, g, p, r, s, f\ \}^3\}$

$\{\ w, p, r, s, f\ \}^1$

$$w = \begin{array}{|c|}\hline \{X\} \\ \hline (a) \\ \hline\hline \end{array}\ p = \begin{array}{|c|}\hline \{X\} \\ \hline (a) \\ \hline (c) \\ \hline\hline \end{array}\ r = \begin{array}{|c|}\hline \{X, Y\} \\ \hline (a, c) \\ \hline\hline \end{array}\ s = \begin{array}{|c|}\hline \{Y\} \\ \hline (c) \\ \hline\hline \end{array}\ f = \begin{array}{|c|}\hline \{X, Y\} \\ \hline\hline (a, b) \\ \hline \end{array}$$

$\{\ g, p, r, s, f\ \}^2$

$$g = \begin{array}{|c|}\hline \{X\} \\ \hline (a) \\ \hline\hline \end{array}\ p = \begin{array}{|c|}\hline \{X\} \\ \hline (a) \\ \hline (c) \\ \hline\hline \end{array}\ r = \begin{array}{|c|}\hline \{X, Y\} \\ \hline (a, c) \\ \hline\hline \end{array}\ s = \begin{array}{|c|}\hline \{Y\} \\ \hline (c) \\ \hline\hline \end{array}\ f = \begin{array}{|c|}\hline \{X, Y\} \\ \hline\hline (a, b) \\ \hline \end{array}$$

$\{\ w, g, p, r, s, f\ \}^3$

$$w = \begin{array}{|c|}\hline \{X\} \\ \hline (a) \\ \hline\hline \end{array}\ g = \begin{array}{|c|}\hline \{X\} \\ \hline (a) \\ \hline\hline \end{array}\ p = \begin{array}{|c|}\hline \{X\} \\ \hline (a) \\ \hline (c) \\ \hline\hline \end{array}\ r = \begin{array}{|c|}\hline \{X, Y\} \\ \hline (a, c) \\ \hline\hline \end{array}\ s = \begin{array}{|c|}\hline \{Y\} \\ \hline (c) \\ \hline\hline \end{array}\ f = \begin{array}{|c|}\hline \{X, Y\} \\ \hline\hline (a, b) \\ \hline \end{array}$$

Add DModel to TempModel.

By step 2 (h), SModel=*Minimize* (TempModel)

The algorithm stops when there is no change in SModel. We then skip further iterations and write the final result:

Minimal QC Model = $\{\ \{\ w, p, r, s, f\}^1, \{g, p, r, s, f\}^2\ \}$

$\{\ w, p, r, s, f\ \}^1$

$$w = \begin{array}{|c|}\hline \{X\} \\ \hline (a) \\ \hline\hline \end{array}\ p = \begin{array}{|c|}\hline \{X\} \\ \hline (a) \\ \hline (c) \\ \hline\hline \end{array}\ r = \begin{array}{|c|}\hline \{X, Y\} \\ \hline (a, c) \\ \hline\hline \end{array}\ s = \begin{array}{|c|}\hline \{Y\} \\ \hline (c) \\ \hline\hline \end{array}\ f = \begin{array}{|c|}\hline \{X, Y\} \\ \hline\hline (a, b) \\ \hline \end{array}$$

$\{\ g, p, r, s, f\ \}^2$

$$g = \begin{array}{|c|}\hline \{X\} \\ \hline (a) \\ \hline\hline \end{array} \quad p = \begin{array}{|c|}\hline \{X\} \\ \hline (a) \\ \hline (c) \\ \hline\hline \end{array} \quad r = \begin{array}{|c|}\hline \{X, Y\} \\ \hline (a, c) \\ \hline\hline \end{array} \quad s = \begin{array}{|c|}\hline \{Y\} \\ \hline (c) \\ \hline\hline \end{array}$$

$$f = \begin{array}{|c|}\hline \{X, Y\} \\ \hline \\ \hline (a, b) \\ \hline \end{array}$$

In other words, Minimal QC Model = {

$\{w(a), p(a), p(c), r(a, c), s(c), \neg f(a, b)\}$,

$\{g(a), p(a), p(c), r(a, c), s(c), \neg f(a, b)\}$

}

Gelfond and Lifschitz adopt the way of trivializing results [6] while the algorithm tolerates inconsistencies. However, we observe that we have not proven the CORRECTNESS of the algorithm. Our immediate future work is to prove that the algorithm mimics fixed point semantics (Proposition 1 and Proposition 2) [12].

CHAPTER 5

CONSTRUCTION OF P-MINIMAL MODELS USING PARACONSISTENT RELATIONAL MODEL

5.1 Introduction

In this chapter, we borrow operators from Chapter 4 and construct p-minimal for positive extended disjunctive deductive database. However, fixed-point semantics for p-minimal model is different from QC models. We first define p-minimal model and construct the model in the extended database setting.

Given a first order language $\mathcal{L}$, a disjunctive deductive database P [22] consists of logical inference rules of the form: r (rule) $= l_0 \vee \cdots \vee l_n \leftarrow l_{n+1}, \ldots, l_m$. A rule is called a positive disjunctive rule if the rule has both head (disjunction of literals) and body (conjunction of literals). Concretely, the rule r is called positive extended disjunctive rule if $l_0, \ldots, l_n, l_{n+1}, \ldots, l_m$ are literals which are either positive or negative ($\neg$) atoms. For the given syntax of positive extended disjunctive deductive databases, we reproduce the fixed point semantics of P [7].

Fixed-point Semantics. Let P be a positive extended disjunctive deductive database and $\mathcal{I}$ be a set of interpretations, then $\mathcal{T}_P(\mathcal{I}) = \bigcup_{I \in \mathcal{I}} T_P(I)$

$$T_P(I) = \begin{cases} \emptyset, \text{ if } l_{n+1}, \ldots, l_m \subseteq I \text{ for some} \\ \text{ ground constraint } \leftarrow l_{n+1} \ldots l_m \text{from } P; \\ \{J \mid \text{for each ground clause} \\ \quad r_i \colon l_0 \vee \cdots \vee l_n \leftarrow l_{n+1}, \ldots, l_m \text{ such that} \\ \quad \{l_{n+1}, \ldots, l_m\} \subseteq I, J = I \cup \bigcup_{r_i} \{l_j\} (1 \le j \le n)\}, \text{otherwise.} \end{cases}$$

In the definition of $T_P(I)$, $\{l_j\}(1 \le j \le n)$ is a collection of sets where every set in the collection contains a disjunct. For any positive extended disjunctive deductive database P, $\mathcal{T}_P$ is finite and

$\mathcal{T}_P \uparrow$ n = $\mathcal{T}_P \uparrow \omega$ where n is a successor ordinal and ω is a limit ordinal. For any positive extended disjunctive deductive database P, p-minimal models = $min(\mu(\mathcal{T}_P \uparrow \omega)$[1]$)$ where $min(\mathcal{I}) = \{I \in \mathcal{I} \mid \nexists J \in \mathcal{I}$ such that $J \subset I\}$.

5.2 Algorithm

In this section, we serialize the clauses into equations and form a system of equation, which is very similar to section 4.2.

Algorithm. SERIALIZE

Input. A positive extended disjunctive deductive database clause $l_0 \vee \cdots \vee l_n \leftarrow l_{n+1} \ldots l_m$. For any i, $0 \leq i \leq m$, l_i is either of the form $p_i(A_{i1} \ldots A_{ik_i})$ or $\neg p_i(A_{i1} \ldots A_{ik_i})$, and let V_i be the set of all variables occurring in l_i.

Output. An algebraic expression involving paraconsistent relations.

Method. The expression is constructed by the following steps:

1. For each argument A_{ij} of literal l_i, construct argument B_{ij} and condition C_{ij} as follows:

   (a) If A_{ij} is a constant a, then B_{ij} is any brand new variable and C_{ij} is B_{ij}=a.

   (b) If A_{ij} is a variable, such that for each k, $1 \leq k < j$, $A_{ik} \neq A_{ij}$, then B_{ij} is A_{ij} and C_{ij} is true.

   (c) If A_{ij} is a variable, such that for some k, $1 \leq k < j$, A_{ik}=A_{ij}, then B_{ij} is a brand new variable and C_{ij} is $A_{ij} = B_{ij}$.

2. Let $\hat{l}_i$ be the atom $p_i(B_{i1} \ldots B_{ik_i})$, and F_i be the conjunction $C_{i1} \wedge \cdots \wedge C_{ik_i}$. If l_i is a positive literal, then Q_i is the expression $\dot{\pi}_{V_i}\dot{\sigma}_{F_i}(\hat{l}_i))$. Otherwise, let Q_i be the expression $\dot{-}\dot{\pi}_{V_i}(\dot{\sigma}_{F_i}(\hat{l}_i))$.

   As a syntatic optimisation, if all conjuncts of F_i are true (i.e. all arguments of l_i are distinct variables), then both $\dot{\sigma}_{F_i}$ and $\dot{\pi}_{V_i}$ are reduced to identity operations, and are hence dropped from the expression $\dot{\sigma}_{F_i}$.

---

[1] $\mu(\mathcal{T}_P \uparrow \omega) = \{I \mid I \in \mathcal{T}_P \uparrow \omega$ and $I \in \mathcal{T}_{\mathcal{P}}(I)\}$

3. Let U be the union ($\dot{\cup}$) of the Q_i's thus obtained, $0 \leq i \leq n$. The output expression is $(\dot{\sigma}_{F_1}(\dot{\pi}_{DV}(U)))\ [B_{01} \ldots B_{n_{kn}}]$ where DV is the set of distinct variables occurring in all l_i.

4. Let E be the natural join ($\bowtie$) of the Q_i's thus obtained, $n+1 \leq i \leq m$. The output expression is $(\dot{\sigma}_{F_1}(\dot{\pi}_{DV}(E)))\ [B_{01} \ldots B_{n_{kn}}]$. As in step 2, if all conjuncts are true, then $\dot{\sigma}_{F1}$ is dropped from the output expression.

From the algebraic expression of the algorithm, we then construct a system of equations.

For any positive extended disjunctive deductive database P, EQN (P) is a set of all equations of the form $U_r = D_{U_r}$, where U_r is a union of the head predicate symbols of P, and D_{U_r} is the paraconsistent union ($\dot{\cup}$) of all expressions obtained by the algorithm *SERIALIZE* for clauses in P with the same U_r in their head. If all literals in the head are the same for any two rules, then U_r is the same for the two rules.

The final step is then to construct the model by incrementally constructing the relation values in P. For any positive extended disjunctive deductive database, P_E is the non disjunctive-facts (clauses in P without bodies), and P_B is the disjunctive rules (clauses in P with bodies). P_E^* refers to a set of all ground instances of clauses in P_E. Then, $P_I = P_E^* \cup P_B$.

The following algorithm finds p-minimal models for P.

ALGORITHM. MODEL CONSTRUCTION

Input. A positive extended disjunctive deductive database (P)

Output. P-minimal models for P.

Method: The values are computed by the following steps.

1. (Initialization)

   (a) Compute EQN(P_I) using the algorithm *SERIALIZE* for each clause in P_I.

   (b) SModel= $\emptyset$, For each predicate symbol p in P_E, set

   $p^+ = \{(a1 \ldots ak) \mid p(a1 \ldots, ak) \in P_E^*\}$, and $p^- = \emptyset$ or

   $p^- = \{(a1 \ldots ak) \mid \neg p(a1, \ldots ak) \in P_E^*\}$ and $p^+ = \emptyset$

   SModel=p

End for.

2. (Rule Application)

(a) For every SModel (SModel $\neq \emptyset$), create copies of the relations in SModel and replace the SModel with the copies. DModel= $\emptyset$.

(b) For every equation r of the form $U_r = D_{U_r}$, create DR_r and insert the tuples from the copies in SModel into the corresponding exact relation in the equation r. Apply Θ to every relation in U_r and map the definite tuples for the relations in U_r to DR_r. Again, apply Θ to every relation in U_r. Compute the expression D_{U_r} and set the relations in U_r with $D^+_{U_r}$.

(c) Apply Θ to every relation in U_r, map the newly added tuples of U_r to DR_r and create a set of proper disjunctive relations (PDR_r) from the DR_r.

(d) Delete all tuples for the relations in U_r and create multiple replicas of U_r, which is denoted by the set C_r, where $|C_r| = |PDR_r|$.

(e) Re-map each p in PDR_r to C where $C \in C_r$.

For every $C \in C_r$,

C= *Relationalize*(C)

For every $R \in C$

$R = \Theta(R)$

End For.

End For.

DModel = DModel $\bigcup C_r$/* Merging relations of every equation */

(f) Once all equations are evaluated for the current SModel, perform the following: i) for every $M \in$ DModel and for every exact relation for SModel that is not in M, create the exact relation in M; and ii) for every $M \in$ DModel and for every exact relation for SModel that is in M, insert the tuples from the copy relation in SModel into the exact relation of M. Then add DModel to TempModel.

(g) Once every SModel is applied, start from step 2 (a) with

SModel= *Minimize*(TempModel) and stop when there is no change in SModel.

3. P-models: rewrite the set of relations in SModel as a set of literals. P-minimal models= *min*(P-models) (*min*() is defined in Preliminaries).

It is very intuitive from the algorithm that if the computation of D_{U_r} is empty for any SModel, then discard the SModel. We found that the algorithm should be extended a little to accommodate for disjunctive facts, duplicate variables in disjunctive literals, and constants in disjunctive literals.

5.3 Example

The following example shows that how the algorithm works.

Example 5. *Let P be a positive extended disjunctive deductive database. It has the following facts and rules:*

r(a,c), p(a), p(c), ¬f(a,b), s(c)

g(X) ∨¬ p(X) ← r(X,Y), s(Y)

g(X) ∨¬ p(X) ← ¬f(X,Y)

Solution. After step 1 (a) in initialization, EQN(P_I) returns:

$(U_1)(\dot{\pi}_{\{X\}}(g(X)\dot{\cup}\dot{-}p(X))[X]=$

$(\dot{\pi}_{\{X\}}(r(X,Y)\dot{\bowtie}s(Y)))^{+}[X]\dot{\cup}(\dot{\pi}_{\{X\}}(\dot{-}f(X,Y)))^{+}[X]$

After step 1 (b) in initialization, SModel = $\{r, p, s, f\}$ where

$$r\ \begin{array}{|c|}\hline \{X,Y\} \\ \hline (a,c) \\ \hline \end{array}\quad p\ \begin{array}{|c|}\hline \{X\} \\ \hline (a) \\ \hline (c) \\ \hline \end{array}\quad s\ \begin{array}{|c|}\hline \{Y\} \\ \hline (c) \\ \hline \end{array}\quad f\ \begin{array}{|c|}\hline \{X,Y\} \\ \hline (a,b) \\ \hline \end{array}$$

After step 2 (a), SModel = $\{r', p', s', f'\}$ (COPIES) where

$$r'\ \begin{array}{|c|}\hline \{X,Y\} \\ \hline (a,c) \\ \hline \end{array}\quad p'\ \begin{array}{|c|}\hline \{X\} \\ \hline (a) \\ \hline (c) \\ \hline \end{array}\quad s'\ \begin{array}{|c|}\hline \{Y\} \\ \hline (c) \\ \hline \end{array}\quad f'\ \begin{array}{|c|}\hline \{X,Y\} \\ \hline (a,b) \\ \hline \end{array}$$

In step 2 (b), there is only one SModel and one equation. It is necessary to insert the tuples from the copies in SModel to the corresponding relations in the equation. DModel= $\emptyset$. Then map the definite tuples to DR_1 for the current SModel. Compute the expression and assign it to U_1.

DR_1

$\{g.X\}$	$\{p.X\}$
	(a)
	(c)

By step 2 (c), map the newly added (disjunctive) tuples to DR_1.

DR_1

$\{g.X\}$	$\{p.X\}$
(a) $\vee$	(a)
	(a)
	(c)

$PDR_1 = \{PDR_1^1, PDR_1^2, PDR_1^3\}$

PDR_1^1

$\{g.X\}$	$\{p.X\}$
(a) $\vee$	(a)
	(a)
	(c)

PDR_1^2

$\{g.X\}$	$\{p.X\}$
	(a)
	(a)
	(c)

PDR_1^3

$\{g.X\}$	$\{p.X\}$
(a)	
	(a)
	(c)

We skip a step (2 (d)) here. Map every p in PDR_1 back to a set of base relation. We write after relationalizing the set of relations and applying Θ (step 2 (e)).

$C_1 = \{\{g, p\}^1, \{p\}^2, \{g, p\}^3\}$

$\{g, p\}^1$

g

$\{X\}$
(a)

p

$\{X\}$
(a)
(c)
(a)

$\{p\}^2$

p

$\{X\}$
(a)
(c)
(a)

$\{g, p\}^3$

g

$\{X\}$
(a)

p

$\{X\}$
(a)
(c)

DModel= DModel $\bigcup C_1$

By step 2 (f), DModel = $\{\{g, p, r, s, f\}^1, \{p, r, s, f\}^2, \{g, p, r, s, f\}^3\}$

$\{g, p, r, s, f\}^1$

g

$\{X\}$
(a)

p

$\{X\}$
(a)
(c)
(a)

r

$\{X, Y\}$
(a, c)

s

$\{Y\}$
(c)

f

$\{X, Y\}$
(a, b)

$$\{\, p, r, s, f \,\}^2 \; p \begin{array}{|c|}\hline \{X\} \\ \hline (a) \\ \hline (c) \\ \hline \hline (a) \\ \hline \end{array} \; r \begin{array}{|c|}\hline \{X, Y\} \\ \hline (a, c) \\ \hline \hline \end{array} \; s \begin{array}{|c|}\hline \{Y\} \\ \hline (c) \\ \hline \hline \end{array} \; f \begin{array}{|c|}\hline \{X, Y\} \\ \hline \hline (a, b) \\ \hline \end{array}$$

$$\{\, g, p, r, s, f \,\}^3 \; g \begin{array}{|c|}\hline \{X\} \\ \hline (a) \\ \hline \hline \end{array} \; p \begin{array}{|c|}\hline \{X\} \\ \hline (a) \\ \hline (c) \\ \hline \hline \end{array} \; r \begin{array}{|c|}\hline \{X, Y\} \\ \hline (a, c) \\ \hline \hline \end{array} \; s \begin{array}{|c|}\hline \{Y\} \\ \hline (c) \\ \hline \hline \end{array} \; f \begin{array}{|c|}\hline \{X, Y\} \\ \hline \hline (a, b) \\ \hline \end{array}$$

Add DModel to TempModel.

By step 2 (g), SModel=*Minimize*(TempModel). The algorithm stops when there is no change in SModel. We skip further iterations and go to the final step (3). In the final step, we first rewrite the relation in the form of literals,

P-models = $\{\{g(a), p(a), p(c), \neg p(a), r(a, c), s(c), \neg f(a, b)\},$

$\{p(a), p(c), \neg p(a), r(a, c), s(c), \neg f(a, b)\}, \{g(a), p(a), p(c), r(a, c), s(c), \neg f(a, b)\}\}.$

Then, p-minimal models = $\{\{p(a), p(c), \neg p(a), r(a, c), s(c), \neg f(a, b)\},$

$\{g(a), p(a), p(c), r(a, c), s(c), \neg f(a, b)\}\}.$

CHAPTER 6

HANDLING INCONSISTENT CLOSED PREDICATES: A PARACONSISTENT APPROACH

6.1 Introduction

The Semantic Web [34], which is an extension of the World Wide Web (WWW), adds metadata to the content in the Web so that the machine can interpret the content. The Semantic Web's vision is achieved with the help of ontologies, which formally represent the data so that software agents can understand the data. The Web Ontology Language (OWL) [35], which is a W3C recommendation standard, is an ontology language whose semantics are based on description logic [24].

Traditionally, the Semantic Web is based on OWA which does not allow inference of negative information based on non-provability. However in real world KBs, all predicates do not need to be opened at all times because some predicates are not changing. In other words, either some predicates that exist in the KBs are complete or some predicates are migrated from relational databases. For example, in earth KB, assume there are two predicates, Population and Country; Population can be changing, but not Country which could be closed.

As our motivating example, consider the following $\mathcal{ALC}$ KB. For the purpose of understanding, consider $LatinAmericanCountries$ are only $Mexico$ and $Brazil$.

Example 6. $KB = \{$

$LatinAmericanCountries \sqsubseteq \bot$

$LatinAmericanCountries(Mexico)$

$LatinAmericanCountries(Brazil)$

$RestaurantsInLatinAmerica(McDonalds)$

$RestaurantsInLatinAmerica(KFC)$ }.

In Example 6, $LatinAmericanCountries$ is a closed concept and $RestaurantsInLatinAmerica$ is an open concept. According to [23] and [26], an ABox is satisfiable with respect to a TBox and close predicates iff there is a model that satisfies the TBox, the closed predicates and the ABox. It is easy to observe that Example 6 has no models. However, consider a database containing the relation *LatinAmericanCountries* and the tuples such as *Mexico*. When a user queries the database, the user gets the answer. If the same data is migrated to KB, then the data becomes inconsistent because of some inconsistent axioms and querying such KBs is trivialized (anything can be a consequent of the KB). Hence, it is extremely vital to handle such inconsistencies for KBs with closed predicates.

In description logic without closed predicates, many works [36–39] have been proposed to handle inconsistencies. Particularly, we are very interested in the works [40] and [41], where the authors introduced four-valued description logic and transformed it to two-valued description logic. Then the transformed description logic is reasoned over standard reasoners. There is an advancement to this approach called quasi-classic description logic [42] whose inference power is stronger than four-valued description logic. As we are focused on handling inconsistencies in closed predicates rather than obtaining stronger inference, we choose four-valued description logic in our work. Specifically, we borrow some ideas from [43] to represent closed predicates in four-valued description logic.

6.2 Algorithm

The predicates in description logic are usually open because description logic is OWA. To introduce closed predicates, a new component is added to the TBox (T) which is Σ (a set of closed predicates). For any ABox $\mathcal{A}$, a model $\mathcal{I}$ of (T, Σ) and $\mathcal{A}$ is an interpretation $\mathcal{I}$ with Ind $(\mathcal{A})$[1] $\subseteq \Delta^{\mathcal{I}}$ that satisfies T and $\mathcal{A}$ and such that the extensions of all closed predicates satisfies the explicitly stated assertions in the ABox [23] and [26]. In other words,

$C^{\mathcal{I}}$={ $a \mid C(a) \in \mathcal{A}\} \; \forall C \in \Sigma \cap N_C$

$r^{\mathcal{I}}$={$(a, b) \mid r(a, b) \in \mathcal{A}\} \; \forall r \in \Sigma \cap N_R$

Let KB be (T, Σ) and A, and Q be a query. If, for every interpretation $\mathcal{I}$, $\mathcal{I} \models_2 KB$ and $\mathcal{I} \models_2 Q$, then $KB \models_2 Q$. It is important to note that Σ contains only atomic predicates and no closed

[1] Ind($\mathcal{A}$) refers to a set of all individuals in the ABox $\mathcal{A}$.

predicates have negation ($\neg$) in front of it in any KBs. In Table I, $C \in N_C \cap \Sigma$.

To handle inconsistencies associated with the closed predicates using four-valued logic, it is required to define the semantics for closed predicates in four-valued logic. The following section discusses the same.

6.2.1 Semantics of four-valued $\mathcal{ALC}$ with closed predicates

The four valued logic for $\mathcal{ALC}$ is created by assigning both positive P and negative N extensions for any open concept description. As four-valued $\mathcal{ALC}$ is based on Belnap's four valued logic, it has four truth values (true (1), false(0), nothing (n), both (b)). If $A \in N_C$, then $A(a)$ is:

- 1 if $a^{\mathcal{I}} \in P$ and $a^{\mathcal{I}} \notin N$
- 0 if $a^{\mathcal{I}} \notin P$ and $a^{\mathcal{I}} \in N$
- n if $a^{\mathcal{I}} \notin P$ and $a^{\mathcal{I}} \notin N$
- b if $a^{\mathcal{I}} \in P$ and $a^{\mathcal{I}} \in N$

If a predicate is a closed predicate, it should have 0 and 1 as truth values. Closed predicates are always considered to be complete and no new information can be inferred on it. In other words, closed predicates have classical semantics in four-valued semantics of $\mathcal{ALC}$. $\mathcal{ALC}$ does not have constructors like $\neg r(a, b)$. So, roles in four-valued $\mathcal{ALC}$ are considered to have classical semantics [41]. Based on [44] and [45], the authors [40, 43] and [41] introduced three types of different semantics to inclusions.

$A \longmapsto B$ (material inclusion)

$A \sqsubset B$ (internal inclusion)

$A \rightarrow B$ (strong inclusion)

Although all three inclusions are valid for four-valued $\mathcal{ALC}$, in this chapter we use only internal implication while we are working with closed predicates. Hence, we specify $A \sqsubseteq B$ as $A \sqsubset B$ for four-valued $\mathcal{ALC}$ with closed predicates. Table 6.1 shows the semantics of four-valued

Table (6.1) Semantics of four-valued $\mathcal{ALC}$ with closed predicates

Syntax	Name	Semantics
A	open concept	$A^{\mathcal{I}'}=\langle P,N\rangle$ where P and $N \subseteq \Delta^{\mathcal{I}'}$
C	closed concept	$C^{\mathcal{I}'} \subseteq \Delta^{\mathcal{I}'}$
r	open or closed role	$r^{\mathcal{I}'} \subseteq \Delta^{\mathcal{I}'} \times \Delta^{\mathcal{I}'}$
$\bot$	bottom	$\langle \emptyset, \Delta^{\mathcal{I}'}\rangle$
$\top$	top	$\langle \Delta^{\mathcal{I}'}, \emptyset\rangle$
$\neg A$	negation	$\neg A^{\mathcal{I}'} = \langle N,P\rangle$ if $A^{\mathcal{I}'} = \langle P,N\rangle$
$A \sqcap B$	conjunction	$\langle P_1 \cap P_2, N_1 \cup N_2\rangle$ if $A= \langle P_1,N_1\rangle$ and $B= \langle P_2,N_2\rangle$ where A and B are open concepts.
$A \sqcap C$	conjunction	$\langle P_1 \cap C^{\mathcal{I}'}, N_1\rangle$ if $A = \langle P_1,N_1\rangle$. where A is an open concept and C is a closed concept.
$A \sqcap C$	conjunction	$A^{\mathcal{I}'} \cap C^{\mathcal{I}'}$ where A and C are closed concepts.
$A \sqcup B$	disjunction	$\langle P_1 \cup P_2, N_1 \cap N_2\rangle$ if $A= \langle P_1,N_1\rangle$ and $B= \langle P_2,N_2\rangle$ where A and B are open concepts.
$A \sqcup C$	disjunction	$\langle P_1 \cup C^{\mathcal{I}'}, N_1\rangle$ if $A = \langle P_1,N_1\rangle$ where A is an open concept and C is a closed concept.
$A \sqcup C$	disjunction	$A^{\mathcal{I}'} \cup C^{\mathcal{I}'}$ if where A and C are open concepts.
$\exists r.A$	existential restrictions	$\langle\{x \mid \exists y \in \Delta^{\mathcal{I}'} : (x,y) \in r^{\mathcal{I}'}\ and\ y \in proj^{+}(A^{\mathcal{I}'})\}$, $\{x \mid \forall y \in \Delta^{\mathcal{I}'} : (x,y) \in r^{\mathcal{I}'} \rightarrow y \in proj^{-}(A^{\mathcal{I}'})\}\rangle$ where A is an open concept (This semantics work for both closed and open r).
$\exists r.C$	existential restrictions	$\{x \mid \exists y \in \Delta^{\mathcal{I}'} : (x,y) \in r^{\mathcal{I}'}\ and\ y \in C^{\mathcal{I}'}\}$ where C is a closed concept (This semantics work for both closed and open r).
$\forall r.A$	universal restrictions	$\langle\{x \mid \forall y \in \Delta^{\mathcal{I}'} : (x,y) \in r^{\mathcal{I}'} \rightarrow y \in proj^{+}(A^{\mathcal{I}'})\}$, $\{x \mid \exists y \in \Delta^{\mathcal{I}'} : (x,y) \in r^{\mathcal{I}'}\ and\ y \in proj^{-}(A^{\mathcal{I}'})\}\rangle$ where A is an open concept (This semantics work for both closed and open r).
$\forall r.C$	universal restrictions	$\{x \mid \forall y \in \Delta^{\mathcal{I}'} : (x,y) \in r^{\mathcal{I}'} \rightarrow y \in C^{\mathcal{I}'}\}$ where C is a closed concept (This semantics work for both closed and open r).
$C \sqsubset A$	internal inclusion	$C^{\mathcal{I}'} \subseteq proj^{+}(A^{\mathcal{I}'})$
$A(a)$	open concept assertion	$a^{\mathcal{I}'} \in proj^{+}(A^{\mathcal{I}'})$
$C(a)$	closed concept assertion	$a^{\mathcal{I}'} \in C^{\mathcal{I}'}$
$r(a,b)$	role assertion	$(a^{\mathcal{I}'}, b^{\mathcal{I}'}) \in (r^{\mathcal{I}'})$ r can be either open or closed

$\mathcal{ALC}$ with closed predicates. In Table 6.1, closing both antecedent and consequent of the internal inclusion axiom (inclusion) lead to the database integrity constraint [46].

The semantics of four-valued $\mathcal{ALC}$ with closed predicates are given by a four-valued interpretation $\mathcal{I}'= (\Delta^{\mathcal{I}'}, \cdot^{\mathcal{I}'})$ where $\Delta^{\mathcal{I}'}$ is a finite non-empty set abstract domain and $\cdot^{\mathcal{I}'}$ is a mapping where it assigns subsets of $(\Delta^{\mathcal{I}'})^2$ to open concepts, elements of $\Delta^{\mathcal{I}'}$ to individuals and elements of $\Delta^{\mathcal{I}'}$ to closed concepts such that condition in Table 6.2 is satisfied (the mapping for roles is similar to two-valued $\mathcal{ALC}$). If A is an open concept and $\mathcal{I}'$ is a four-valued interpretation such that $A^{\mathcal{I}'}=\langle P, N\rangle$, then $\text{proj}^{+}(A^{\mathcal{I}'})=P$ and $\text{proj}^{-}(A^{\mathcal{I}'})=N$. A four-valued model of the KB is defined as a four-valued interpretation which satisfies every axiom in the TBox, every assertion in the ABox and the extensions of closed predicates should agree on what is explicitly stated in the ABox. A KB is called satisfiable (unsatisfiable) iff there exists (does not exist) a model. KB entails ($\models_4$) the query Q iff every four-valued model of KB is a four-valued model of Q.

Definition 3. *Let C be a closed concept, A be an open concept and KB be a four-valued knowledge base of $\mathcal{ALC}$ with closed predicates.*

1. *C is two-valued satisfiable wrt the KB if there is a four-valued model $\mathcal{I}'$ of the KB such that $C^{\mathcal{I}'}$ is not empty. The open concept A is four-valued satisfiable when $proj^{+}(A^{\mathcal{I}'})$ is not empty.*

2. *C is subsumed by A wrt the KB if $C^{\mathcal{I}'} \subseteq proj^{+}(A^{\mathcal{I}'})$ in every four-valued model of $\mathcal{I}'$ in the KB. Similarly, A is subsumed by C can be defined.*

Proposition 5. *Let C be a closed concept and KB be a four-valued $\mathcal{ALC}$ knowledge base with closed predicates. C is two-valued unsatisfiable iff C is subsumed by $\perp$.*

Proposition 6. *For any closed concept C and open concept A in four-valued $\mathcal{ALC}$. C is subsumed by A iff $C \sqsubset A$.*

Proof. *The proof is immediate from Table 6.1 and Definition 3.*

Even though we have given the semantics of closed predicates in four-valued $\mathcal{ALC}$, it is necessary to show that it is sound with respect to two-valued $\mathcal{ALC}$ with closed predicates which is discussed in the following section. Moreover, the proofs of next two sections are similar to [43], but we analyze particularly in the presence of closed predicates, which is novel here. All of the propositions discussed in the next two sections are true for open predicates which are given in [43].

6.2.2 Four-valued $\mathcal{ALC}$ with Closed Predicates is Sound

There exists a correspondence between a two-valued interpretation $\mathcal{I}$ and a four valued interpretation $\mathcal{I}_c$ which is referred as correspondence to $\mathcal{I}$ in any KB with closed predicates. As the domain for $\mathcal{I}$ and $\mathcal{I}_c$ are the same, it is referred to as Δ. Then

- For any $a \in N_{\mathcal{I}}$, $a^{\mathcal{I}_C} = a^{\mathcal{I}}$
- For any $r \in N_r$, $r^{\mathcal{I}_C} = r^{\mathcal{I}}$
- For any $A \in N_C$, $A^{\mathcal{I}_C} = \langle A^{\mathcal{I}}, \Delta \setminus A^{\mathcal{I}} \rangle$

Proposition 7. *Let $\mathcal{I}$ be a two-valued interpretation and its correspondence is $\mathcal{I}_C$. Then for any concept closed concept $C \in \Sigma \cap N_C$ or closed role $r \in \Sigma \cap N_r$,*

$C^{\mathcal{I}_C} = C^{\mathcal{I}}$

$r^{\mathcal{I}_C} = r^{\mathcal{I}}$

Proof. *The semantics in Table 6.1 and Table 3.1 yield the proof trivially.*

Proposition 8. *If $\mathcal{I}$ is a two valued interpretation and $\mathcal{I}_C$ is its correspondence, then for any closed assertion C or r, $\mathcal{I}$ is a two valued model of $C(r)$ iff $\mathcal{I}_C$ is a two valued model of $C(r)$.*

Proof. *Assume C as $C(a)$. Then $\mathcal{I}_C$ is a two-valued model of $C(a)$ iff $a^{\mathcal{I}_C} \in C^{\mathcal{I}_C}$ iff $a^{\mathcal{I}} \in C^{\mathcal{I}}$ iff $\mathcal{I}$ is a two valued model of $C(a)$. Assume r as $r(a, b)$. Then $\mathcal{I}_C$ is a two valued model of $r(a, b)$ iff $(a^{\mathcal{I}_C}, b^{\mathcal{I}_C}) \in r^{\mathcal{I}_C}$ iff $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in r^{\mathcal{I}}$ iff $\mathcal{I}$ is a two valued model of $r(a, b)$.*

Proposition 9. *If $\mathcal{I}$ is a two-valued interpretation and $\mathcal{I}_C$ is its correspondence, then for any axiom Ax containing closed predicates, $\mathcal{I}$ is a two valued model of Ax iff $\mathcal{I}_C$ is a four valued model of Ax.*

Proof. *Consider Ax as $C \sqsubset A$ where $C \in \Sigma \cap N_C$ and $A \notin \Sigma$ but $A \in N_C$. Then $\mathcal{I}_C$ is four valued model of Ax iff $C^{\mathcal{I}_C} \subseteq proj^{+}(A^{\mathcal{I}_C})$ iff $C^{\mathcal{I}} \subseteq A^{\mathcal{I}}$ iff $\mathcal{I}$ is a two valued model of Ax.*

Proposition 8 considers only closed predicates. However, for any open predicates, $\mathcal{I}_C$ is a four-valued model iff $\mathcal{I}$ is a two-valued model [43]. Moreover, Proposition 9 is true whether the axioms contain closed predicates or not.

Hence, by proposition 8 and 9, the following proposition is true with closed predicates in the KB.

Proposition 10. *Let KB be a four-valued $\mathcal{ALC}$ and Ax be an axiom or assertion in four-valued $\mathcal{ALC}$. If $KB \models_4 Ax$, then $KB \models_2 Ax$.*

Proposition 11. *Let Q be a four-valued query and Q' be the same query Q but the representation is in two-valued logic. Let KB be a knowledge base of four-valued $\mathcal{ALC}$ with closed predicates and KB' be the knowledge base of two-valued $\mathcal{ALC}$ with closed predicates, which is obtained by replacing every occurrence of $\sqsubset$ with $\sqsubseteq$. If $KB \models_4 Q$, then $KB' \models_2 Q'$.*

Proof. *Suppose $KB \models_4 Q$ and $\mathcal{I}$ is a two valued model of KB'. Then, the correspondence of $\mathcal{I}$ is $\mathcal{I}_C$ satisfies KB' by Proposition 8, 9 and [43, Proposition 22]. In addition to that, $\mathcal{I}_C$ satisfies KB. $KB \models_4 Q$ leads to $\mathcal{I}_C \models_4 Q$. By Proposition 8, 9, and [43, Proposition 22], $\mathcal{I} \models_2 Q$. Since Q and Q' are semantically the same, $\mathcal{I} \models_2 Q'$ is true.*

In section 6.2.1 and 6.2.2, we introduced closed predicates in four-valued $\mathcal{ALC}$ and showed that it is sound. In the following section, we handle the inconsistency associated with closed predicates and transform it to two-valued $\mathcal{ALC}$ with closed predicates.

6.2.3 Four-valued to Two-valued Transformation of $\mathcal{ALC}$ with closed predicates

Before we elaborate on the transformation from four-valued $\mathcal{ALC}$ with closed predicates to two-valued $\mathcal{ALC}$ with closed predicates, it is important to note that in Proposition 3.1, we state the unsatisfiability for any closed concept. The unsatisfiability applies to any open concept as well. For an open concept A, if A is subsumed by $\perp$, it is converted into a satisfiable form in which $\perp$ is transformed to $A_{new} \sqcap \neg A_{new}$ (A_{new} is a new open concept). We apply the same technique here to address the unsatisfiability associated with closed concepts. Let C be a closed concept ($C \in \Sigma$). If $C \sqsubseteq \perp$, then we convert into satisfiable form by rewriting the axiom in following

Table (6.2) Transformation

Concept/Axioms/Assertions	Transformation Result
$\pi(A)$	A where $A \notin \Sigma$ and $A \in N_C$.
$\pi(C)$	C where $C \in \Sigma \cap N_C$.
$\pi(\neg A)$	C' where C' is a new concept and $C' \notin \Sigma$ and $C' \in N_C$
$\pi(\top)$	$\top$
$\pi(\bot)$	$\bot$
$\pi(\neg\top)$	$\bot$
$\pi(\neg\bot)$	$\top$
$\pi(A \sqcap C)$	$\pi(A) \sqcap \pi(C)$
$\pi\neg(A \sqcap B)$	$\pi(\neg A) \sqcap \pi(\neg B)$
$\pi(A \sqcup C)$	$\pi(A) \sqcup \pi(C)$
$\pi\neg(A \sqcup B)$	$\pi(\neg A) \sqcup \pi(\neg B)$
$\pi(\exists r.C)$	$\exists r.\pi(C)$
$\pi(\forall r.C)$	$\forall r.\pi(C)$
$\pi\neg(\exists r.A)$	$\forall r.\pi(\neg A)$
$\pi\neg(\forall r.A)$	$\exists r.\pi(\neg A)$
$\pi(\neg\neg A)$	$\pi(A)$
$\pi(A \sqsubset C)$	$\pi(A) \sqsubseteq \pi(C)$
$\pi(A(a))$	$\pi(A)(a)$
$\pi(C(a))$	$\pi(C)(a)$
$\pi(r(a,b))$	$r(a,b)$ where r can be either closed or open

way: $C \sqsubseteq O_{new} \sqcap \neg O_{new}$ where $O_{new} \notin \Sigma$ but $O_{new} \in N_C$. Now, C is two-valued satisfiable and O_{new} is four-valued satisfiable in four-valued logic. In classical logic, C is satisfiable but O_{new} causes inconsistencies. **In this way, the inconsistencies associated with closed concepts are pushed to open concepts.** To handle the inconsistencies associated with open concepts, we use the transformation from [40,43] and [41], which is represented in Table 6.2, but Table 6.2 is added with transformation for closed concepts. In Table 6.2, B is an open concept. The transformed KB is then reasoned with description logic reasoners.

Let L_4 be the language of four-valued $\mathcal{ALC}$ with closed predicates and L_2 be the language transformed (π) from four-valued $\mathcal{ALC}$ with closed predicates. L_2 is two valued logic. Let $\mathcal{I}_4$ be the interpretation of L_4 and $\mathcal{I}_2$ be the interpretation of L_2 and it is also a two-valued correspondence of $\mathcal{I}_4$.

$\Delta^{\mathcal{I}_2}$=$\Delta^{\mathcal{I}_4}$

For each individual $a \in N_{\mathcal{I}}$, $a^{\mathcal{I}_2}$=$a^{\mathcal{I}_4}$

For each role $r \in N_R$, $r^{\mathcal{I}_4}$= $r^{\mathcal{I}_4}$

For each concept $A \in N_C$ and $A \notin \Sigma$, $A^{\mathcal{I}_2}$= proj$^+(A^{\mathcal{I}_4})$ and $A'^{\mathcal{I}_2}$= proj$^-(A^{\mathcal{I}_2})$

For each concept $C \in N_C \cap \Sigma$, $C^{\mathcal{I}_4}$= $C^{\mathcal{I}_2}$

Proposition 12. *Let $\mathcal{I}_4$ be a four-valued interpretation, $\mathcal{I}_2$ be the two valued correspondence of $\mathcal{I}_4$, and C be a atomic closed concept or a concept descriptions containing all closed predicates in four-valued $\mathcal{ALC}$. Then $C^{\mathcal{I}_4}$= $\pi(C)^{\mathcal{I}_2}$ is true.*

Proof. *If C is a atomic closed predicate, the application of transformation results $\pi(C)$= C. Therefore, $C^{\mathcal{I}_4}$ = $C^{\mathcal{I}_2}$=$\pi(C)^{\mathcal{I}_2}$. If C is $D \sqcap E$ where $\{D, E\} \subseteq N_C \cap \Sigma$, then $C^{\mathcal{I}_4}$ = $D^{\mathcal{I}_4} \cap E^{\mathcal{I}_4}$ = $\pi(D)^{\mathcal{I}_2} \cap \pi(E)^{\mathcal{I}_2}$ =$(\pi(D) \sqcap \pi(E))^{\mathcal{I}_2}$ = $\pi(D \sqcap E)^{\mathcal{I}_2}$. If C is $\forall r.D$ where $D \in \Sigma \cap N_C$, then $C^{\mathcal{I}_4}$ = $\{x \mid \forall y \mid (x, y) \in r^{\mathcal{I}_4} \rightarrow y \in D^{\mathcal{I}_4}\}$ = $\{x \mid \forall y \mid (x, y) \in r^{\mathcal{I}_2} \rightarrow y \in \pi(D)^{\mathcal{I}_2}\}$ =$(\forall r.\pi(D))^{\mathcal{I}_2}$ = $\pi(\forall r.D)^{\mathcal{I}_2}$.*

Proposition 13. *Let $\mathcal{I}_4$ be a four-valued interpretation, $\mathcal{I}_2$ be the two valued correspondence of $\mathcal{I}_4$, and C be non-atomic closed concept descriptions containing some (not all) closed predicates in four-valued $\mathcal{ALC}$. Then $proj^+(C^{\mathcal{I}_4})$= $\pi(C)^{\mathcal{I}_2}$ and $proj^-(C^{\mathcal{I}_4})$= $\pi(\neg C)^{\mathcal{I}_2}$ is true.*

Proof. *If C is $B \sqcap D$ where $D \in N_C \cap \Sigma$ and $B \in N_C$, then $proj^+(C^{\mathcal{I}_4})$ = $proj^+(B^{\mathcal{I}_4}) \cap D^{\mathcal{I}_4}$ = $\pi(B)^{\mathcal{I}_2} \cap D^{\mathcal{I}_2}$ =$(\pi(B) \sqcap \pi(D))^{\mathcal{I}_2}$ = $\pi(B \sqcap D)^{\mathcal{I}_2}$ and $proj^-(C^{\mathcal{I}_4})$ = $proj^-(B^{\mathcal{I}_4})$ =$\pi(\neg B)^{\mathcal{I}_2}$. If C is $\forall r.A$ where $A \in \Sigma \cap N_C$, then $proj^+(C^{\mathcal{I}_4})$ = $\{x \mid \forall y \mid (x, y) \in r^{\mathcal{I}_4} \rightarrow y \in A^{\mathcal{I}_4}\}$ = $\{x \mid \forall y \mid (x, y) \in r^{\mathcal{I}_2} \rightarrow y \in \pi(A)^{\mathcal{I}_2}\}$ =$(\forall r.\pi(A))^{\mathcal{I}_2}$ = $\pi(\forall r.A)^{\mathcal{I}_2}$ and $proj^-(A^{\mathcal{I}_4})$ is empty because there is no negative part for closed concepts and open (closed) roles. The other constructors with closed predicates can be proved very similarly.*

To reiterate, there is no negation before closed concepts in any *KB*s. For any open predicates O, $proj^+(O^{\mathcal{I}_4})$ is $\pi(O)^{\mathcal{I}_2}$ and $proj^-(O^{\mathcal{I}_4})$ is $\pi(\neg O)^{\mathcal{I}_2}$ [43].

Proposition 14. *Let $\mathcal{I}_4$ be a four-valued interpretation, $\mathcal{I}_2$ be the two valued correspondence of $\mathcal{I}_4$, and Ax be an axiom or assertion in four-valued $\mathcal{ALC}$ that has closed predicates. $\mathcal{I}_4$ is a model for Ax iff $\mathcal{I}_2$ is a model for $\pi(Ax)$.*

Proof. *If $Ax= C(a)$ where $C \in N_C \cap \Sigma$, then $\pi(Ax) = \pi(C)(a)$. Therefore, $a^{\mathcal{I}_4} \in C^{\mathcal{I}_4}$ iff $a^{\mathcal{I}_2} \in C^{\mathcal{I}_2}$ (By Proposition 12 and Proposition 13). If $Ax= C \sqsubset D$ where $C \in N_C \cap \Sigma$ and $D \notin \Sigma$ but $D \in N_C$, then $C^{\mathcal{I}_4} \subseteq proj^{+}(D^{\mathcal{I}_4})$ iff $\pi(C)^{\mathcal{I}_2} \subseteq \pi(D)^{\mathcal{I}_2}$ iff $\mathcal{I}_2$ is a two valued model of $\pi(C) \sqsubset \pi(D)$ iff $\mathcal{I}_2$ is a two valued model of $\pi(C \sqsubset D)$.*

Proposition 15. *Let KB be a four-valued $\mathcal{ALC}$ with closed predicates and Q be a query from four-valued $\mathcal{ALC}$. $KB \vDash_4 Q$ iff $\pi(KB) \vDash_2 \pi(Q)$.*

Proposition 12, Proposition 13 and Proposition 14 yield the proof for Proposition 15.

6.3 Example

The following example consolidates the works presented in the chapter. The example is taken from [23], and it is extended to $\mathcal{ALC}$. Moreover, we introduced an inconsistency in it.

Example 7. *Let KB be a $\mathcal{ALC}$ that has TBox, closed predicates, and ABox.*

TBox:

$ScandComp \sqsubseteq \exists based\_in.ScandCountry$

$ScandComp \sqsubseteq \bot$

The closed predicate (Σ) is $ScandComp$.

ABox:

$ScandComp(cp), ScandCountry(denmark),$

$ScandCountry(normway), ScandCountry(sweden),$

$TimberExporter(denmark), TimberExporter(norway),$

$TimberExporter(sweden)$

Query (Q):

$(\exists based\_in.TimberExporter \sqcap ScandComp)(cp)$

Solution. By Proposition 5, $ScandComp$ is unsatisfiable. So we start handling it by representing the KB and Q as four-valued $\mathcal{ALC}$.

$ScandComp \sqsubset \exists based\_in.ScandCountry$

$ScandComp \sqsubset \bot$

Now, we convert the TBox into a satisfiable form.

$ScandComp \sqsubset \exists based\_in.ScandCountry$

$ScandComp \sqsubset A_{new} \sqcap \neg A_{new}$

Then, we transform (π) KB and Q.

$ScandComp \sqsubseteq \exists based\_in.ScandCountry$

$ScandComp \sqsubseteq A_{new} \sqcap A'_{new}$

Now, Q is true for the transformed KB.

CHAPTER 7

DESCRIPTION LOGIC PROGRAMS: A PARACONSISTENT RELATIONAL MODEL APPROACH

7.1 Introduction

The web ontology language (OWL) [35, 49], which is a W3C recommendation, is primarily based on description logic formalism [24], and is a backbone for future information systems. Although description logic is used for modeling the domain of interest, the rule-based systems [50] have many commercial applications [51]. Moreover, both types of formalism (description logic and rule) are based on first-order logic (FOL). This led to the development of the W3C Recommendation rule interchange format (RIF) [52, 53].

In this chapter, we focus on *dl-programs* [31], which is a loose coupling method (rules may contain queries to description logic) that provides the semantics for the integration of description logic and rules. The integration is achieved through the use of *dl-atoms*, which is a special type of atom that occurs only in the body of the rules. Concretely, the *dl-atom* enables a bi-directional flow of information between description logic and the logic program. The main reason for choosing *dl-programs* in this thesis is that the satisfaction of *dl-programs* is an extension of the usual notion of satisfaction of logic programs by Herbrand Interpretation.

Bagai and Sunderraman [19] proposed a data model to represent incomplete and inconsistent information in databases. The paraconsistent logic studied by da Costa [82] and Belnap [4] forms the basis for this data model. Instead of eliminating incomplete and inconsistent information, this model attempts to operate in its presence. The mathematical structures underlying the model, called paraconsistent relations, are a generalization of ordinary relations. Paraconsistent relations represent both positive and negative tuples.

Moreover, using the paraconsistent relation model, Bajai and Sunderraman [19] proposed some elegant methods for determining weak well-founded model [83] and well-founded model

[20] for general deductive databases. But, there are not any methods based on the paraconsistent relational model proposed to determine models for *dl-programs*. In this paper, we propose an algorithm to determine the fixed-point semantics of the function free positive *dl-program*. Our idea essentially involves creating a paraconsistent relation for each predicate symbol in the rules and then forming a system of algebraic equations using paraconsistent algebraic operators for all *dl-rules* (ordinary rules containing *dl-atoms*) in *dl-programs*. Then, solve the equations to find the fixed-point semantics of the positive *dl-programs*.

This algebraic approach of finding the fixed-point semantics of a positive *dl-program* has two main advantages: it operates on a set of tuples in contrast to non-algebraic approaches, which operate on a tuple at a time basis; the algebraic expression in the equations can be optimized using various laws of equality, which is very similar to the ordinary relation case where selection and projection are pushed deeper into expressions whenever necessary.

7.2 Algorithm

Before we define the algorithm, we first define the DL-relations, which are equivalent of dl-atoms in description logic program.

7.2.1 DL-Relations

As we already stated in the Introduction section, we will construct a relation for every atom in the rules during model construction. In *dl-programs*, the body of the rules can have *dl-atoms*. It is necessary to have equivalent relations for such *dl-atoms*. It is achieved by redefining *dl-atoms* in terms of paraconsistent relations. To recall, *dl-atoms* are of the form $DL[S_1 op_1 p_1, \ldots, S_m op_m p_m; Q](\mathbf{t}), m \geqslant 1$. Specifically, $p_1, \ldots, p_m$ are input predicate symbols and $Q(\mathbf{t})$ is a *dl-query*. An important observation is that $Q(\mathbf{t})$ performs query entailment and not query answering for a given description logic knowledge base. This is because *dl-atoms* are grounded to determine models [30, 31]. As a first step towards modifying *dl-atoms* for our purpose, we will transform constants in $\mathbf{t}$ of dl-queries to variables. By doing so, the *dl-query* performs query answering in the given description logic knowledge base. Next, we create paraconsistent relations for every input predicate symbols. We know that $op_i = \{\uplus, \uminus\}$. In order to denote that $\uplus$ ($\uminus$) adds tuples from relations to concepts or roles, we write $\uplus$ ($\uminus$) as $\dot{\uplus}$ ($\dot{\uminus}$). The update operator $\dot{\uplus}$ takes every tuple from $\dot{\pi}^{+}_{\{\Sigma\}}(p_i)$, where p_i is a relation, and inserts it in S_i. Similarly, $\dot{\uminus}$ takes every tuple from $\dot{\pi}^{+}_{\{\Sigma\}}(p_i)$, where p_i is a relation, and inserts it in $\neg S_i$. Hence the *dl-relation* is,

$$DL[S_1 o\dot{p}_1 \dot{\pi}^{+}_{\{\Sigma\}}(p_1), \ldots, S_m o\dot{p}_m \dot{\pi}^{+}_{\{\Sigma\}}(p_m); Q](\mathcal{V}), m \geqslant 1 \tag{7.1}$$

$p_1, \ldots, p_m$ are relations and $o\dot{p}_i = \{\dot{\uplus}, \dot{\uminus}\}$. $\mathcal{V}$ is the scheme of the *dl-relation* shown in (7.1). $Q(\mathcal{V})$ is called *dl-query* answering, and $Q(\mathcal{V})$ is a concept assertion or negated concept assertion, a role assertion or negated role assertion, or an equality or inequality axiom. Since $Q(\mathcal{V})$ is a query answering, it returns a set of individuals which is then added as tuples in the positive part of a *dl-relation*.

During model computation, instead of representing the *dl-relation* as shown in (7.1), we created a new relation for it. For the *dl-relation* shown in (7.1), the new relation is $R_{DL[S_1 o\dot{p}_1 \dot{\pi}^{+}_{\{\Sigma\}}(p_1), \ldots, S_m o\dot{p}_m \dot{\pi}^{+}_{\{\Sigma\}}(p_m); Q]}(\mathcal{V})$. In a *dl-relation*, we never insert any result of query answering into a *dl-relation* as tuples unless the result is in accordance to the domain values of the *dl-relation*'s scheme. In other words, $I_P \subseteq C \equiv dom(a) \subseteq \mathbf{I} \cup \mathbf{V}$, where I_P is the set of all constant symbols appearing in P and all $a \in \mathcal{V}$.

In the following section, we will explain two steps for the algorithm to determine the fixed-point semantics of *dl-programs*. In addition to that, we prove that the algorithm is correct and provide an example for it.

7.2.2 Fixed-Point Semantics for *Dl-programs*

By using the algebra of the relational model, we present a bottom-up method for constructing models of *dl-programs* that mimics the immediate consequence operator (T_{KB}). The algorithm presented in this thesis is based on the construction of well-founded semantics [20] and weak well-founded semantics [83] using the relational model for general deductive database. The model construction involves two steps. The first step is to convert P into a set of relation definitions for the predicate symbols occurring in P. These definitions are of the form

$$p = D_p$$

where p is a predicate symbol of P, and D_p is an algebraic expression involving predicate symbols of P and relation operators. The second step is to evaluate iteratively the expressions in these definitions to construct incrementally the relations associated with the predicate symbols. The first step is called SERIALIZE and the second step is called MODEL CONSTRUCTION.

The schemes of the relations are set internally. Hence, the following definition. Let $\Gamma_n = \langle v_1, v_2 \ldots \rangle$ be an infinite sequence of some distinct attribute names. For any $n \geq 1$, let Γ_n be the scheme $\{v_1 \ldots v_n\}$. The following operator renames the scheme of the relation from one to another.

Definition 4. *Let $\Sigma = \{A_1 \ldots A_n\}$ be any scheme. Then,*

1. *for any relation R on scheme Γ_n, $R(A_1 \ldots A_n)$ is the relation*

$$\delta_{v_1 \ldots v_n \rightarrow A_1 \ldots A_n}(R)$$

   on scheme Σ, and

2. *for any relation R on scheme Σ, $R(v_1 \ldots v_n)$ is the relation*

$$\delta_{A_1 \ldots A_n \rightarrow v_1 \ldots v_n}(R)$$

   on scheme Γ_n.

Before we get into details of the algorithm, we should replace every *dl-atom* $DL[\lambda; Q](\mathbf{t})$ by a fresh predicate $p_{DL[\lambda;Q]}(\mathbf{t})$ so that it would be easy to create the corresponding *dl-relation*.

Example 8. *Using the same KB from Example 3.*

Solution. $r(a) \leftarrow g_{DL[S \uplus q;C]}(a)$; $q(a) \leftarrow p(a)$; $p(a) \leftarrow$

Here, $g_{DL[S \uplus q;C]}$ is a new predicate symbol.

In the remaining part of this section we describe our method to convert the given *dl-rules* in KB ($KB = (L, P)$) into a set of definitions for the predicate symbol in P.

ALGORITHM. SERIALIZE

Input. A *dl-rule* (definite rule) $l_0 \leftarrow l_1, \ldots, l_z$. Let l_0 be an atom of the form $p_0(A_{01} \ldots A_{0k_0})$, and each l_i, $1 \leq i \leq z$, be an atom either of the form $p_i(A_{i1} \ldots A_{ik_i})$ or $p_{DL[\lambda;Q]}(\mathbf{t})$[1]. Let V_i be the set of all variables occurring in l_i

Output. An algebraic expression involving paraconsistent relations.

Method. The expression is constructed by the following steps:

1. For each argument A_{ij} of literal l_i, construct argument B_{ij} and condition C_{ij} as follows:

   (a) If A_{ij} is a constant a, then B_{ij} is any brand new variable and C_{ij} is B_{ij}=a.

   (b) If A_{ij} is a variable, such that for each k, $1 \leq k < j$, $A_{ik} \neq A_{ij}$, then B_{ij} is A_{ij} and C_{ij} is true.

   (c) If A_{ij} is a variable, such that for some k, $1 \leq k < j$, A_{ik}=A_{ij}, then B_{ij} is a brand new variable and C_{ij} is $A_{ij} = B_{ij}$.

2. Let $\hat{l}_i$ be the atom $p_i(B_{i1} \ldots B_{ik_i})$, and F_i be the conjunction $C_{i1} \wedge \cdots \wedge C_{ik_i}$. Then Q_i is the expression $\dot{\pi}_{V_i}\dot{\sigma}_{F_i}(\hat{l}_i)$.

   As a syntatic optimisation, if all conjuncts of F_i are true (i.e. all arguments of l_i are distinct variables), then both $\dot{\sigma}_{F_i}$ and $\dot{\pi}_{V_i}$ are reduced to identity operations, and are hence dropped from the expression.

   If $\hat{l}_i$ is the atom $p_{DL[\lambda;Q]}(\mathbf{t})$, then Q_i is the expression $\dot{\pi}_{V_i}\dot{\sigma}_{F_i}(\hat{l}_i)$ where every input predicate symbol p_i in λ is $\dot{\pi}_{V_{p_i}}(p_i)^{+}$ in which V_{p_i} refers to a set of variables in p_i.

3. Let E be the natural join ($\dot{\bowtie}$) of the Q_i's thus obtained, $1 \leq i \leq z$. The output expression is $(\dot{\sigma}_{F_0}(\dot{\pi}_{V'}(E)))\ [B_{01} \ldots B_{n_{kn}}]$. V' is a set of variables occurring in l_0.

   As in step 2, if all conjuncts in F_0 are true, then $\dot{\sigma}_{F_0}$ is dropped from the output expression. However, $\dot{\pi}_{V'}$ is never dropped, as the rule may contain variables not in V'.

From the algebraic expression of the algorithm, we construct a system of equations.

[1] $\lambda = S_1 op_1 p_1, \ldots, S_m op_m p_m$

For any *dl-program* $KB = (L, P)$, $EQN(P)$ is a set of all equations of the form $p = D_p$, where p is a predicate symbol, and D_p is the union ($\dot{\cup}$) of all expressions obtained by the algorithm *SERIALIZE* for the rules (*dl-rules*) in P with symbol p in their head. The algebraic expression D_p is also called a *definition* of p.

It is easy to observe that a predicate symbol may have many definitions. We now show that the above method for converting a *dl-program KB* into definitions for its predicate symbols terminates, and that the definitions produced mimics the immediate consequence operator (T_{KB}).

Proposition 16 (Termination)**.** *The procedure of constructing EQN(P) terminates for any dl-program* $KB(KB = (L, P))$*.*

Proof. *The proof is immediate from the fact that P has only a finite number of rules (dl-rules) that each rule contains a finite number of atoms (dl-atoms) and each atom (dl-atom)* [2] *has a finite number of arguments.*

The transformation between a relation and an interpretation is necessary for the correctness of the proof.

Definition 5. *Let I be any interpretation and* $r(X_1 \ldots X_n)$ *be any atom (dl-atom), where the* X_i*'s are distinct variables. Then* $I \triangleright r$ *is the following relation*

$r^+ = \langle t \in \tau(\Sigma) \mid r(t(X_1) \ldots t(X_n) \in I \rangle$

$r^- = \emptyset$

on scheme $\Sigma = \{X_1, \ldots, X_n\}$*. Moreover, for any relation R on scheme* Σ*,* $r\lfloor R \rfloor$ *is the following interpretation*

$\langle r(t(X_1) \ldots t(X_n)) \mid t \in R^+ \rangle$

In the following, we show the correctness of SERIALIZE.

Proposition 17 (Correctness)**.** *Let* $a_1, \ldots, a_n$ *be atoms (dl-atoms) occurring in the definition of some equation p=*D_p *in* $EQN(P)$*, for any dl program* $KB = (L, P)$*. Let* k_0 *be the arity of p and let*

[2] *dl-atoms cannot have more than two arguments*

each a_i be of the form $p_i(B_{i1},\ldots,B_{ik_i})$ or $p_{DL[\lambda;Q]}(\boldsymbol{t})$[3]*. For all i, $1 \le i \le n$, let R_i be any relation on scheme Γ_{k_i}, such that if for any i, j, $p_i = p_j$, then $R_i = R_j$. Then, the relation R on scheme Γ_{k_0} obtained by evaluating D_p by interpreting each p_i as the relation R_i is*

$$T_{KB}(\langle\bigcup_{i=1}^{n} p_i\lfloor R_i\rfloor\rangle) \rhd p$$

,

Proof. *The proof essentially involves the definitions of the relation operators defined earlier. Here we give an easy to understand sketch. Let*

$$I = (\langle\bigcup_{i=1}^{n} p_i\lfloor R_i\rfloor\rangle)$$

. We divide the proof in the following two parts:

1. *($\rightarrow$) Suppose for any $t \in R^+$. Then, by the definition of $\dot\cup$, t is in the positive part of the expression*

   $$(\dot{\sigma}_{F_0}\dot{\pi}_{V_1}(E))[B_{01}\ldots B_{0k_0}])$$

   output by step 3 of the algorithm SERIALIZE, for some rule (dl-rule) in P with symbol p in its head. Let Σ be the scheme of the relation E. Then, for some tuple $t' \in (\delta_{v_1\ldots v_{k_0}\rightarrow B_{01}\ldots B_{0k_0}}$ $(t))^{V'\cup\Sigma}$ is in E^+. Thus, for each Q_i in E, there is a tuple $t_i \in Q_i^+$ such that for each variable $X \in V_i$, $t'(X) = t_i(X)$. If Q_i is a dl-relation, then the tuple (t_i) is in the positive part of Q. By step 2 of the algorithm,

   (a) *if the corresponding atom l_i in the rule is positive, then $t_i \in Q_i^+$*

   (b) *if the corresponding atom l_i is a dl-atom in the rule, then $t_i \in Q_i^+$. Here Q_i is a dl-relation.*

   Therefore, due to the ground instance of this rule (dl-rule) for the "substitution" t′, we have that $t \in T_{KB}(I)$.

---

[3] $\lambda = S_1 op_1 p_1, \ldots, S_m op_m p_m$

2. *($\leftarrow$) Suppose $t \in (T_{KB}(I) \triangleright p)^+$. Then, for some ground instance,*

$$p(t(B_{01})\ldots t(B_{0k_0})) \leftarrow l_1 \ldots l_z$$

of a clause in P, we have that the atom of each l_i is in the correct part of I. For that clause of P, let

$$(\dot{\sigma}_{F_0}\dot{\pi}_{V'}(E))[B_{01}\ldots B_{0k_0}])$$

be the expression output by step 3 of the algorithm SERIALIZE, and let Σ be the scheme of E. So, for each Q_i in E, there is a tuple $t_i \in Q_i^+$ such that for all $X \in V_i$, $t_i(X) = t'(X)$ for some $t' \in (\delta_{v_1\ldots v_{k_0}\to B_{01}\ldots B_{0k_0}}(t))^{V\cup\Sigma}$. Hence, $t \in R^+$.

Example 9. *Consider $KB = (L, P)$, where $L = \{S \sqsubseteq C \sqcap D, D(b)\}$ and P is as follows:*

$r(X) \leftarrow DL[S \uplus q; D](X);$

$r(X) \leftarrow DL[S \uplus q; C](X), w(X);$

$q(X) \leftarrow p(X);$

$p(a) \leftarrow$

$w(a) \leftarrow$

Solution. We construct an equation for every rule (*dl-rule*) in the *KB*. Since we have *dl-rules* in the *KB*, we need to replace *dl-atoms* with a new predicate.

$r(X) \leftarrow f_{DL[S\uplus q;D]}(X);$

$r(X) \leftarrow g_{DL[S\uplus q;C]}(X), w(X);$

$q(X) \leftarrow p(X);$

Now, $f_{DL[S\uplus q;D]}$ and $g_{DL[S\uplus q;C]}$ are two new predicate symbols. Then, we convert *KB* into a system of equations.

1. $r = \dot{\pi}_{\{X\}}(f_{DL[S\dot{\uplus}\dot{\pi}_{\{X\}}(q)^+;D]}(X))[X]$

2. $r = (\dot{\pi}_{\{X\}}(g_{DL[S\dot{\uplus}\dot{\pi}_{\{X\}}(q)^+;C]}(X)\dot{\bowtie} w(X)))[X]$

3. $q = \dot{\pi}_{\{X\}}(p(X))[X]$

The LHS expression of the first and second equation are the same. Therefore,

1. $r = \dot{\pi}_{\{X\}}(f_{DL[S \dot{\uplus} \dot{\pi}_{\{X\}}(q)^{+};D]}(X))[X] \dot{\cup}$

   $(\dot{\pi}_{\{X\}}(g_{DL[S \dot{\uplus} \dot{\pi}_{\{X\}}(q)^{+};C]}(X) \dot{\bowtie} w(X)))[X]$

2. $q = \dot{\pi}_{\{X\}}(p(X))[X]$

The second step is to construct the model by incrementally constructing the relation values in P. For any P in *dl-program* $KB = (L, P)$, P_E are the facts (rules in P without bodies), and P_I are the rules (rules in P with bodies). P_E^* refers to a set of all ground instances of rules in P_E. Without the loss of generality, we assume that no predicate symbol occurs both in P_E and in P_I.

ALGORITHM. MODEL CONSTRUCTION

Input. A *dl-program* $(KB = (L, P))$

Output. Relation values for the predicate symbols in P.

Method: The following steps compute the values:

1. (Initialization)

   (a) Compute EQN(P_I) using the algorithm *SERIALIZE* for each clause in P_I.

   (b) For each predicate symbol p in P_E, set

   $p^{+} = \{\langle a1 \ldots ak \rangle \mid p(a1 \ldots, ak) \in P_E^*\}$, and

   $p^{-} = \emptyset$

   (c) For each predicate symbol p in P_I, set $p^{+} = \emptyset$ and $p^{-} = \emptyset$.

2. For each equation of the form $p = D_p$ in $EQN(P_I)$, compute the expression D_p and set p to the following relation. If the expression contains a *dl-relation*, then perform query answering in the given description logic knowledge base (L).

3. If step 2 involved a change in the value of some p, goto 2.

4. Output the final values of all predicate symbol in P_E and P_I.

Now, we prove the termination of the second step of the algorithm.

Proposition 18 (Termination)**.** *Algorithm* ***MODEL CONSTRUCTION*** *terminates for all dl-programs.*

Proof. *By Proposition 16, step 1 always terminates. By Proposition 4 and 17, the loop in step 2-3 always terminates.*

Next, we prove that the algorithm MODEL CONSTRUCTION is correct.

Theorem 1 (Correctness)**.** *A tuple $\langle a_1, \dots a_k \rangle$ is in p^+ computed by the algorithm MODEL CONSTRUCTION iff $p(a_1, \dots, a_k) \in T_{KB} \uparrow \omega$.*

Proof. *Following from the fact that $T_{KB} \uparrow 1$ is set up by the initialization step, and by Proposition 17, step 2 mimics the T_{KB} operator, whose power always converges by Proposition 4.*

7.3 Example

The following example consolidates our work presented in this thesis. Here we represent relations in the form of tables in which the positive and negative parts are separated by a double line.

Example 10. *Using the same KB from Example 9.*

Solution. By step 1 (a), $EQN(P_I)$ returns two equations:

1. $r = \dot{\pi}_{\{X\}}(f_{DL[S \dot{\uplus} \dot{\pi}_{\{X\}}(q)^+;D]}(X))[X] \dot{\cup}$

   $(\dot{\pi}_{\{X\}}(g_{DL[S \dot{\uplus} \dot{\pi}_{\{X\}}(q)^+;C]}(X) \dot{\bowtie} w(X)))[X]$

2. $q = \dot{\pi}_{\{X\}}(p(X))[X]$

The domain value of every relation's attribute is $\{a\}$. By step 1 (b),

$w = \begin{array}{|c|}\hline \{X\} \\ \hline \langle a \rangle \\ \hline\hline \end{array}$ and $p = \begin{array}{|c|}\hline \{X\} \\ \hline \langle a \rangle \\ \hline\hline \end{array}$

Step 1 mimics the $T_{KB} \uparrow 1$.

In step 2, we have two equations. After applying the second equation,

$$w = \begin{array}{|c|}\hline \{X\} \\ \hline \langle a \rangle \\ \hline \end{array},\ p = \begin{array}{|c|}\hline \{X\} \\ \hline \langle a \rangle \\ \hline \end{array} \text{ and } q = \begin{array}{|c|}\hline \{X\} \\ \hline \langle a \rangle \\ \hline \end{array}$$

Now, it is important to observe that the first equation has two *dl-relations*. So, it is necessary to perform query answering on the description logic knowledge base (L).

$$f_{DL[S \dot{\uplus} \dot{\pi}_{\{X\}}(q)^+;D]} = \begin{array}{|c|}\hline \{X\} \\ \hline \langle a \rangle \\ \hline \end{array}$$

In the above relation, the tuples in q are inserted into concept S, and query answering ($D(X)$) is performed. The description logic knowledge base (L) already contains an assertion $D(b)$ but the domain values of *dl-relation* scheme does not contain b. Hence, the above relation has only one tuple. Next,

$$g_{DL[S \dot{\uplus} \dot{\pi}_{\{X\}}(q)^+;C]} = \begin{array}{|c|}\hline \{X\} \\ \hline \langle a \rangle \\ \hline \end{array}$$

After computing the second equation, we have the following:

$$r = \begin{array}{|c|}\hline \{X\} \\ \hline \langle a \rangle \\ \hline \end{array}$$

Finally, we have the following:

$$r = \begin{array}{|c|}\hline \{X\} \\ \hline \langle a \rangle \\ \hline \end{array},\ w = \begin{array}{|c|}\hline \{X\} \\ \hline \langle a \rangle \\ \hline \end{array},\ p = \begin{array}{|c|}\hline \{X\} \\ \hline \langle a \rangle \\ \hline \end{array} \text{ and } q = \begin{array}{|c|}\hline \{X\} \\ \hline \langle a \rangle \\ \hline \end{array}$$

Further iterations of step 2 do not change the values of relations. Step 2 mimics the T_{KB} operator.

In other words, $T_{KB} \uparrow \omega = \{r(a), w(a), p(a), q(a)\}$

CHAPTER 8

CONCLUSIONS

This thesis presents a variety of algorithms suitable for inconsistent handling in different representations. As an illustration, we presented a working example for each and every type of algorithm. We already know that traditional reasoners fail to reason in the presence of consistencies; the approaches presented in this thesis help to build an inconsistent tolerant reasoner. These are very sophisticated approaches that handle not only inconsistencies but also incompleteness.

In Chapter 4 (and 5), we proposed an algorithm to find QC models (and p-minimal models) for any positive extended disjunctive deductive databases. Specifically, in Chapter 4, we introduced a new disjunctive relational model to represent the relations containing paraconsistent unions. Also, in Chapter 5, we reused the disjunctive relational model to construct p-minimal models.

The algorithms in Chapter 4 (and 5) that we presented here is based on the algorithm that is used to compute the well-founded model using a relational model. In query-intensive applications, this precomputation of the model enables efficient processing of subsequent queries. In [6], Gelfond and Lifschitz adopt the way of trivializing results while the algorithm tolerates inconsistencies. Though we find the model to be correct for any given positive extended disjunctive deductive database, the algorithm doesn't find models for the databases with recursions and constraints. One direction of future work can be expanding the algorithm to allow recursions and constraints.

Moreover, the models that we construct is too strong in Chapter 4; it causes disjunction introduction and modus tollens to fail, but they are supported by QC logic. To compute the QC entailment, it is necessary to have both weak and strong models. So another direction of future work will be finding the weak models for the same program. Hence, the QC entailment can be done.

For models presented in Chapter 4 and 5, the creation of many proper disjunctive databases are expensive, given the QC logic (or p-minimal) model computation and are probably not worth the extra computation. It would be very interesting to analyze the algorithm by allowing default negation in program P. Finally, we notice that we have not stated the complexities, and we have left it for future work.

In Chapter 6, the previous research on the mix of open and closed predicates [23] and [26] is extended to $\mathcal{ALC}$. We then handled inconsistencies associated with closed predicates in $\mathcal{ALC}$. The novelty of the approach is the representation of closed predicates in four-valued $\mathcal{ALC}$ and handling its inconsistencies. We have used the ideas of Maier et al. [43] and proved that four-valued $\mathcal{ALC}$ with closed predicates is sound on two-valued $\mathcal{ALC}$ with closed predicates. In addition to that, we transformed four-valued $\mathcal{ALC}$ with closed predicates to two-valued $\mathcal{ALC}$ with closed predicates.

Even though the works presented in the thesis can handle inconsistencies associated with closed predicates, it is not complete. It would be very interesting to analyze this work by allowing strong inclusion, which supports contrapositive reasoning, in four-valued $\mathcal{ALC}$ with closed predicates. As material inclusion is not a very strong form of reasoning, we chose not to analyze it. Another interesting direction of future work could be extending the queries to conjunctive queries. One more direction of future would be using more expressive description logic like $\mathcal{SROIQ}$ [84] for LCWR and handling the inconsistencies associated with it.

In Chapter 7, we took Eiter et al.'s *dl-program* [29–33] and represented it in terms of the paraconsistent relational model. we also introduced the *dl-relation* to represent the *dl-atom*, which gets its tuples from description logic knowledge base. We then determined the fixed-point semantics of positive *dl-programs* using paraconsistent algebraic operators and proved the correctness of it.

It is important to note that we can use the paraconsistent relations that were obtained at the end of the algorithm for querying using paraconsistent tuple relational calculus [85]. Thus, expressive queries can be given to paraconsistent relations. Even though we correctly find the fixed-point semantics of *dl-programs*, the given algorithm in this thesis is not complete. There are two more possible directions of future works for Chapter 7. The first work would be extending the algorithm to accommodate default negation (*not*) and to determine the well-found semantics of *dl-programs* in the paraconsistent relation model. The second work would be representing different formalisms (as mentioned in the Introduction section of Chapter 7) for integration of description logic and rules in the paraconsistent relation model to find its model.

REFERENCES

[1] A. Hunter, "Paraconsistent logics, handbook of defeasible reasoning and uncertainty management systems: volume 2: reasoning with actual and potential contradictions," 1998.

[2] W. Carnielli, M. E. Coniglio, and J. Marcos, "Logics of formal inconsistency," in *Handbook of philosophical logic*. Springer, 2007, pp. 1–93.

[3] O. Arieli, "Distance-based paraconsistent logics," *International Journal of Approximate Reasoning*, vol. 48, no. 3, pp. 766–783, 2008.

[4] N. D. Belnap Jr, "A useful four-valued logic," in *Modern uses of multiple-valued logic*. Springer, 1977, pp. 5–37.

[5] H. A. Blair and V. Subrahmanian, "Paraconsistent logic programming," *Theoretical computer science*, vol. 68, no. 2, pp. 135–154, 1989.

[6] M. Gelfond and V. Lifschitz, "Classical negation in logic programs and disjunctive databases," *New generation computing*, vol. 9, no. 3-4, pp. 365–385, 1991.

[7] C. Sakama and K. Inoue, "Paraconsistent stable semantics for extended disjunctive programs," *Journal of Logic and Computation*, vol. 5, no. 3, pp. 265–285, 1995.

[8] J. Alcântara, C. V. Damásio, and L. M. Pereira, "A declarative characterisation of disjunctive paraconsistent answer sets," in *ECAI*, vol. 16. Citeseer, 2004, p. 951.

[9] ——, "Paraconsistent logic programs," in *Logics in Artificial Intelligence*. Springer, 2002, pp. 345–356.

[10] O. Arieli, "Paraconsistent declarative semantics for extended logic programs," *Annals of Mathematics and Artificial Intelligence*, vol. 36, no. 4, pp. 381–417, 2002.

[11] C. V. Damásio and L. M. Pereira, "A survey of paraconsistent semantics for logic programs," in *Reasoning with Actual and Potential Contradictions*. Springer, 1998, pp. 241–320.

[12] Z. Zhang, Z. Lin, and S. Ren, “Quasi-classical model semantics for logic programs–a paraconsistent approach,” in *Foundations of Intelligent Systems*. Springer, 2009, pp. 181–190.

[13] A. Hunter, “Reasoning with contradictory information using quasi-classical logic,” *Journal of Logic and Computation*, vol. 10, no. 5, pp. 677–703, 2000.

[14] M. Osorio, C. Zepeda, J. C. Nieves, and J. L. Carballido, “G3-stable semantics and inconsistency,” *Computación y Sistemas*, vol. 13, no. 1, pp. 75–86, 2009.

[15] N. Mayatskiy and S. P. Odintsov, “On deductive bases for paraconsistent answer set semantics,” *Journal of Applied Non-Classical Logics*, vol. 23, no. 1-2, pp. 131–146, 2013.

[16] J. Alcântara, C. V. Damásio, and L. M. Pereira, “An encompassing framework for paraconsistent logic programs,” *Journal of Applied Logic*, vol. 3, no. 1, pp. 67–95, 2005.

[17] M. J. O. Galindo, J. R. A. Ramírez, and J. L. Carballido, “Logical weak completions of paraconsistent logics,” *J. Log. Comput.*, vol. 18, no. 6, pp. 913–940, 2008. [Online]. Available: http://dx.doi.org/10.1093/logcom/exn015

[18] M. Osorio, J. A. N. Pérez, J. R. A. Ramírez, and V. B. Macías, “Logics with common weak completions,” *J. Log. Comput.*, vol. 16, no. 6, pp. 867–890, 2006. [Online]. Available: http://dx.doi.org/10.1093/logcom/exl013

[19] R. Bagai and R. Sunderraman, “A paraconsistent relational data model,” *International Journal of Computer Mathematics*, vol. 55, no. 1-2, pp. 39–55, 1995.

[20] ——, “Computing the well-founded model of deductive databases,” *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, vol. 4, no. 02, pp. 157–175, 1996.

[21] V. Subrahmanian, “Paraconsistent disjunctive deductive databases,” in *Multiple-Valued Logic, 1990., Proceedings of the Twentieth International Symposium on*. IEEE, 1990, pp. 339–346.

[22] J. Minker and D. Seipel, "Disjunctive logic programming: A survey and assessment," in *Computational Logic: Logic Programming and Beyond, Essays in Honour of Robert A. Kowalski, Part I*. Springer-Verlag, 2002, pp. 472–511.

[23] C. Lutz, I. Seylan, and F. Wolter, "Ontology-based data access with closed predicates is inherently intractable(sometimes)," in *IJCAI 2013, Proceedings of the 23rd International Joint Conference on Artificial Intelligence, Beijing, China, August 3-9, 2013*, F. Rossi, Ed. IJCAI/AAAI, 2013. [Online]. Available: http://www.aaai.org/ocs/index.php/IJCAI/IJCAI13/paper/view/6870

[24] F. Baader, D. Calvanese, D. L. McGuinness, D. Nardi, and P. F. Patel-Schneider, Eds., *The Description Logic Handbook: Theory, Implementation, and Applications*. New York, NY, USA: Cambridge University Press, 2010.

[25] B. Glimm, "Querying description logic knowledge bases," Ph.D. dissertation, University of Manchester, 2007.

[26] C. Lutz, I. Seylan, and F. Wolter, "Mixing open and closed world assumptionin ontology-based data access: Non-uniform data complexity," in *Proceedings of the 2012 International Workshop on Description Logics, DL-2012, Rome, Italy, June 7-10, 2012*, ser. CEUR Workshop Proceedings, Y. Kazakov, D. Lembo, and F. Wolter, Eds., vol. 846. CEUR-WS.org, 2012. [Online]. Available: http://ceur-ws.org/Vol-846

[27] M. Fitting, "Fixpoint semantics for logic programming a survey," *Theoretical computer science*, vol. 278, no. 1, pp. 25–51, 2002.

[28] I. Horrocks and P. Patel-Schneider, "Reducing owl entailment to description logic satisfiability," *Web Semantics: Science, Services and Agents on the World Wide Web*, vol. 1, no. 4, pp. 345–357, 2004.

[29] T. Eiter, G. Ianni, T. Lukasiewicz, R. Schindlauer, and H. Tompits, "Combining answer set programming with description logics for the semantic web," *Artificial Intelligence*, vol. 172, no. 12, pp. 1495–1539, 2008.

[30] T. Eiter, T. Lukasiewicz, R. Schindlauer, and H. Tompits, "Well-founded semantics for description logic programs in the semantic web," in *Rules and Rule Markup Languages for the Semantic Web: Third International Workshop, RuleML 2004, Hiroshima, Japan, November 8, 2004. Proceedings*, 2004, pp. 81–97. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-30504-0_7

[31] T. Eiter, G. Ianni, T. Lukasiewicz, and R. Schindlauer, "Well-founded semantics for description logic programs in the semantic web," *ACM Transactions on Computational Logic (TOCL)*, vol. 12, no. 2, p. 11, 2011.

[32] T. Eiter, T. Lukasiewicz, R. Schindlauer, and H. Tompits, "Combining answer set programming with description logics for the semantic web," in *Principles of Knowledge Representation and Reasoning: Proceedings of the Ninth International Conference (KR2004), Whistler, Canada, June 2-5, 2004*, 2004, pp. 141–151. [Online]. Available: http://www.aaai.org/Library/KR/2004/kr04-017.php

[33] T. Eiter, G. Ianni, A. Polleres, R. Schindlauer, and H. Tompits, "Reasoning with rules and ontologies," in *Reasoning web*. Springer, 2006, pp. 93–127.

[34] T. Berners-Lee, J. Hendler, O. Lassila *et al.*, "The semantic web," *Scientific american*, vol. 284, no. 5, pp. 28–37, 2001.

[35] D. L. McGuinness, F. Van Harmelen *et al.*, "Owl web ontology language overview," *W3C recommendation*, vol. 10, no. 10, p. 2004, 2004.

[36] K. Wang, D. Billington, J. Blee, and G. Antoniou, "Combining description logic and defeasible logic for the semantic web," in *Rules and Rule Markup Languages for the Semantic Web*. Springer, 2004, pp. 170–181.

[37] M. Dao-Tran, T. Eiter, and T. Krennwallner, "Realizing default logic over description logic knowledge bases," in *Symbolic and Quantitative Approaches to Reasoning with Uncertainty*. Springer, 2009, pp. 602–613.

[38] K. Moodley, T. Meyer, and I. J. Varzinczak, "A defeasible reasoning approach for description logic ontologies," in *Proceedings of the South African Institute for Computer Scientists and Information Technologists Conference*. ACM, 2012, pp. 69–78.

[39] L. Giordano, V. Gliozzi, N. Olivetti, and G. L. Pozzato, "A non-monotonic description logic for reasoning about typicality," *Artificial Intelligence*, vol. 195, pp. 165–202, 2013.

[40] Y. Ma, P. Hitzler, and Z. Lin, "Algorithms for paraconsistent reasoning with owl," in *The Semantic Web: Research and Applications*. Springer, 2007, pp. 399–413.

[41] Y. Ma and P. Hitzler, "Paraconsistent reasoning for owl 2," in *Web Reasoning and Rule Systems*. Springer, 2009, pp. 197–211.

[42] X. Zhang, G. Xiao, Z. Lin, and J. V. den Bussche, "Inconsistency-tolerant reasoning with OWL DL," *Int. J. Approx. Reasoning*, vol. 55, no. 2, pp. 557–584, 2014. [Online]. Available: http://dx.doi.org/10.1016/j.ijar.2013.09.005

[43] F. Maier, Y. Ma, and P. Hitzler, "Paraconsistent owl and related logics," *Semantic Web*, vol. 4, no. 4, pp. 395–427, 2013.

[44] O. Arieli and A. Avron, "Reasoning with logical bilattices," *Journal of Logic, Language and Information*, vol. 5, no. 1, pp. 25–63, 1996.

[45] ——, "The value of the four values," *Artif. Intell.*, vol. 102, no. 1, pp. 97–141, 1998. [Online]. Available: http://dx.doi.org/10.1016/S0004-3702(98)00032-0

[46] S. Abiteboul, R. Hull, and V. Vianu, *Foundations of databases*. Addison-Wesley Reading, 1995, vol. 8.

[47] S. Tobies, "Complexity results and practical algorithms for logics in knowledge representation," *arXiv preprint cs/0106031*, 2001.

[48] S. S. Sahoo and K. Thirunarayan, "Tableau algorithm for concept satisfiability in description logic alch," *Kno. e. sis Center, Tech. Rep*, 2009.

[49] P. Hitzler, M. Krötzsch, B. Parsia, P. F. Patel-Schneider, and S. Rudolph, "Owl 2 web ontology language primer," *W3C recommendation*, vol. 27, no. 1, p. 123, 2009.

[50] A. Paschke, "Rules and logic programming for the web," in *Reasoning Web. Semantic Technologies for the Web of Data*. Springer, 2011, pp. 326–381.

[51] A. Krisnadhi, F. Maier, and P. Hitzler, "Owl and rules," in *Reasoning Web. Semantic Technologies for the Web of Data*. Springer, 2011, pp. 382–415.

[52] H. Boley and M. Kifer, "Rif basic logic dialect," *W3C Working Draft (July 2009)*, 2009.

[53] H. Boley, G. Hallmark, M. Kifer, A. Paschke, A. Polleres, and D. Reynolds, "Rif core dialect," *W3C recommendation*, vol. 22, 2010.

[54] I. Horrocks and P. F. Patel-Schneider, "A proposal for an owl rules language," in *Proceedings of the 13th international conference on World Wide Web*. ACM, 2004, pp. 723–731.

[55] I. Horrocks, P. F. Patel-Schneider, S. Bechhofer, and D. Tsarkov, "Owl rules: A proposal and prototype implementation," *Web Semantics: Science, Services and Agents on the World Wide Web*, vol. 3, no. 1, pp. 23–40, 2005.

[56] I. Horrocks, P. F. Patel-Schneider, H. Boley, S. Tabet, B. Grosof, M. Dean *et al.*, "Swrl: A semantic web rule language combining owl and ruleml," *W3C Member submission*, vol. 21, p. 79, 2004.

[57] B. Motik, "Reasoning in description logics using resolution and deductive databases." Ph.D. dissertation, Karlsruhe Institute of Technology, 2006.

[58] B. Motik, U. Sattler, and R. Studer, "Query answering for owl-dl with rules," *Web Semantics: Science, Services and Agents on the World Wide Web*, vol. 3, no. 1, pp. 41–60, 2005.

[59] ——, "Query answering for owl-dl with rules," in *The Semantic Web–ISWC 2004*. Springer, 2004, pp. 549–563.

[60] R. Volz, "Web ontology reasoning with logic databases," Ph.D. dissertation, Karlsruhe, Univ., Diss., 2004, 2004.

[61] B. N. Grosof, I. Horrocks, R. Volz, and S. Decker, "Description logic programs: Combining logic programs with description logic," in *Proceedings of the 12th international conference on World Wide Web*. ACM, 2003, pp. 48–57.

[62] F. M. Donini, M. Lenzerini, D. Nardi, and A. Schaerf, "A hybrid system with datalog and concept languages," in *Trends in Artificial Intelligence*. Springer, 1991, pp. 88–97.

[63] ——, "Al-log: Integrating datalog and description logics," *Journal of Intelligent Information Systems*, vol. 10, no. 3, pp. 227–252, 1998.

[64] A. Y. Levy and M.-C. Rousset, "Carin: A representation language combining horn rules and description logics'," in *ECAI*. Citeseer, 1996, pp. 323–327.

[65] ——, "Combining horn rules and description logics in carin," *Artificial Intelligence*, vol. 104, no. 1, pp. 165–209, 1998.

[66] R. Rosati, "Towards expressive kr systems integrating datalog and description logics: preliminary report." *Description Logics*, vol. 22, 1999.

[67] ——, "On the decidability and complexity of integrating ontologies and rules," *Web Semantics: Science, Services and Agents on the World Wide Web*, vol. 3, no. 1, pp. 61–73, 2005.

[68] ——, "Semantic and computational advantages of the safe integration of ontologies and rules," in *Principles and Practice of Semantic Web Reasoning*. Springer, 2005, pp. 50–64.

[69] ——, "Dl+ log: Tight integration of description logics and disjunctive datalog." *KR*, vol. 6, pp. 68–78, 2006.

[70] ——, "Integrating ontologies and rules: Semantic and computational issues," in *Reasoning Web*. Springer, 2006, pp. 128–151.

[71] U. Hustadt, B. Motik, and U. Sattler, “Data complexity of reasoning in very expressive description logics,” in *IJCAI*, vol. 5, 2005, pp. 466–471.

[72] M. Krötzsch, S. Rudolph, and P. Hitzler, “Complexity boundaries for horn description logics,” in *AAAI*, vol. 7, 2007, pp. 452–457.

[73] B. Motik, I. Horrocks, R. Rosati, and U. Sattler, “Can owl and logic programming live together happily ever after?” in *The Semantic Web-Iswc 2006*. Springer, 2006, pp. 501–514.

[74] B. Motik and R. Rosati, “Closing semantic web ontologies,” Technical report, University of Manchester, UK, Tech. Rep., 2006.

[75] ——, “A faithful integration of description logics with logic programming.” in *IJCAI*, vol. 7, 2007, pp. 477–482.

[76] ——, “Reconciling description logics and rules,” *Journal of the ACM (JACM)*, vol. 57, no. 5, p. 30, 2010.

[77] T. Lukasiewicz, “A novel combination of answer set programming with description logics for the semantic web,” in *The Semantic Web: Research and Applications*. Springer, 2007, pp. 384–398.

[78] B. Motik, B. C. Grau, I. Horrocks, and U. Sattler, “Representing ontologies using description logics, description graphs, and rules,” *Artificial Intelligence*, vol. 173, no. 14, pp. 1275–1309, 2009.

[79] ——, “Representing structured objects using description graphs.” in *KR*, 2008, pp. 296–306.

[80] ——, “Modeling ontologies using owl, description graphs, and rules.” in *OWLED*, 2008.

[81] B. Motik, B. Cuenca Grau, and U. Sattler, “Structured objects in owl: Representation and reasoning,” in *Proceedings of the 17th international conference on World Wide Web*. ACM, 2008, pp. 555–564.

[82] N. C. Da Costa *et al.*, "On the theory of inconsistent formal systems." *Notre dame journal of formal logic*, vol. 15, no. 4, pp. 497–510, 1974.

[83] R. Bagai and R. Sunderraman, "Bottom-up computation of the fitting model for general deductive databases," *Journal of Intelligent Information Systems*, vol. 6, no. 1, pp. 59–75, 1996.

[84] I. Horrocks, O. Kutz, and U. Sattler, "The even more irresistible $\mathcal{SROIQ}$." *KR*, vol. 6, pp. 57–67, 2006.

[85] R. Bagai, "Tuple relational calculus for paraconsistent databases," in *Advances in Artificial Intelligence*. Springer, 2000, pp. 409–416.