Reasoning with Inconsistent Ontologies *

Zhisheng Huang, Frank van Harmelen, and Annette ten Teije
Department of Computer Science, Vrije Universiteit Amsterdam, The Netherlands
{huang, Frank.van.Harmelen, annette}@cs.vu.nl

Abstract

In this paper we present a framework of reasoning with inconsistent ontologies, in which pre-defined selection functions are used to deal with concept relevance. We examine how the notion of "concept relevance" can be used for reasoning with inconsistent ontologies. We have implemented a prototype called PION (Processing Inconsistent ONtologies), which is based on a syntactic relevance-based selection function. In this paper, we also report the experiments with PION.

1 Introduction

There are two main ways to deal with inconsistency in ontologies. One is to diagnose and repair it when we encounter inconsistency[Schlobach and Cornet, 2003]. Another approach is to simply avoid the inconsistency and to apply a non-standard reasoning method to obtain meaningful answers. In this paper, we will focus on the latter, which is more suitable for the setting in the web area. For example, in a typical Semantic Web setting, one would be importing ontologies from other sources, making it impossible to repair them, and the scale of the combined ontologies may be too large to make repair effective.

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. The general task of an inconsistency reasoner is: given an inconsistent ontology, return *meaningful* answers to queries.

Reasoning with inconsistency is a well-known topic in logics and AI. Many approaches have been proposed to deal with inconsistency [Benferhat and Garcia, 2002; Beziau, 2000; Lang and Marquis, 2001; Marquis and Porquet, 2003]. The development of paraconsistent logics was initiated to challenge the 'explosive' problem of the standard logics. Paraconsistent logics [Beziau, 2000] 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 inference [Levesque, 1989] allows the interpretation of a language in which a truth assignment may map both a letter and its negation to true. Extending the idea of Levesque's limited inference, [Schaerf and Cadoli, 1995] proposes S-3-entailment and S-1-entailment for approximate reasoning with tractable results. Based on Schaerf and Cadoli's S-3-entailment, [Marquis and Porquet, 2003] presents a framework for reasoning with inconsistency by introducing a family of resource-bounded paraconsistent inference relations. Several policies, e.g., the linear order policy, are proposed. [Chopra *et al.*, 2000] incorporates the local change of belief revision and relevance sensitivity by means of Schaerf and Cadoli's approximate reasoning method, and shows how relevance can be introduced for approximate reasoning in belief revision. Both approaches, Marquis', 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*: given a selection function, which can be defined on the syntactic or semantic relevance, we select some consistent sub-theory from an inconsistent ontology. Then we apply standard reasoning on the selected sub-theory to find meaningful answers. If a satisfying answer cannot be found, the relevance degree of the selection function is made less restrictive thereby extending the consistent subtheory for further reasoning.

The main contributions of this paper are:(1) a set of *formal definitions* to capture reasoning with inconsistent ontologies; (2) a *general framework* for reasoning with inconsistent ontologies based on selection functions, and (3) some *preliminary experiments* with an implementation of this framework using a rather simple selection function.

This paper is organised as follows: Section 2 overviews inconsistency in the Semantic Web by examining several typical examples and scenarios. Section 3 presents the framework of reasoning with inconsistent ontologies. Section 4 discusses the selection functions Section 5 presents the algorithms of PION. Section 6 examines how the syntactic-relevance-based selection function can be developed. Section 7 describes a prototype of PION and its evaluation. Section 8 discusses further work and concludes the paper.

*The work reported in this paper was partially supported by the EU-funded SEKT project (IST-506826).

2 Inconsistency in the Semantic Web

Here are several typical scenarios which may cause inconsistencies in the Semantic Web.

- **Inconsistency Caused by mis-presentation of defaults**. A notorious example is the bird ontology in which penguins are specified as the birds which cannot fly, however, it contradicts the default statement 'birds are flying animals'. Another typical example is the MadCows ontology[1] in which MadCow is specified as a Cow which eats brains of sheep, whereas a Cow is considered as a vegetarian by default.
- **Inconsistency Caused by Polysemy** Polysemy refers to the concept of words with multiple meanings. An example of an inconsistent ontology which is caused by polysemy is the MarriedWoman Example[Huang *et al.*, 2004], in which the concept 'married woman' is used to refer both to a woman who has a husband and to a woman who *had* a husband but may no longer have one.
- **Inconsistency through Migration from Another Formalism** When an ontology specification is migrated from other data sources, inconsistencies may occur. As has been found in [Schlobach and Cornet, 2003], the DICE terminology (a DL terminology in the Intensive Care domain) suffered from a high number of unsatisfiable concepts due to its migration from a frame-based system. In order to make the semantics as explicit as possible, a very restrictive translation has been chosen to highlight as many ambiguities as possible. [Schlobach and Cornet, 2003] shows the inconsistent ontology specification in the Brain Example, in which a brain is considered to be both a body part and a central nervous system, whereas body parts and nervous systems are considered to be disjoint.
- **Inconsistency Caused by Multiple Sources**. When a large ontology specification is generated from multiple sources, in particular when these sources are created by several authors, inconsistencies easily occur. There are three possibilities for ontology reconciliation: merging, aligning, or integrating. 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.

3 Formal Definitions

We do not restrict ontology specifications to a particular language (although OWL and its underlying 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. Thus, we can consider an ontology specification as a formula set. 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.

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 consequence 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. For reasoning with inconsistent ontologies, it is more suitable to use Belnap's four valued logic [Belnap, 1977] to distinguish the following four epistemic states of the answers:

Definition 1
(a) Over-determined: $\Sigma \mid\!\!\approx \phi$ *and* $\Sigma \mid\!\!\approx \neg\phi$.
(b) Accepted: $\Sigma \mid\!\!\approx \phi$ *and* $\Sigma \not\mid\!\!\approx \neg\phi$.
(c) Rejected: $\Sigma \not\mid\!\!\approx \phi$ *and* $\Sigma \mid\!\!\approx \neg\phi$.
(d) Undetermined: $\Sigma \not\mid\!\!\approx \phi$ *and* $\Sigma \not\mid\!\!\approx \neg\phi$.

For an inconsistency reasoner it is expected that is able to return meaningful answers to queries, given an inconsistent ontology. 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 classical entailment. However, often only a small part of Σ has been incorrectly constructed or modelled, while the remainder of Σ is correct. Therefore, we propose the following definition of soundness.

Definition 2 (Soundness) *An inconsistency reasoner* $\mid\!\!\approx$ *is sound if the formulas that follow from an inconsistent theory* Σ *follow from a consistent subtheory of* Σ *using classical reasoning, namely, the following condition holds:*

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

In other words, the $\mid\!\!\approx$-consequences must be justifiable on the basis of a consistent subset of the theory. 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. In other words, the selection of specific subsets on which $\mid\!\!\approx$ will be based is an integral part of the definition of an inconsistency reasoner, and will be discussed in more detail in the next section.

Definition 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:*

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

An inconsistency reasoner is said to be meaningful iff all of the answers are meaningful.

Because of inconsistencies, classical completeness is impossible. We suggest the notion of local completeness:

[1] http://www.daml.org/ontologies/399

Definition 4 (Local Completeness) *An inconsistency reasoner is locally complete with respect to a consistent subtheory Σ' iff for any formula ϕ, the following condition holds:*

$$\Sigma' \models \phi \Rightarrow \Sigma \mid\!\approx \phi.$$

Since the condition can be represented as:

$$\Sigma \not\mid\!\approx \phi \Rightarrow \Sigma' \not\models \phi,$$

local completeness can be considered as a complement to the soundness property. An answer to a query ϕ_i on Σ is said to be locally complete with respect to a consistent set Σ' iff the following condition holds:

$$\Sigma' \models \phi_i \Rightarrow \Sigma \mid\!\approx \phi_i.$$

Definition 5 (Maximality) *An inconsistency reasoner is maximal iff there is a maximal consistent subtheory such that its consequence set is the same as the consequence set of the inconsistency reasoner:*

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

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

Definition 6 (Local Soundness) *An answer to a query $\Sigma \mid\!\approx \phi$ is said to be locally sound with respect to a consistent set $\Sigma' \subseteq \Sigma$, iff the following condition holds:*

$$\Sigma \mid\!\approx \phi \Rightarrow \Sigma' \models \phi.$$

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

Proposition 3.1
(a) Local Soundness implies Soundness and Meaningfulness.
(b) Maximal completeness implies Local Completeness.

Given a query, there might exist more than one maximal consistent subset and more than one locally-complete consistent subset. Such different maximally consistent subsets may give different $\mid\!\approx$-consequences for a given query ϕ. Therefore, arbitrary (maximal) consistent subsets may not be very useful for the evaluation of a query by some inconsistency reasoner. The consistent subsets should be chosen on structural or semantic grounds indicating the relevance of the chosen subset with respect to some query.

4 Selection Functions

An inconsistency reasoner uses a selection function to determine which consistent subsets of an inconsistent ontology should be considered in its reasoning process. The general framework is independent of the particular choice of selection function. The selection function can either be based on a syntactic approach, like Chopra, Parikh, and Wassermann's syntactic relevance [Chopra *et al.*, 2000], or based on semantic relevance like for example in computational linguistics as in Wordnet [Budanitsky and Hirst, 2001].

Given an ontology (i.e., a formula set) Σ and a query ϕ, a selection function s is one which returns a subset of Σ at the step $k > 0$. Let $\mathbf{L}$ be the ontology language, which is denoted as a formula set. We have the general definition about selection functions as follows:

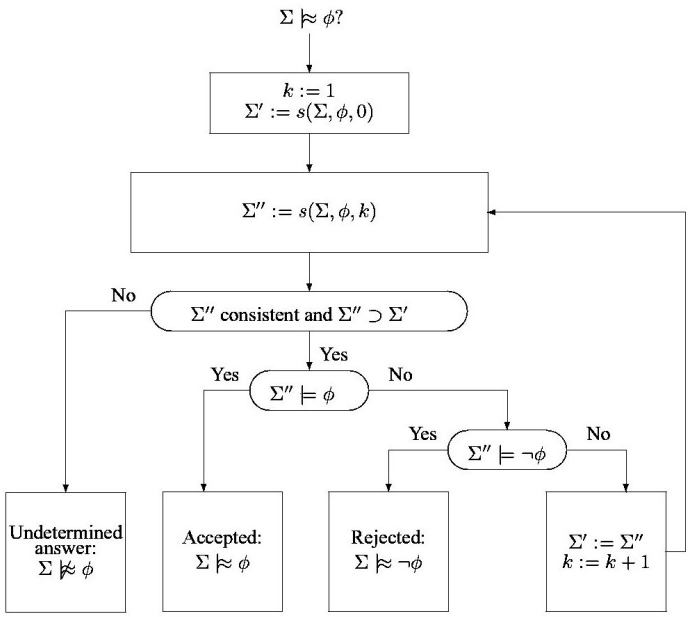

Figure 1: Linear Extension Strategy.

Definition 7 (Selection Functions) *A selection function s is a mapping $s : \mathcal{P}(\mathbf{L}) \times \mathbf{L} \times N \rightarrow \mathcal{P}(\mathbf{L})$ such that $s(\Sigma, \phi, k) \subseteq \Sigma$.*

Definition 8 *A selection function s is called* monotonic *if the subsets it selects monotonically increase or decrease, i.e., $s(\Sigma, \phi, k) \subseteq s(\Sigma, \phi, k+1)$, or vice verse.*

For monotonically increasing selection functions, the initial set is either an emptyset, i.e., $s(\Sigma, \phi, 0) = \emptyset$, or a fixed set Σ_0 when the locality is required. For monotonically decreasing selection functions, usually the initial set $s(\Sigma, \phi, 0) = \Sigma$. The decreasing selection functions will reduce some formulas from the inconsistent set step by step until they find a maximally consistent set.

Monotonically increasing selection functions have the advantage that they do 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 for some value of k, then other subsets (for higher values of k) don't have to be considered any more, because they will not change the answer of the inconsistency reasoner.

5 Strategies

An inconsistency reasoner that uses a monotonically increasing/decreasing selection function will be called an inconsistency reasoner that uses a *linear extension strategy* and a *linear reduction strategy* respectively.

A linear extension strategy is carried out as shown in Figure 1. Given a query $\Sigma \mid\!\approx \phi$, the initial consistent subset Σ' is set. Then the selection function is called to return a consistent subset Σ'', which extends Σ', i.e., $\Sigma' \subset \Sigma'' \subseteq \Sigma$ for the linear extension strategy. If the selection function cannot find a consistent superset of Σ', the inconsistency reasoner returns the answer 'undetermined' (i.e., unknown) to the query. 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 'accepted' answer $\Sigma \mid\!\approx \phi$. 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 'rejected' answer $\Sigma \not\mid\!\approx \neg\phi$, otherwise the current result is undetermined (def.1), and the whole process is repeated by calling 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 'undetermined' 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. Notice, that this depends on the choice of the monotonic selection function.

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 5.1 (Linear Extension) *An inconsistency reasoner using a linear extension strategy satisfies the following properties:*
(a) never over-determined,
(b) may be undetermined,
(c) always sound,
(d) always meaningful,
(e) always locally complete,
(f) may not be maximal,
(g) always locally sound.

Therefore, an inconsistency reasoner using a linear extension strategy is useful to create meaningful and sound answers to queries. It is always locally sound and locally complete with respect to a consistent set Σ_0 if the selection function always starts with the consistent set Σ_0 (i.e., $s(\Sigma, \phi, 0) = \Sigma_0$). 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 exists other strategies which can improve the linear extension approach, for example, by backtracking and heuristics evaluation. We are going to discuss a backtracking strategy in Section 6. The second reason why we call the strategy linear is that the computational complexity of the strategy is linear with respect to the complexity of the ontology reasoning. Let n be the cardinality $|\Sigma|$ of an ontology Σ and let the complexity of $\models$ be E.

Proposition 5.2 (Complexity of Linear Extension) *The complexity of $\mid\!\approx$ in the linear extension strategy is $n \cdot E$.*

In other words, the linear extension strategy does not significantly increase the complexity of the ontology reasoning, because typically E is already *PSPACE-complete* for standard concept languages [Donini, 2003].

6 Syntactic Relevance-based Selection Functions

[Chopra *et al.*, 2000] proposes 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 approximate reasoning. 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: (a) p, p_1 are directly relevant; (b) p_i, p_{i+1} are directly relevant, $i = 1, \ldots, k-1$; and (c) p_k, q are directly relevant (i.e., directly relevant is k-relevant for $k = 0$).

The notions of relevance above are based on propositional logics. However, ontology languages are usually written in some fragment of first order logic. We extend the ideas of relevance to those first-order logic-based languages by restricting relevance to the co-occurrence of only the predicate letters or constant symbols. The following definition specialises the general definition of relevance for the case where ϕ is a formula in an ontology.

Given a formula ϕ, we use $I(\phi), C(\phi), R(\phi)$ to denote the sets of individual names, concept names, and relation names that appear in the formula ϕ respectively.

Definition 9 (Direct relevance) *Two formula ϕ, ψ are directly relevant iff there is a common name which appears both in formula ϕ and formula ψ, i.e., $I(\phi) \cap I(\psi) \neq \varnothing \vee C(\phi) \cap C(\psi) \neq \varnothing \vee R(\phi) \cap R(\psi) \neq \varnothing$.*

Definition 10 (Direct relevance to a set) *A formula ϕ is relevant to a formula set Σ iff there exists a formula $\psi \in \Sigma$ such that ϕ and ψ are directly relevant.*

We can similarly specialise the notion of k-relevance.

Definition 11 (k-relevance) *Two formulas ϕ, ϕ' are k-relevant with respect to a formula set Σ iff there exist formulas $\psi_0, \ldots \psi_k \in \Sigma$ such that ϕ and ψ_0, ψ_0 and ψ_1, ..., and ψ_k and ϕ' are directly relevant.*

Definition 12 (k-relevance to a set) *A formula ϕ is k-relevant to a formula set Σ iff there exists a formula $\psi \in \Sigma$ such that ϕ and ψ are k-relevant with respect to Σ.*

In inconsistency reasoning we can use syntactic relevance to define a selection function s to extend the query '$\Sigma \mid\!\approx \phi$?' as follows: We start with the query formula ϕ as a starting point for the selection based on syntactic relevance. Namely, we define:

$$s(\Sigma, \phi, 0) = \varnothing.$$

Then the selection function selects the formulas $\psi \in \Sigma$ which are directly relevant to ϕ as a working set (i.e. $k = 1$) to see whether or not they are sufficient to give an answer to the query. Namely, we define:

$$s(\Sigma, \phi, 1) = \{\psi \in \Sigma \mid \phi \text{ and } \psi \text{ are directly relevant}\}.$$

If the reasoning process can obtain an answer to the query, it stops. Otherwise the selection function increases the relevance degree by 1, thereby adding more formulas that are relevant to the current working set. Namely, we have:

$$s(\Sigma, \phi, k) = \{\psi \in \Sigma \mid \psi \text{ is directly relevant to } s(\Sigma, \phi, k-1)\},$$

for $k > 1$. This leads to a "fan out" behavior of the selection function: the first selection is the set of all formulae that are directly relevant to the query; then all formulae are selected that are directly relevant to that set, etc. This intuition is formalized in the following:

Proposition 6.1 *The syntactic relevance-based selection function s is monotonically increasing.*

Proposition 6.2 *If $k \geq 1$, then*

$$s(\Sigma, \phi, k) = \{\phi | \phi \text{ is (k-1)-relevant to } \Sigma\}$$

The syntactic relevance-based selection functions defined above usually grows up to an inconsistent set rapidly. That may lead to too many undetermined answers. In order to improve it, we can require that the selection function returns a consistent subset Σ'' at the step k when $s(\Sigma, \phi, k)$ is inconsistent such that $s(\Sigma, \phi, k-1) \subset \Sigma'' \subset s(\Sigma, \phi, k)$. It is actually a kind of backtracking strategies which are used to reduce the number of undetermined answers to improve the linear extension strategy. We call the procedure an *over-determined processing*(ODP) of the selection function. Note that the over-determined processing need no to exhaust the powerset of the set $s(\Sigma, \phi, k) - s(\Sigma, \phi, k-1)$, because of the fact that if a consistent set S cannot prove or disprove a query, then nor can any subset of S. Therefore, one approach of ODP is to return just a maximally consistent subset. Let n be $|\Sigma|$ and k be $n - |S|$, i.e., the cardinality difference between the ontology Σ and its maximal consistent subset S (note that k is usually very small), and let C be the complexity of the consistency checking. The complexity of the over-determined processing is polynomial to the complexity of the consistency checking:

Proposition 6.3 (Complexity of ODP) *The complexity of the over-determined processing is $n^k \cdot C$.*

Note that ODP introduces a degree of non-determinism: selecting different maximal consistent subsets of $s(\Sigma, \phi, k)$ may yield different answers to the query $\Sigma \mid\!\approx \phi$. The simplest example of this is $\Sigma = \{\phi, \neg\phi\}$.

7 Prototype of PION

7.1 Implementation

We are implementing the prototype of PION by using SWI-Prolog.[2] PION implements an inconsistency reasoner based on an linear extension strategy and the syntactic relevance-based selection function as discussed in Sections 5 and 6. The selection function returns the first maximal consistent subset for its over-determined processing. PION is powered by XDIG, an extended DIG Description Logic interface for Prolog [Huang and Visser, 2004]. PION supports the TELL requests in DIG data format and in OWL, and the ASK requests in DIG data format. A prototype of PION is available for download at the website: http://wasp.cs.vu.nl/sekt/pion.

7.2 Experiments and Evaluation

We have tested the prototype of PION by applying it on several example ontologies. These example ontologies are the bird example, the brain example, the MarriedWoman example, and the MadCow Ontolog, which are discussed in Section 2. We compare PION's answers with their intuitive answers which is supposed by a human to see to what extend PION can provide intended answers.

For a query, there might exist the following difference between an answer by PION and its intuitive answer.

- **Intended Answer**: PION's answer is the same as the intuitive answer;
- **Counter-intuitive Answer**: PION's answer is opposite to the intuitive answer. Namely, the intuitive answer is 'accepted' whereas PION's answer is 'rejected', or vice versa.
- **Cautious Answer**: The intuitive answer is 'accepted' or 'rejected', but PION's answer is 'undetermined'.
- **Reckless Answer**: PION's answer is 'accepted' or 'rejected' whereas the intuitive answer is 'undetermined'. We call it a reckless answer, because under this situation PION returns just one of the possible answers without seeking other possibly opposite answers, which may lead to 'undetermined'.

For each concept C in those ontologies, we create an instance 'the_C' on them. We make both a positive instance query and a negative instance query of the instance 'the_C' for some concepts C' in the ontologies, like a query 'is the_C a C'?'. PION test results are shown in Figure 2. Of the four test examples, PION can return at least 85.7% intended answers. Of the 396 queries, PION returns 24 cautious answers or reckless answers, and 2 counter-intuitive answers. However, we would like to point out that the high rate of the intended answers includes many 'undetermined' answers. One interesting (and we believe realistic) property of the Mad Cows Ontology is that many concepts which are intuitively disjoint (such as cows and sheep) are not actually declared as being disjoint (keep in mind that OWL has an open world semantics, and does not make the unique name assumption). As a result, many queries such as "is the_cow a sheep" are indeed undetermined on the basis of the ontology, and PION correctly reports them as undetermined. The average time cost of the tested queries is about 5 seconds even on a low-end PC (with 550 mhz CPU, 256 MB memory under Windows 2000).

The counter-intuitive results occurs in the MadCows Example. PION returns the 'accepted' answer to the query 'is the_mad_cow a vegetarian?'. This counter-intuitive answer results from the weakness of the syntactic relevance-based selection function, because it always prefers a shorter relevance path when a conflict occurs. In the mad cow example, the path 'mad cow - cow - vegetarian' is shorter than the path 'mad cow - eat brain - eat bodypart - sheep are animals - eat animal - NOT vegetarian'. Therefore, the syntactic relevance-based selection function finds a consistent sub-theory by simply ignoring the fact 'sheep are animals'. The problem results from the unbalanced specification between Cow and MadCow, in which Cow is directly specified as a vegetarian whereas there is no direct statement 'a MadCow is not a vegetarian'.

There are several alternative approaches to solve this kind of problems. One is to introduce the locality requirement. Namely, the selection function starts with a certain sub-theory which must always be selected. For example, the statement 'sheep are animals' can be considered to be a knowledge

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

Example	Queries	IA	CA	RA	CIA	IA Rate(%)	ICR Rate(%)
Bird	50	50	0	0	0	100	100
Brain	42	36	4	2	0	85.7	100
MarriedWoman	50	48	0	2	0	96	100
MadCow	254	236	16	0	2	92.9	99

IA = Intended Answers, CA = Cautious Answers, RA = Reckless Answers, CIA = Counter-Intuitive Answers, IA Rate = Intended Answers(%), ICR Rate = IA+CA+RA(%).

Figure 2: PION Test Results

statement which cannot be ignored. Another approach is to add a shortcut path, like the path 'mad cow - eat animal - NOT vegetarian' to achieve the relevance balance between the concepts 'vegetarian' and 'NOT vegetarian', as shown in the second mad cow example of PION testbed. The latter approach can be achieved automatically by accommodation of the semantic relevance from the user queries. The hypothesis is that both concepts appear in a query more frequently, when they are semantically more relevant. Therefore, from a semantic point of view, we can add a relevance shortcut path between strongly relevant concepts.

8 Discussion and Conclusions

In this paper, we have presented a framework for reasoning with inconsistent ontologies. We have introduced the formal definitions of the selection functions, and investigated the strategies of inconsistency reasoning processing based on a linear extension strategy.

One of the novelties of our approach is that the selection functions depend on individual queries. Our approach differs from the traditional one in paraconsistent reasoning, nonmonotonic reasoning, and belief revision, in which a predefined preference ordering for all of the queries is required. This makes our approach more flexible, and less inefficient to obtain intended results. The selection functions can be viewed as ones creating query-specific preference orderings.

We have implemented and presented a prototype of PION. In this paper we have provided the evaluation report of the prototype by applying it to the several inconsistent ontology examples. The tests show that our approach can obtain intuitive results in most cases for reasoning with inconsistent ontologies. Considering the fact that standard reasoners always results in either meaningless answers or incoherence errors for queries on inconsistent ontologies, we can claim that PION can do much better, because it can provide a lot of intuitive, thus meaningful answers. This is a surprising result given the simplicity of our selection function.

In future work, we are going to test PION with more large-scale ontology examples. We are also going to investigate different approaches for selection functions (e.g., semantic-relevance based) and different extension strategies as alternatives to the linear extension strategy in combination with different selection functions, and test their performance.

References

[Belnap, 1977] N. Belnap. A useful four-valued logic. In *Modern Uses of Multiple-Valued Logic*, pages 8–37, Dordrecht, 1977. Reidel.

[Benferhat and Garcia, 2002] S. Benferhat and L. Garcia. Handling locally stratified inconsistent knowledge bases. *Studia Logica*, pages 77–104, 2002.

[Beziau, 2000] J.-Y. Beziau. What is paraconsistent logic. In *Frontiers of paraconsistent logic*, pages 95–111, Baldock, 2000. Research Studies Press.

[Budanitsky and Hirst, 2001] A. Budanitsky and G. Hirst. Semantic distance in wordnet: An experimental, application-oriented evaluation of five measures. In *Workshop on WordNet and Other Lexical Resources*, Pittsburgh, PA., 2001.

[Chopra *et al.*, 2000] S. Chopra, R. Parikh, and R. Wassermann. Approximate belief revision- prelimininary report. *Journal of IGPL*, 2000.

[Donini, 2003] F. Donini. Complexity of reasoning. In *Description Logic Handbook*, pages 96–136, 2003.

[Huang and Visser, 2004] Z. Huang and C. Visser. *Extended DIG Description Logic Interface Support for Prolog*. SEKT Deliverable D3.4.1.2, 2004.

[Huang *et al.*, 2004] Z. Huang, F. van Harmelen, A. ten Teije, P. Groot, and C. Visser. *Reasoning with Inconsistent Ontologies: framework and prototype*. SEKT Deliverable D3.4.1, 2004.

[Lang and Marquis, 2001] J. Lang and P. Marquis. Removing inconsistencies in assumption-based theories through knowledge-gathering actions. *Studia Logica*, pages 179–214, 2001.

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

[Marquis and Porquet, 2003] P. Marquis and N. Porquet. Resource-bounded paraconsistent inference. *Annals of Mathematics and Artificial Intelligence*, pages 349–384, 2003.

[Schaerf and Cadoli, 1995] M. Schaerf and M. Cadoli. Tractable reasoning via approximation. *Artificial Intelligence*, pages 249–310, 1995.

[Schlobach and Cornet, 2003] S. Schlobach and R. Cornet. Non-standard reasoning services for the debugging of description logic terminologies. In *Proceedings of IJCAI 2003*, 2003.