Reasoning with Inconsistent Ontologies: a general framework

Zhisheng Huang, Frank van Harmelen, Annette ten Teije, Perry Groot, and Cees Visser
(Vrije Universiteit Amsterdam)

Abstract.
EU-IST Integrated Project (IP) IST-2003-506826 SEKT
Deliverable D3.4.1.1 (WP3.4)
This document is an informal deliverable provided to SEKT WP3 partners. In this document, a general framework for reasoning with inconsistent ontologies is proposed. An inconsistency reasoner is one which is able to return meaningful answers to queries, given an inconsistent ontology. The formal definitions of soundness, meaningfulness, local completeness, and maximal completeness of an inconsistency reasoner are introduced. A pre-processing algorithm and a strategy for inconsistency reasoning based on linear extensions and selection functions are investigated. This document also contains a chapter that discusses the design of reasoners with inconsistent ontologies (RIO). A RIO architecture, which is based on the DIG description logic interface, is proposed.
Keyword list: ontology management, inconsistent ontologies, reasoning

Document Id.	SEKT/2004/D3.4.1.1/v1.0
Project	SEKT EU-IST-2003-506826
Date	June 30, 2004
Distribution	internal

SEKT Consortium

This document is part of a research project partially funded by the IST Programme of the Commission of the European Communities as project number IST-2003-506826.

British Telecommunications plc.
Orion 5/12, Adastral Park
Ipswich IP5 3RE
UK
Tel: +44 1473 609583, Fax: +44 1473 609832
Contactperson: John Davies
E-mail: john.nj.davies@bt.com

Jozef Stefan Institute
Jamova 39
1000 Ljubljana
Slovenia
Tel: +386 1 4773 778, Fax: +386 1 4251 038
Contactperson: Marko Grobelnik
E-mail: marko.grobelnik@ijs.si

University of Sheffield
Department of Computer Science
Regent Court, 211 Portobello St.
Sheffield S1 4DP
UK
Tel: +44 114 222 1891, Fax: +44 114 222 1810
Contactperson: Hamish Cunningham
E-mail: hamish@dcs.shef.ac.uk

Intelligent Software Components S.A.
Francisca Delgado, 11 - 2
28108 Alcobendas
Madrid
Spain
Tel: +34 913 349 797, Fax: +49 34 913 349 799
Contactperson: Richard Benjamins
E-mail: rbenjamins@isoco.com

Ontoprise GmbH
Amalienbadstr. 36
76227 Karlsruhe
Germany
Tel: +49 721 50980912, Fax: +49 721 50980911
Contactperson: Hans-Peter Schnurr
E-mail: schnurr@ontoprise.de

Vrije Universiteit Amsterdam (VUA)
Department of Computer Sciences
De Boelelaan 1081a
1081 HV Amsterdam
The Netherlands
Tel: +31 20 444 7731, Fax: +31 84 221 4294
Contactperson: Frank van Harmelen
E-mail: frank.van.harmelen@cs.vu.nl

Empolis GmbH
Europaallee 10
67657 Kaiserslautern
Germany
Tel: +49 631 303 5540, Fax: +49 631 303 5507
Contactperson: Ralph Traphöner
E-mail: ralph.traphoener@empolis.com

University of Karlsruhe, Institute AIFB
Englerstr. 28
D-76128 Karlsruhe
Germany
Tel: +49 721 608 6592, Fax: +49 721 608 6580
Contactperson: York Sure
E-mail: sure@aifb.uni-karlsruhe.de

University of Innsbruck
Institute of Computer Science
Techikerstraße 13
6020 Innsbruck
Austria
Tel: +43 512 507 6475, Fax: +43 512 507 9872
Contactperson: Jos de Bruijn
E-mail: jos.de-bruijn@deri.ie

Kea-pro GmbH
Tal
6464 Springen
Switzerland
Tel: +41 41 879 00, Fax: 41 41 879 00 13
Contactperson: Tom Bösser
E-mail: tb@keapro.net

Sirma AI EOOD (Ltd.)
135 Tsarigradsko Shose
Sofia 1784
Bulgaria
Tel: +359 2 9768, Fax: +359 2 9768 311
Contactperson: Atanas Kiryakov
E-mail: naso@sirma.bg

Universitat Autonoma de Barcelona
Edifici B, Campus de la UAB
08193 Bellaterra (Cerdanyola del Vallès)
Barcelona
Spain
Tel: +34 93 581 22 35, Fax: +34 93 581 29 88
Contactperson: Pompeu Casanovas Romeu
E-mail: pompeu.casanovasquab.es

Changes

Version	Date	Author	Changes
0.3	12.5.04	Zhisheng Huang	Creation
0.4	10.6.04	Zhisheng Huang	First Draft
0.5	27.6.04	Perry Groot	Second Draft
0.6	30.6.04	Zhisheng Huang	Some Changes
1.0	30.6.04	Zhisheng and Cees	Final Draft

Executive Summary

This document is an informal deliverable provided to SEKT WP3 partners.

The classical entailment in logics is *explosive*: any formula is a logical consequence of a contradiction. Therefore, conclusions drawn from an inconsistent ontology by classical inference may be completely meaningless. An inconsistency reasoner is one which is able to return meaningful answers to queries, given an inconsistent ontology.

In this document, we overview reasoning with inconsistency in logics and AI, in particular, in paraconsistent logics and approximation reasoning. Furthermore, we examine several typical examples and scenarios of inconsistency in the context of the Semantic Web. Based on the technology overview, we propose a general framework for reasoning with inconsistent ontologies. We present the formal definitions of soundness, meaningfulness, local completeness, and maximal completeness of an inconsistency reasoner. We propose and investigate a pre-processing algorithm, discuss the strategies of inconsistency reasoning based on pre-defined selection functions dealing with concept relevance.

In this document, we also discuss how the syntactic relevance can be used for reasoning with inconsistent ontologies. We also briefly discuss how particular semantic relevance approaches which have been used in computational linguistics can be applied to this task.

In addition, we propose an architecture for systems which support client side and server side reasoning with inconsistent ontologies, based on the DIG description logic interface.

Contents

1 Introduction **2**

2 Inconsistency in the Semantic Web **5**
2.1 Inconsistency by Mis-presentation of Default Rules 5
2.2 Inconsistency Caused by Polysemy 6
2.3 Inconsistency through Migration from Another Formalism 7
2.4 Inconsistency Caused by Multiple Sources 7

3 Reasoning with Inconsistent ontologies **9**
3.1 Formal Definitions 9
3.2 Pre-processing 11
3.3 Inconsistency Processing 12
3.3.1 Linear extension 13
3.4 Inconsistency Processing with Approximate Reasoning 15

4 Selection Functions **19**
4.1 Syntactic Relevance 19
4.2 Semantic Relevance 21

5 Design of Reasoning with Inconsistent Ontologies **23**
5.1 General Consideration 23
5.2 Architecture 25
5.3 Implementation 25

6 Discussion and Conclusions **26**

Chapter 1

Introduction

The Semantic Web is characterized by scalability, distribution, and multi-authorship. All these characteristics may introduce inconsistencies in the Semantic Web. Limiting language expressivity with respect to negation (like RDF and other languages that are based on negation as failure) can avoid inconsistencies to a certain extent. However, for specifications the expressivity of these languages is quite limited.

There are two main ways to deal with inconsistency. One is to diagnose and repair it when we encounter inconsistencies. In [23] Schlobach and Cornet propose a non-standard reasoning service for debugging inconsistent terminologies. Another one is to simply avoid it and to apply a non-standard reasoning method to obtain meaningful answers. In this report, we will focus on the latter.

We will consider ontology specifications for which their logical foundations are based on Description Logics [2] and OWL, the Web counterpart of Description Logics. Semantically, description logics can be considered as a subset of first order logic. The classical entailment in logics is *explosive*: any formula is a logical consequence of a contradiction. Therefore, conclusions drawn from an inconsistent knowledge base by classical inference may be completely meaningless.

In this report, we propose a general framework for reasoning with inconsistent ontologies. We investigate how an inconsistency reasoner can be developed for the Semantic Web. The general task of an inconsistency reasoner is: given an inconsistent ontology, return meaningful answers to queries.

Reasoning with inconsistency is an old topic in logics and AI. Many approaches have been proposed to deal with inconsistency [5, 6, 19, 17]. The development of paraconsistent logics was initiated to challenge the 'explosive' problem of the standard logics. Paraconsistent logics [6] allow theories that are inconsistent but non-trivial. There are many different paraconsistent logics. Most of them are defined on a semantics which allows both a letter and its negation to hold for an interpretation. Levesque's limited reference[18] allows the interpretation of a language in which a truth assignment may map both a letter l and its negation $\neg l$ to true. Extending the idea of Levesque's limited

reference, Schaerf and Cadoli propose S-3-entailment and S-1-entailment for approximate reasoning with tractable results. The main idea of Schaerf and Cadoli's approach is to introduce a subset S of the language used, which can be used as a parameter in their framework and allows their reasoning procedure to focus on a part of the theory while the remaining part is ignored. We will refer to the subset S as an approximation set. The approximation set S can be extended to make their reasoning procedure more classical. However, how to construct and extend the approximation sets is still an open question.

Based on Schaerf and Cadoli's S-3-entailment, Marquis and Porquet present a framework for reasoning with inconsistency by introducing a family of resource-bounded paraconsistent inference relations [19]. In Marquis and Porquet's approach, consistency is restored by removing variables from the approximation set S instead of removing some explicit beliefs from the belief base, like the standard approaches do in belief revision. Their framework enables some forms of graded paraconsistency by explicit handling of preferences over the approximation set S. In [19], Marquis and Porquet propose several policies, e.g., the linear order policy and the lexicographic policy, for the preference handling in paraconsistent reasoning. In [9], Chopra, Parikh, and Wassermann incorporate the local change of belief revision and relevance sensitivity by means of Schaerf and Cadoli's approximation reasoning, and show how relevance can be introduced for approximate reasoning in belief revision. Both approaches, Marquies', and Chopra's, depend on syntactic selection procedures for extending the approximation set.

Our approach borrows some ideas from Schaerf and Cadoli's approximation approach, Marquis and Porquet's paraconsistent reasoning approach, and Chopra, Parikh, and Wassermann's relevance approach. However, our main idea is relatively *simple* (It is not a bad thing, isn't?): given a selection function, which can be defined on the syntactic or semantic relevance, like those have been used in computational linguistics, we can always select some consistent sub-theory from an inconsistent ontology. Then we apply standard reasoning on the selected sub-theory to find meaningful answers. If it cannot give a satisfying answer, the selection function would loose the relevance degree to extend consistent sub-theory for further reasoning. We believe that our approach is intuitive and efficient, just like human beings, who can deal with inconsistency very efficiently. When we encounter inconsistency, we can always focus on the important and consistent part of it, by ignoring other parts which may cause inconsistency. Then we apply standard reasoning to the selected part to obtain the solutions. Moreover, we believe that the importance here can always be measured, or exactly interpreted, as some degree of relevance among the concepts. In general, the solution for reasoning with inconsistency is to use nonstandard reasoning to deal with inconsistency.

This report is organized as follows: Chapter 2 overviews inconsistency in the Semantic Web by examining several typical examples and scenarios. Chapter 3 proposes a general framework of reasoning with inconsistent ontologies. Chapter 4 examines how the selection function can be developed, based on the syntactic relevance approach. We also discuss briefly how the semantic relevance approach can be developed for the task. Chapter 5 presents a design for reasoning with inconsistent ontologies. In particular, an

architecture which is based on the DIG interface for DL reasoning, is proposed. Chapter 6 discusses further work and concludes the report.

Chapter 2

Inconsistency in the Semantic Web

For the Semantic Web, inconsistencies easily occur, sometimes even in small ontologies. Here are several scenarios which may cause inconsistencies:[1]

2.1 Inconsistency by Mis-presentation of Default Rules

When a knowledge engineer specifies an ontology statement, in particular a rule, she/he has to check carefully that the new statement is consistent, not only with respect to existing rules, but also with respect to rules that may be added in the future, which of course may not always be known at that moment. This makes it very difficult to maintain the consistency in ontology specifications. Just consider a situation in which a knowledge engineer wants to create an ontology about animals:

$bird \sqsubseteq animal$ (Birds are animals),
$bird \sqsubseteq fly$ (Birds are flying animals).

Although the knowledge engineer may realize that 'birds can fly' is not generally valid, he still wants to add it if he does not find any counterexample in the current knowledge base, because flying is one of the main features of birds. An ontology about birds without talking about flying is not satisfactory.

Later on, one may want to extend the ontology with the following statements:

[1]By inconsistency we mean that the set implies a contradiction, i.e., the set $\Sigma \models \bot$, given an entailment relation '$\models$'. It is usually called *incoherence*. We continue using the term 'inconsistency', because it is common practice in the ontology community.

$eagle \sqsubseteq bird$ (Eagles are birds),
$penguin \sqsubseteq bird$ (Penguins are birds),
$penguin \sqsubseteq \neg fly$ (Penguins are not flying animals).

The concept $penguin$ in that ontology of birds is already unsatisfiable, because it implies penguins can both fly and not fly. That would lead to an inconsistent ontology. One may remove the rule 'birds can fly' from the existing ontology to restore the consistency. However, this approach is not reliable, because of the following reasons: a) It is hard to check that the removal would not cause any significant information loss in the current ontology, b) One may not have the authority to remove statements which have been created in the current knowledge base, c) It may be difficult to know which part of the existing ontology can be removed if the knowledge base is very large. One would not blame the knowledge engineer for the creation of the rule 'birds are flying animals' at the beginning without considering future extensions, because it is hard for the knowledge engineer to do so.

One may argue that the current ontology languages and their counterparts in the Semantic Web cannot be used to handle this kind of problems, because it requires non-monotonic reasoning. The statement *Birds can fly* has to be specified as a default rule. The ontology language OWL cannot deal with default rules. We have to wait for an extension of OWL to accommodate non-monotonic rules. It is painful that we cannot talk about birds (that can fly) and penguins (that cannot fly) in the same ontology specification. An alternative approach is to divide the inconsistent ontology specification into multiple ontologies or modular ontologies to maintain their local consistency, like one that states 'birds can fly', but doesn't talk about penguins, and another one that specifies penguins, but never mentions that 'birds can fly'. However, the problem for this approach is still the same as other ones. Again, an ontology about birds that cannot talk about both 'birds can fly' and penguins is not satisfactory.

2.2 Inconsistency Caused by Polysemy

Polysemy refers to the concept of words with multiple meanings. One should have a clear understanding of all the concepts when an ontology is formally specified. Here is an example of an inconsistent ontology which is caused by polysemy:

$marriedWoman \sqsubseteq woman$	(a married woman is a woman),
$marriedWoman \sqsubseteq \neg divorcee$	(a married woman is not a divorcee),
$divorcee \sqsubseteq hadHusband \sqcap \neg hasHusband$	(a divorcee had a husband and has no husband),
$hasHusband \sqsubseteq marriedWoman$	(hasHusband means married),
$hadHusband \sqsubseteq marriedWoman$	(hadHusband means married).

In the ontology specification above, the concepts 'divorcee' is unsatisfiable, because of the misuse of the word 'marriedWoman'. Therefore, one has to carefully check if there is some misunderstanding with respect to concepts that has been used in the ontology. Again, when an ontology is large, this kind of requirements may become rather difficult.

2.3 Inconsistency through Migration from Another Formalism

When an ontology specification is migrated from other data sources, inconsistencies may occur. As it has been found by Schlobach and Cornet in [23], the high number of unsatisfiable concepts in DL terminology for DICE is due to the fact that it has been created by migration from a frame-based terminological system.[2] In order to make the semantics as explicit as possible, a very restrictive translation has been chosen to highlight as many ambiguities as possible. In [23], Schlobach and Cornet show the following inconsistent ontology specification:

$brain \sqsubseteq centralNervousSystem$	(a brain is a central nervous system),
$brain \sqsubseteq bodyPart$	(a brain is a body part),
$centralNervousSystem \sqsubseteq nervousSystem$	(a central nervous system is a nervous system),
$bodyPart \sqsubseteq \neg nervousSystem$	(a body part is not a nervous system).

2.4 Inconsistency Caused by Multiple Sources

When a large ontology specification is generated from multiple sources, in paricular when these sources are created by several authors, inconsistencies easily occur.

[2] DICE stands for "Diagnoses for Intensive Care Evaluation". The development of the DICE terminology has been supported by the NICE foundation.

In [12], Hameed, Preece and Sleeman propose approaches of ontology reconciliation, and discuss how the consistency should be maintained and how inconsistency may be created from multiple sources. Accoring to [12], there are the following three possibilities for ontology reconciliation: merging, aligning, or integrating:

Merging implies the creation of a new ontology by unifying several other ontologies into a single one. The new ontology is created from two or more existing ontologies with overlapping parts, and can be either virtual or physical. The ultimate goal is to create a single consistent ontology that includes all information from all the sources.

Aligning is to keep ontologies separately when sources must be made consistent with one another. It involves bringing two or more ontologies into mutual agreement, making them consistent.

Integrating is to build a new ontology by composing parts of other available ontologies. Like merging, this process results in a new ontology.

No matter whether a new ontology is generated by merging or integrating multiple sources, in both cases general consistency objectives are rather difficult to achieve.

Note that the above-mentioned categories don't exclude each other. When we examine an inconsistent ontology which is generated from multiple sources, we may find that it contains several cases of polysemy, or some other inconsistency. The list above is also not exhaustive. There are many other cases that may cause the inconsistency, like inconsistency caused by ambiguities, inconsistency caused by lacking global checking, etc. We do not discuss a complete list in this report.

Chapter 3

Reasoning with Inconsistent ontologies

3.1 Formal Definitions

This report aims at a proposal of a general framework for reasoning with inconsistent ontologies. Therefore, we do not restrict ontology specifications to a particular language (although OWL and its description logic are the languages we have in mind). In general, an ontology language can be considered to be a set that is generated by a set of syntactic rules. We use a non-classical entailment for inconsistency reasoning. In the following, we use $\models$ to denote the classical entailment, and use $\mid\!\approx$ to denote some non-standard inference relation, which may be parameterized to remove ambiguities.

In general, an ontology query Σ can be expressed as '$\Sigma \mid\!\approx \phi$?', where ϕ is a formula. There are two standard answers to a query, either "yes" ($\Sigma \mid\!\approx \phi$), or "no" ($\Sigma \not\mid\!\approx \phi$). In the next section, we will argue that it would be more suitable for an inconsistency reasoner to extend the query answers to three possible answers: "positive" ($\Sigma \mid\!\approx \phi$ and $\Sigma \not\mid\!\approx \neg\phi$), "negative" ($\Sigma \mid\!\approx \neg\phi$ and $\Sigma \not\mid\!\approx \phi$), or "unknown" ($\Sigma \not\mid\!\approx \phi$ and $\Sigma \not\mid\!\approx \neg\phi$).

An inconsistency reasoner is one which is expected to be able to return meaningful answers to queries, given an inconsistent ontology. In the following we propose several formal definitions for inconsistency ontology reasoners.

Soundness: In the case of a consistent ontology Σ, classical reasoning is sound, i.e., a formula ϕ deduced from Σ holds in every model of Σ. This definition is not preferable for an inconsistent ontology Σ as every formula follows from it using using classical entailment. However, often only a small part of Σ has been incorrectly constructed or modelled, while the remainder of Σ is correct. This leads us to the following: an inconsistency reasoner should be considered correct if the formulas that follow from an inconsistent theory Σ follow from a consistent subtheory of Σ using classical reasoning. Therefore, we propose the following definition of

soundness. An inconsistency reasoner $\not\approx$ is sound if the following condition holds:

$$\Sigma \not\approx \phi \Rightarrow (\exists \Sigma' \subseteq \Sigma)(\Sigma' \not\models \bot \text{ and } \Sigma' \models \phi).$$

Note however, that in the previous definition the implication should *not hold* in the opposite direction. If the implication would also hold in the opposite direction it would lead to an inconsistency reasoner, which returns inconsistent answers. For example if $\{a, \neg a\} \subseteq \Sigma$, then the inconsistency reasoner would return that both a and $\neg a$ hold given Σ, which is something we would like to prevent. Hence, the inconsistency reasoner should not return answers that follow from *any* consistent subset of Σ, but from *specifically chosen* subsets of Σ. This 'specific selection of consistent subsets' is part of the inconsistent reasoners strategy and will be discussed in more detail in Section 3.3.

Meaningfulness: An answer given by an inconsistency reasoner is meaningful iff it is consistent and sound. Namely, it requires not only the soundness condition, but also the following condition:

$$\Sigma \not\approx \phi \Rightarrow \Sigma \not\approx \neg\phi.$$

Inconsistency reasoning is said to be meaningful iff all of the answers are meaningful.

Local Completeness: Because of inconsistencies, global completeness is impossible. We suggest the notion of local completeness: inconsistency reasoning is locally complete with respect to a consistent sub-theory Σ' iff for any formula ϕ, the following condition holds:

$$\Sigma' \models \phi \Rightarrow \Sigma \not\approx \phi.$$

Alternatively, the condition can be represented as:

$$\Sigma \not\approx \phi \Rightarrow \Sigma' \not\approx \phi.$$

Therefore, local completeness can be considered as a complement to the soundness property. An answer to a query $\Sigma \not\approx \phi?$ is said to be locally complete with respect to a consistent set Σ' iff the following condition holds:

$$\Sigma' \models \phi \Rightarrow \Sigma \not\approx \phi.$$

Maximality: An inconsistency reasoner is maximal iff there is a maximal consistent sub-theory such that its consequence set is the same as the consequence set of the inconsistency reasoner:

$$\exists(\Sigma' \subseteq \Sigma)(\Sigma' \not\models \bot \wedge (\forall \Sigma'' \supset \Sigma' \wedge \Sigma'' \subseteq \Sigma)(\Sigma'' \models \bot) \wedge \forall \phi(\Sigma' \models \phi \Leftrightarrow \Sigma \not\approx \phi)).$$

We use the same condition to define the maximality for an answer, like we do for local completeness.

Local Soundness: An answer to a query '$\Sigma \not\approx \phi?$' is said to be locally sound/correct with respect to a consistent set $\Sigma' \subseteq \Sigma$, iff the following condition holds:

$$\Sigma \not\approx \phi \Rightarrow \Sigma' \models \phi.$$

Namely, for any positive answer, it should be implied by the given consistent subtheory Σ' under the standard entailment.

From the definitions given above it follows that local soundness implies soundness and meaningfulness. Moreover, it follows that maximal completeness implies local completeness. Given a query, there might exist more than one maximal consistent subset. The same also holds for local completeness. Therefore, arbitrary (maximal) consistent subset may not be very useful for the evaluation of a query by some inconsistency reasoner. The consistent subsets should be chosen on structural or semantical grounds indicating the relevance of the chosen subset with respect to some query. Functions that will select specific subsets of an ontology Σ will be discussed in more detail in Chapter 4.

3.2 Pre-processing

With classical reasoning, a query ϕ given an ontology Σ can be expressed as an evaluation of the consequence relation $\Sigma \models \phi$. There are only two answers to that query: either "yes" ($\Sigma \models \phi$), or "no" ($\Sigma \not\models \phi$). A "yes" answer means that ϕ is a logical conclusion of Σ. A "no" answer, however, means that ϕ cannot be deduced from Σ, because we usually don't adopt the closed world assumption when using an ontology. Hence, a "no" answer does not imply that the negation of ϕ holds given an ontology Σ.

The evaluation of the entailment relation is usually achieved by reducing it to the satisfiablity of the formula set $\Sigma \cup \{\neg\phi\}$, because of the following relation:

$$\Sigma \models \phi \text{ iff } \Sigma \cup \{\neg\phi\} \text{ is not satisfiable.}$$

If the ontology Σ is inconsistent, it is never satisfiable. Hence, for any query ϕ, $\Sigma \cup \{\neg\phi\}$ is never satisfiable. Therefore, any query ϕ is a conclusion of an inconsistent ontology. Classical reasoning is said to be *explosive*, i.e., any formula may be derived from an inconsistent ontology. Hence, classical reasoning leads to completely meaningless answers when using an inconsistent ontology.

To make sure reasoning is reliable when it is unclear if an ontology is consistent or not, we can use the decision tree that is depicted in Figure 3.1. For a query ϕ we test both the consequences $\Sigma \models \phi$ and $\Sigma \models \neg\phi$ using classical reasoning. In case different answers are obtained, i.e., both "yes" and "no", the ontology Σ must be consistent and the answer to $\Sigma \models \phi$ can be returned. In case of two answers that are the same the ontology is either incomplete, i.e., when both answers are "no", or the ontology is inconsistent, i.e.,

when both answers are "yes". When the ontology turns out to be incomplete, either an "unknown" answer can be returned or additional information I can be gathered to answer the query $\Sigma \cup I \models \phi$, but this falls outside the scope of the project. When the ontology turns out to be inconsistent some inconsistency reasoner can be called upon to answer the query $\Sigma \mid\!\approx \phi$.

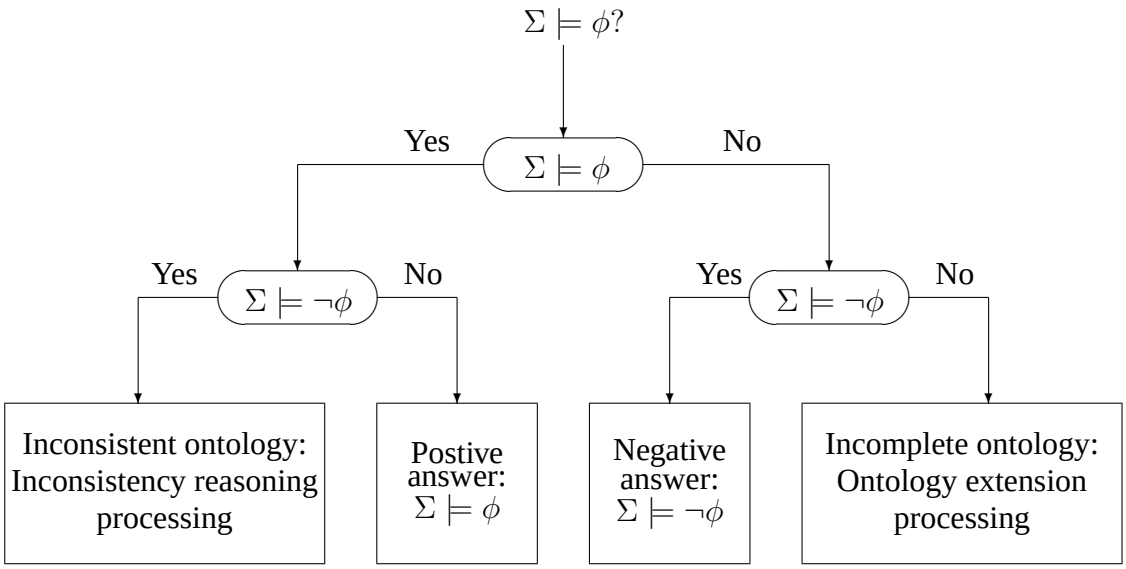

Figure 3.1: Decision tree for obtaining reliable reasoning with an inonsistent ontology.

3.3 Inconsistency Processing

It is assumed that a selection function can be used by an inconsistency reasoner to determine which consistent subsets of an inconsistent ontology should be considered in its reasoning process. The definition of a selection function should be independent of the general framework for reasoning with an inconsistent ontology as given in Figure 3.1. The selection function can either be based on a syntactic approach, like Chopra, Parikh, and Wassermann's syntactic relevance [9], or based on semantic relevance like for example in computational linguistics as in Wordnet. Selection functions will be discussed in more detail in Chapter 4.

Note however that a selection function does not have to return *all* subsets for consideration at the same time. If a query $\Sigma \mid\!\approx \phi$ can be answered after considering some consistent subset of the ontology Σ given by the selection function, other subsets don't have to be considered any more, because *they will not change the answer of the inconsistency reasoner*.

In this project we therefore always use an iterative approach. To answer a query $\Sigma \not\approx \phi$, first start with a consistent subset Σ' of the ontology Σ and try to answer the query $\Sigma' \models \phi$. If the answer is affirmative, the inconsistency reasoner returns "yes" to the query $\Sigma \not\approx \phi$. If the answer is not affirmative, call the selection function again to produce the next consistent subset of the ontology and repeat until there are no more subsets that can be used in the reasoning process. If there are no more consistent subsets to consider the inconsistency reasoner will return "no" or "unknown" to the query $\Sigma \not\approx \phi$.

Note that from a computational point of view it is better to use a "bottom-up" approach, i.e., first consider small subsets before considering larger subsets, because answering a query is computationally easier using a small subset of an ontology.

3.3.1 Linear extension

In this section we describe in more detail selection functions that are *monotone*:

Definition 3.3.1 *A selection function f is* monotone *if the consistent subsets that it selects $S_1, S_2, S_3, \ldots$ monotonically increase or decrease, e.g., $S_1 \subseteq S_2 \subseteq S_3 \cdots$.*

In this project we assume that a monotone selection function always returns increasing subsets. An inconsistency reasoner that uses a monotone selection function will be called an inconsistency reasoner that uses a linear extension strategy.

A linear extension strategy is carried out as shown in Figure 3.2. Given a query $\Sigma \not\approx \phi$, the initial consistent subset Σ' is set to be equal to the empty set $\emptyset$. Then the selection function is called to return a consistent subset Σ'' that subsumes Σ', i.e., $\Sigma' \subset \Sigma'' \subseteq \Sigma$. If the set Σ'' does not exist, the inconsistency reasoner returns the answer "unknown" to the query $\Sigma \not\approx \phi$. If the set Σ'' exists, a classical reasoner is used to check if $\Sigma \models \phi$ holds. If the answer is 'yes', the inconsistency reasoner returns the positive answer $\Sigma'' \models \phi$. If the answer is 'no', the inconsistency reasoner further checks the negation of the query $\Sigma'' \models \neg\phi$. If the answer is "yes", the inconsistency reasoner returns the negative answer $\Sigma \not\approx \neg\phi$, otherwise the whole process is repeated by calling the the selection function for the next consistent subset of Σ which extends Σ''.

It is clear that the linear extension strategy may result in too many "unknown" answers to queries when the selection function picks the wrong sequence of monotonically increasing subsets. It would therefore be useful to measure the succesfulness of (linear) extension strategies. Similarly with Marquis and Porquet's work in [19], we use Belnap's four valued logic [4] to distinguish the following four epistemic status for the extension measurements:

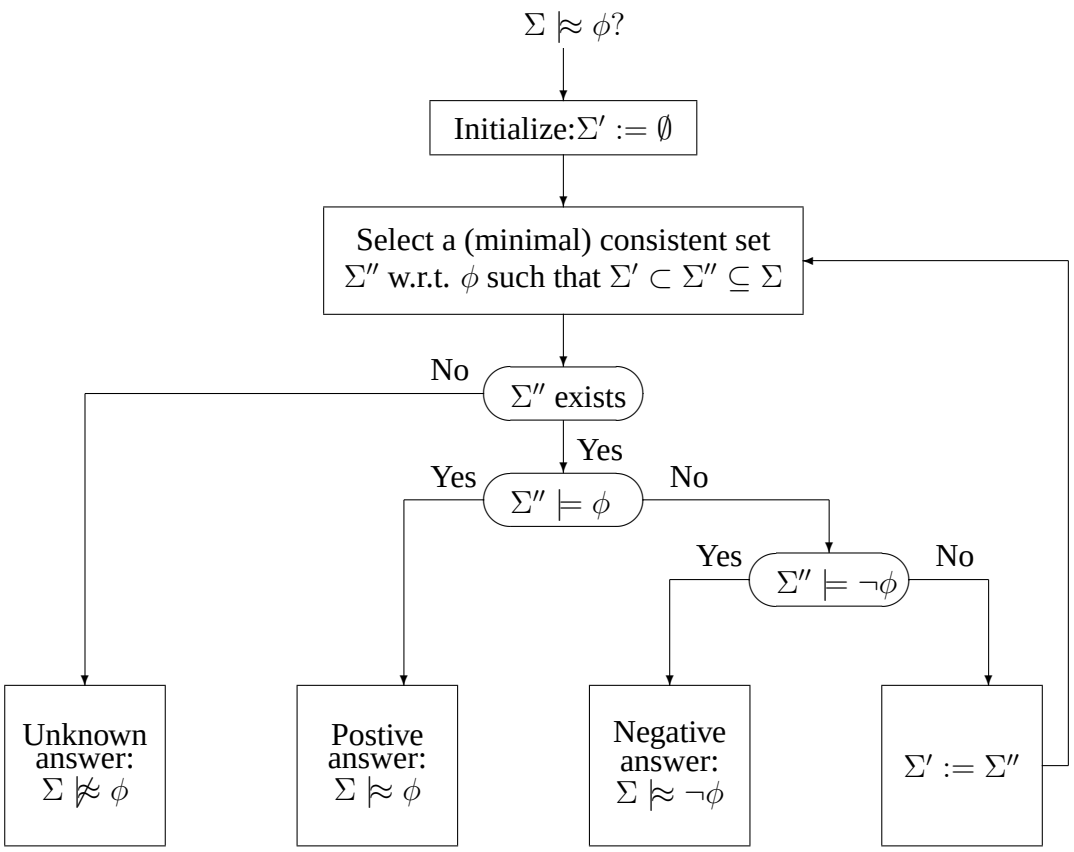

Figure 3.2: An inconsistency reasoner using the Linear Extension strategy.

Definition 3.3.2

- *Over-determined:* $\Sigma \mid\!\approx \phi$ *and* $\Sigma \mid\!\approx \neg\phi$.
- *Accepted:* $\Sigma \mid\!\approx \phi$ *and* $\Sigma \not\mid\!\approx \neg\phi$.
- *Rejected:* $\Sigma \not\mid\!\approx \phi$ *and* $\Sigma \mid\!\approx \neg\phi$.
- *Undetermined:* $\Sigma \not\mid\!\approx \phi$ *and* $\Sigma \not\mid\!\approx \neg\phi$.

In general, one should use an extension strategy that is not over-determined and not undetermined. For the linear extension strategy, we can prove that the following properties hold:

Proposition 3.3.1 (Linear Extension) *An inconsistency reasoner using a linear extension strategy satisfies the following properties:*

- *never over-determined,*

- *may be undetermined,*
- *always sound,*
- *always meaningful,*
- *always locally complete,*
- *may not be maximal,*
- *always locally sound.*

Therefore, an inconsistency reasoner using a linear extension strategy is useful to create meaningful and sound answers to queries. It is also locally complete with certain consistent sets. Unfortunately it may not be maximal. We call this strategy a *linear* one, because the selection function only follows one possible "extension chain" for creating consistent subsets. The advantages of the linear strategy is that the reasoner can always focus on the current working set Σ'. The reasoner doesn't need to keep track of the extension chain. The disadvantage of the linear strategy is that it may lead to an inconsistency reasoner that is undetermined. There exist other strategies which can improve the linear extension approach, for example, by backtracking and heuristics evaluation. These kind of techniques are frequently used in the search methods, which are traditional topics in artificial intelligence [21]. A better selection function would also be very useful to achieve better results. We discuss this issue briefly in the next chapter.

3.4 Inconsistency Processing with Approximate Reasoning

In [22], Schaerf and Cadoli propose S-3-entailment for approximate reasoning with tractable results. The main idea of Schaerf and Cadoli's approach is to introduce and extend a subset of the language used to focus on a part of the language while discarding other formulas.

Let $\mathcal{L}(\Phi_0)$ be a finite language which is generated on a primitive proposition set Φ_0, and S be a subset of Φ_0, called an *approximation set*. A sentence in a propositional language can always be rewritten as a formula with Negation Normal Form (NNF), in which the negation occurs only on letters.

Definition 3.4.1 (S-3-interpretation) *An S-3-interpretation of $\mathcal{L}(\Phi_0)$ is a truth assignment which maps every letter of S and its negation into opposite values. Moreover, it does not map both a letter of Φ_0/S and its negation into 0.*

Definition 3.4.2 (S-3-satisfiability) *A formula is S-3-satisfiable if an S-3-interpretation I exists such that I satisfies it. A formula set Σ is S-3-satisfiable iff an S-3-interpretation I exists and for all the formulas $\phi \in \Sigma$, $\Sigma \models_S^3 \phi$ holds.*

Definition 3.4.3 (S-3-entailment) *A formula set Σ S-3-entails a formula ϕ, denoted by $\Sigma \models_S^3$, iff every S-3-interpretation that satisfies Σ also satisfies ϕ.*

In [22], Schaerf and Cadoli show that S-3-entailment is sound and incomplete. The S-3-entailments can be used to achieve the approximation entailment. Moreover, the S-3-inference can tolerate the inconsistency by selecting the approximation set S. Thus, we can extend the general framework of the inconsistency processing with S-3-entailment. Ontology languages can be usually considered as some fragments of the first order logics. In the following, we consider the Herbrand base of the first-order theory for the approximation, similar with Schaerf and Cadoli's work in [22], which extends the theory from propositional logics to the first-order logic.

Let $atom(\phi)$ be the set of ground atoms of ϕ, i.e., the set of atoms in the Herbrand base of the formula ϕ, which serves as the initial approximation set. In the processing, we extend the atom set instead of the formula set. The procedure is shown in Figure 3.3.

In [22], Schaerf and Cadoli show that the approximation of classical reasoning can be achieved via a simplification of the formula set Σ as follows:

Proposition 3.4.1 (Schaerf and Cadoli 1995) *Let $simplify-3(\Sigma, S)$ be the result of deleting all clauses of Σ which contain an atom outside S. Σ is S-3-satisfiable iff $simplify-3(\Sigma, S)$ is classically satisfiable.*

Therefore, the selection on atomic formula sets for S-3-entailment is equivalent to the selection on the subset of a theory for the standard entailment. It is straightforward to have the following proposition on the linear strategy with S-3-entailment, based on the proposition 3.4.1 and the proposition 3.3.1.

Proposition 3.4.2 (Linear Extension with S-3-entailment) *The linear extension of the inconsistency reasoning with S-3-entailment is:*

- *never over-determined.*
- *may undetermined.*
- *always sound.*
- *always meaningful.*
- *always locally complete.*

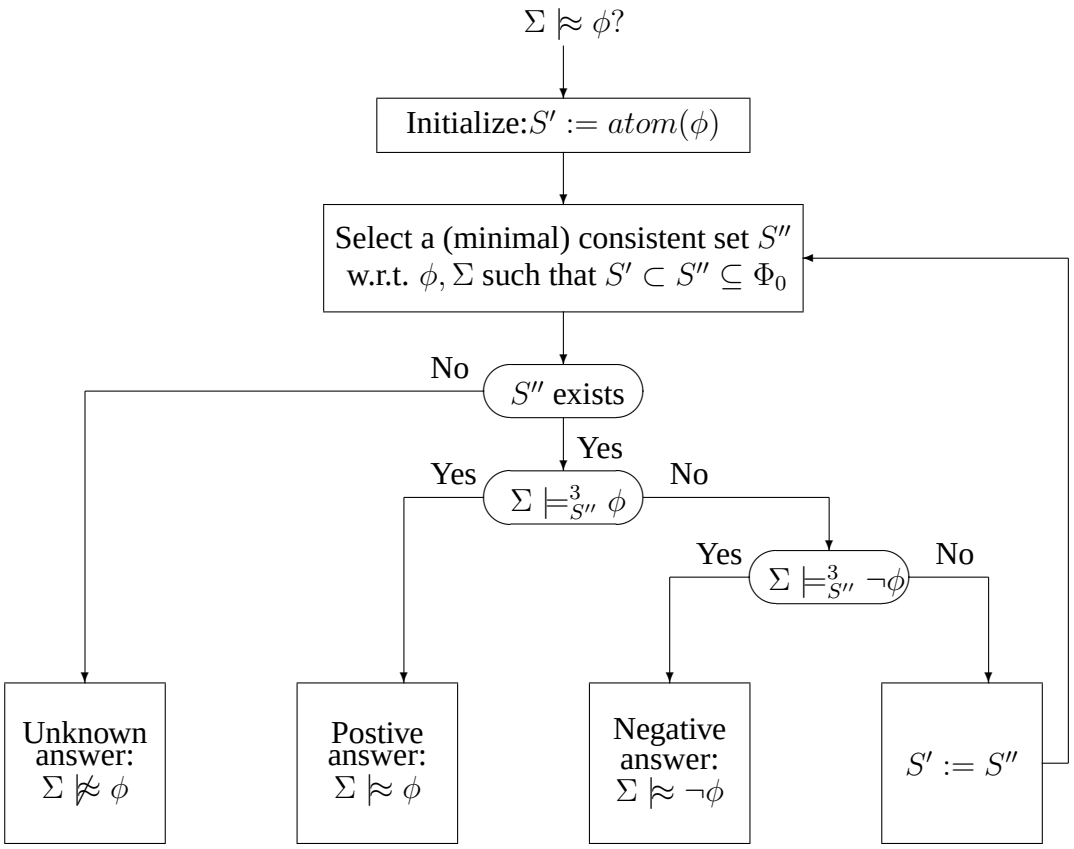

Figure 3.3: Reasoning with Inconsistency: Linear Extension with S-3-Entailment.

- *may not maximal.*
- *always locally sound.*

Reasoning about inconsistency with S-3-entailment is suitable for the selection function which is based on concept relevance. Moreover, it also reduces the complexity of the reasoning[22]. Note that S-3-entailment ignores the inconsistency by excluding it from the approximation set S. Sometimes it cannot deal with inconsistency properly. For instance, in the bird example, $\{bird, fly, penguin\}$ cannot be a subset of the approximation set S, otherwise it would cause an inconsistency.

Schaerf and Cadoli's S-3-approximation requires that formulas be the negation normal forms (NNF). In [19], Marquis and Proguet propose a framework of approximation reasoning which combines S-3-approximation and a paraconsistent logic J_3 which is based on 3-valued logic. Under Marquis and Proguet's framework, there is no need to restrict formulas to NNF.

Chapter 4

Selection Functions

As we have pointed out in Chapter 3, the definition of the selection function should be independent of the general procedure of the inconsistency processing. In the sequel report, we are going to investigate how the selection functions can be developed formally. However, in this report, we would like to point out that there exist several alternatives which can be used for an inconsistency reasoner. We discuss them in the following sections.

4.1 Syntactic Relevance

In [9], Chopra, Parikh, and Wassermann propose syntactic relevance to measure the relationship between two formulas in belief sets, so that the relevance can be used to guide the belief revision based on Schaerf and Cadoli's method of approximation reasoning.

Definition 4.1.1 (Direct relevance and k-relevance [9]) *Given a formula set Σ, two atoms p, q are directly relevant, denoted by $R(p, q, \Sigma)$ iff there is a formula $\alpha \in \Sigma$ such that p, q appear in α. A pair of atoms p and q are k-relevant with respect to Σ iff there exist $p_1, p_2, \ldots, p_k \in \mathcal{L}$ such that:*

- *p, p_1 are directly relevant;*
- *p_i, p_{i+1} are directly relevant, $i = 1, \ldots, k-1$;*
- *p_k, q are directly relevant.*

The notions of relevance are based on propositional logics. However, ontology languages are usually written in some subset of first order logic. It would not be too difficult to extend the ideas of relevance to those first-order logic-based languages by considering an atomic formula in first-order logic as a primitive proposition in propositional logic. In our sequel report, we investigate this extension of relevance definitions formally. However, in this report, we just adapt the idea informally.

In inconsistency reasoning we can use syntactic relevance to define a selection function to extend the query '$\Sigma \mid\!\approx \phi?$' as follows: we start with the set $atom(\phi)$ as a departure point for the selection based on syntactic relevance. First we consider the formula $\psi \in \Sigma$ such that $atom(\psi) \subseteq atom(\phi)$ to see whether or not they are sufficient to give an answer to the query, for we consider any letter to be most relevant to itself. If the reasoning process can obtain an answer to the query, it stops. Otherwise the selection function increases the relevance degree by adding to the selected formula set all formulas that are directly relevant to it. The selection procedure can be described as two working sets Σ' and Σ'' as shown in the linear extension algorithm. At the beginning, the first working set Σ' is defined with respect to the query ϕ and the formula set Σ as follows:

$$\Sigma' = \{\psi \in \Sigma | atom(\psi) \subseteq atom(\phi)\}.$$

The second working set is based on the first working set Σ' as follows:

$$\Sigma'' = \{\psi \in \Sigma | \forall p \in atom(\psi) \exists \phi \in \Sigma' \text{ such that } (\exists q \in atom(\phi) R(p, q, \Sigma))\}.$$

Consider the bird ontology. Some example follow of the linear extension strategy with a selection function based on the syntactic relevance approach.

We add a fact $penguin(tweety)$ (Tweety is a penguin) into the A-box of the ontology. Note that we need that fact to state that Penguin exists. Let Σ be the formula set of the bird ontology, as defined in Chapter 2. Thus, $\Sigma_1 = \Sigma \cup \{penguin(tweenty)\}$ is an inconsistent formula set. For the query '$\Sigma \mid\!\approx fly(tweety)?$' (can Tweety fly?), at the beginning, the selection function would pick up the formula set $\Sigma'' = \{bird \sqsubseteq fly, penguin \sqsubseteq \neg fly, penguin(tweety)\}$, based on the direction relevance to the letter set of the query, i.e., $atom(fly(tweety))$. We have

$$\Sigma'' \models \neg fly(tweety).$$

Therefore, the inconsistency reasoner returns a negative answer to the query. Namely, we get the intended answer:

$$\Sigma_1 \mid\!\approx \neg fly(tweety).$$

The same result would also be valid for the query '$\neg fly(tweety)?$' (Can Tweety not fly?). For the queries on other birds, i.e., not about penguins, the reasoning process would always give the intended results, because the statement $penguin \sqsubseteq \neg fly$ would never be involved. Therefore, it would never lead to an inconsistency.

Now, consider the brain ontology example. Similarly, we add a fact $brain(a)$ into the A-box to state that the concept 'brain' is not empty. For the query $nervousSystem(a)?$ (Is the object 'a' an element of a nervous system?). Based on the direction relevance to the set $atom(nervousSystem(a))$, the selection function would select the following formula set at the beginning:

$$\Sigma_3 = \{centralNervousSystem \sqsubseteq NervousSystem, bodyPart \sqsubseteq \neg nervousSystem\}.$$

However, it is undetermined. The selection function would extend it into 1-relevance. This would lead to all formulas in the example being selected. It is inconsistent. Thus, it is over-determined. To solve this problem, the selection function would select only a part of it. The set Σ_4 is based on the extension on the atom $centralNervousSystem$ and another one Σ_5 is based on the atom $bodyPart$ as follows respectively:

$$\Sigma_4 = \{brain \sqsubseteq centralNervousSystem, brain(a)\} \cup \Sigma_3,$$

$$\Sigma_5 = \{brain \sqsubseteq bodyPart, brain(a)\} \cup \Sigma_3.$$

The answer to the former is still undetermined, however, the latter can give an answer. Therefore, the reasoning process can stop and return the negative answer $\neg nervousSystem(a)$.

The notions of direction relevance and k-relevance are a syntactic approach. Two formula sets which are logically equivalent may lead to different relevance results. For example, the following two sets are logically equivalent: $\Sigma_1 = \{p \rightarrow q, q \rightarrow r\}$ and $\Sigma_2 = \{p \rightarrow q, q \rightarrow r, p \rightarrow r\}$. For the set Σ_1, p and r are 1-relevant, but for the set Σ_2, p and r are directly relevant. As we have shown above, the syntactic relevance approach can give intuitive results for most cases, because we can consider the formulas that have been stated in the ontologies to be explicit beliefs/knowledge, and the formulas that can be derived from the existing formula sets to be implicit beliefs/knowledge. We can count the relevance on the explicit knowledge, instead of the implicit knowledge. More cases are needed to evaluate syntactic relevance approaches for reasoning with inconsistent ontologies.

In the next section, we briefly discuss semantic relevance approaches, which measure the relevance on a semantic level instead of a syntactic level.

4.2 Semantic Relevance

The relevance of two concepts has been studied in computational linguistics for many years [7, 8, 13, 16, 20]. Several notions have been proposed to measure semantic relevance in computational linguistics.

- semantic relatedness: relatedness between two lexically expressed concepts;
- semantic similarity: similarity is measured by virtue of their likeness;
- semantic distance: distance is measured by virtue of their likeness/dislikeness.

However, as pointed out in [7], those three different terms used by different authors or sometimes interchangeably by same authors. The term *semantic distance* may cause even more confusion, as it can be used when talking about either similarity or relatedness.

In [8], Budanitsky and Hirst compare the five different proposed measures of semantic distance or similarity in WordNet [10] by examining their performance in a real-word spelling correction system. Here are some proposed measures:[1]

- Hirst-St-Onge [13]: The semantic relateness of two concepts is measured by their WordNet synsets' connection of the path length $path$ and the number of change of direction d in the path:

  $$rel_{HS}(C_1, C_2) = C - path(C_1, C_2) - k \times d(C_1, C_2),$$

  where C and k are constants.

- Resnik [20]: The similarity is measured by a corpus, namely, by "extent to which they share information". The similarity between two concepts in WordNet to be the information content of their lowest super-ordinate (most specific common subsumer) $lso(C_1, C_2)$:

  $$sim_R(C_1, C_2) = -log_p(lso(C_1, C_2)).$$

- Jiang-Conrath [16]: The semantic distance is measured by extending Resnik's approach as follows:

  $$dist_{JC}(C_1, C_2) = 2log(p(lso(C_1, C_2))) - log(p(C_1)) + log(p(C_2))).$$

According to [8] Jiang and Conrath's measure gives the best performance result. The computational linguistic approaches based on WordNet offer us alternatives for relevance measures. However, they rely on more additional background information on the relevant concepts. In this document, we do not discuss the details of how the semantic relevance approach can be used for reasoning with inconsistent ontologies.

[1] http://www.cogsci.princeton.edu/∼wn/

Chapter 5

Design of Reasoning with Inconsistent Ontologies

5.1 General Consideration

Based on the general framework which is proposed in Chapter 3, we are going to implement a DL reasoner which can support reasoning with inconsistent ontologies. The reasoner is called RIO (a Reasoner for reasoning with Inconsistent Ontologies). As it has been discussed above, the general strategy of reasoning with inconsistent ontologies will be partially achieved by calling a standard DL reasoner. Therefore, a RIO reasoner would rely on a standard DL reasoner, which can either be integrated, or called as an external component. There exist several well-known DL reasoners, like Racer [11], which are stable and popular. RIO will be based on those external DL reasoners. Moreover, in order to make it independent from any particular DL reasoner, RIO calls those DL reasoners via their DIG interface.

The DIG description logic interface is designed as a simple API for a general description logic system [3]. The DIG interface uses a similar idea like SOAP (Simple Object Access Protocol), which has builtin messaging protocols using XML on top of HTTP. DIG clients communicate with a DIG server through the use of HTTP Post requests, either a ‘tell’ request by which the clients can assert ontology statements to the knowledge base in the server, i.e., the reasoner, or an ‘ask’ request by which the clients can make queries on the knowledge base. The DIG server answers requests by returning an XML encoded message.

A RIO should also serve as a DL reasoner via its own DIG interface. It is designed to be a simple API for a general reasoner with inconsistent ontologies. It supports DIG requests from other ontology applications or other ontology and metadata management systems. Thus, the implementation of RIO will be independent from those particular applications or systems.

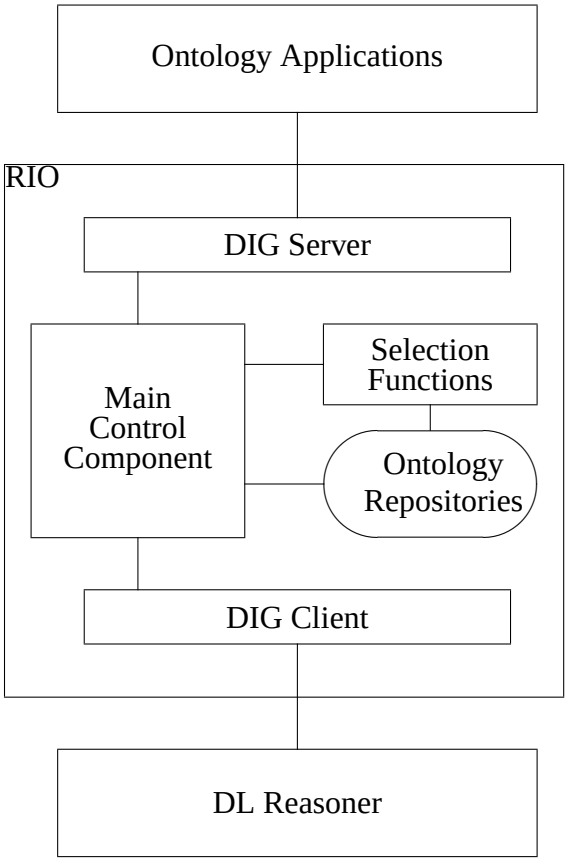

Figure 5.1: Architecture of RIO.

5.2 Architecture

The architecture of a RIO is shown in Figure 5.1. A RIO has the following components:

- **DIG Server**: RIO's DIG server deals with any request from other ontology applications. It supports an extended DIG interface. Namely, it supports not only the standard DIG requests, like 'tell' and 'ask', but also additional reasoning processing, like the identification or change of the selected selection functions.
- **Main Control Component**: The main control component realizes the main processing, like query analysis, query pre-processing, and the extension strategy, by calling the selection function and interacting with the ontology repositories.
- **Selection-Functions**: The selection function component defines the selection functions that can be used in the reasoning process.
- **DIG Client**: RIO's DIG client calls external DL reasoners which support the DIG interface to obtain the standard DL reasoning capabilities.
- **Ontology Repositories**: As the name implies, the ontology repositories are used to store the ontology statements which are provided by external ontology applications.

The main idea for this kind of architecture is to rely on the DIG description logic interface, by which the applications will be independent from the implementations. Of course, scalability and performance are also important issues that we will consider. However, as a first year prototype for the SEKT project, we think this design is more suitable.

5.3 Implementation

We are going to implement the prototype of RIO by using SWI-Prolog.[1] SWI-Prolog is a free software Prolog compiler. Being free, small and mostly standard compliant, SWI-Prolog has become very popular for education and research. We select a logic programming language like Prolog as the tool for the implementation of the prototype, because the logic programming language is convenient and powerful for symbolic processing and list processing.

We are now working on the development of DIG description logic interface libraries for Prolog, which can be used for the implementation of RIO, in particular, for the RIO's DIG client component and server component. A technical report on the DIG description logic interface support for Prolog is available in [15].

[1] http://www.swi-prolog.org

Chapter 6

Discussion and Conclusions

In this report, we have overviewed the state-of-the-art of reasoning with inconsistencies in logics and AI. In particular, we have examined the technologies in paraconsistent logics, approximate reasoning, and belief revision. This part of the document can be considered as a technology match overview on reasoning with inconsistencies.

We have proposed a general framework for reasoning with inconsistent ontologies. We have introduced several formal definitions, such as soundness, meaningfulness, local completeness, and maximal completeness for inconsistency reasoning. We have proposed the pre-processing algorithm and the strategy of inconsistency reasoning processing based on a linear extension strategy, and the strategy of linear extension with S-3-approximate reasoning, and have shown that a linear extension strategy is useful to create meaningful and sound answers to queries, although they may be undetermined and not always maximal.

We have discussed how the selection function can be developed by means of a syntactic relevance measure, and have shown several examples of how the syntactic relevance approach can be used to obtain intuitive reasoning results. We have also discussed briefly how the semantic relevance approaches, which are used in computational linguistics, can also be developed for the task.

In addition, we have presented a design of the RIO system for reasoning with inconsistent ontologies. In particular, we have proposed an architecture of a RIO system, which offers DIG interfaces for both the client side and server side. Thus, our RIO system can support any reasoning request from any application or system via its DIG interface.

In sequel reports, we are going to investigate the formal properties of the selection function, and develop an algorithm how the selection function can be integrated into the inconsistency reasoning processing algorithm. Furthermore, we will implement a RIO system based on the general framework that has been proposed in Chapter 3.

Bibliography

[1] Franz Baader and Ulrike Sattler, An overview of tableau algorithm for description logics, *Studia Logica*, 69: 5-40, Kluwer Academic Publishers, 2000.

[2] Franz Baader, Diego Calvanese, Deborah McGuinness, Daniele Nardi, Peter Patel-Schneider, (eds.), *The Description Logic Handbook Theory, Implementation and Applications*, Cambridge University Press, Cambridge, UK, 2003.

[3] Sean Bechhofer, Ralf Möller, and Peter Crowther. The DIG Description Logic Interface. In DL2003 International Workshop on Description Logics, Rome, September 2003.

[4] N. Belnap, A useful four-valued logic, in: *Modern Uses of Multiple-Valued Logic*, Reidel, Dordrecht, 1977, pp.8-37.

[5] Salem Benferhat, and Laurent Garcia, Handling locally stratified inconsistent knowledge bases, *Studia Logica*, 70: 77-104, Kluwer Academic Publishers, 2002.

[6] Jean-Yves Beziau, What is paraconsistent logic, in: Batens D., Mortensen C., Priest G. and Van Bendegem J.P. (eds). Frontiers of paraconsistent logic. Research Studies Press: Baldock, 2000, 95-111.

[7] Alexander Budanitsky, *Lexical Semantic Relatedness and its Application in Natural Language Processing*, technical report CSRG-390, Department of Computer Science, University of Toronto, August 1999.

[8] Alexander Budanitsky and Graeme Hirst, Semantic distance in WordNet: An experimental, application-oriented evaluation of five measures. In Workshop on WordNet and Other Lexical Resources, Second meeting of the North American Chapter of the Association for Computational Linguistics. Pittsburgh, PA. 2001.

[9] Samir Chopra, Rohit Parikh, and Renata Wassermann, Approximate belief revision-prelimininary report, *Journal of IGPL*, Oxford University Press, 2000.

[10] Christiane Fellbaum, *WordNet, An Electronic Lexical Database*, The MIT Press, 1998.

[11] Volker Haarslev, and Ralf Möller, Description of the RACER System and its Applications, Proceedings of the International Workshop on Description Logics (DL-2001), Stanford, USA, 1.-3. August 2001, pp. 132-141.

[12] A. Hameed, A. Preece and D. Sleeman, Ontology Reconciliation, in: S. Staab and R. Studer (eds), Handbook on Ontologies in Information Systems, Springer Verlag, 231-250, 2003.

[13] Graeme Hirst and David St-Onge, Lexical chains as representations of context for the detection and correction of malapropisms, in Fellbaum 1998, 305-332.

[14] Ian Horrocks, and Peter F. Patel-Schneider, Reducing OWL entailment to description logics satisfiability, Description Logics 2003.

[15] Zhisheng Huang and Cees Visser, Extended DIG Description Logic Interface Support for Prolog, SEKT Report D3.4.1.2, 2004.

[16] Jay Jiang and David Conrath, Semantic similarity based on corpus statistics and lexical taxonomy, in: *Proceedings of International Conference on Research in Computational Linguistics*, Taiwan, 1997.

[17] Jerome Lang, and Pierre Marquis, Removing inconsistencies in assumption-based theories through knowledge-gathering actions, *Studia Logica*, 67: 179-214, 2001.

[18] H. J. Levesque, A knowledge-level account of abduction, *Proceedings of IJCAI'89*, 1061-1067, 1989.

[19] Pierre Marquis and Nadege Porquet, Resource-bounded paraconsistent inference, *Annals of Mathematics and Artificial Intelligence* 39: 349-384, 2003.

[20] Philip Resnik, Using information content to evaluate semantic similarity, in *Proceedings of the 14th IJCAI*, 448-453, 1995.

[21] Russell, S., and Norvig, P., *Artificial Intelligence*, Prentice Hall, 1995.

[22] Marco Schaerf and Marco Cadoli, Tractable reasoning via approximation. *Artificial Intelligence*, 74:249-310, 1995.

[23] Stefan Schlobach, and Ronald Cornet, Non-standard reasoning services for the debugging of description logic terminologies, *Proceedings of IJCAI 2003*, 2003.

[24] A. ten Teije and F. van Harmelen. Computing approximate diagnoses by using approximate entailment. in G. Aiello and J. Doyle, editors, *Proceedings of the 5th International Conference on Principles of Knowledge Representation and Reasoning (KR-96)*, Boston, Massachusetts. Morgan Kaufman, 1996.