A Query-specific Reasoning Method for Inconsistent and Uncertain Ontology

Bo Liu, Jianqiang Li, and Yu Zhao

***Abstract*—In knowledge-based systems, the quality and correctness of ontologies play an important role in semantic representation and knowledge sharing. But in reality, ontologies are often inconsistent and uncertain. Because of the difficulty in ensuring the quality of ontologies, there is an increasing need for dealing with the inconsistency and uncertainty in real-world applications of ontological reasoning and management. This paper proposes algorithm RMIU to perform a query-specific reasoning method for inconsistent and uncertain ontologies without changing the original ontologies. The reasoning route and certainty degree of each answer are also provided to the user for facilitating his selection of the most credible answer. Finally the prototype system is constructed and the experiment results validate the usability and effectiveness of our approach.**

***Index Terms*—ontology reasoning, semantic web, inconsistency, uncertainty**

I. Introduction

Ontologies play an important role in the Semantic Web[1]. The quality and correctness of ontologies have great effects on semantic representation and knowledge sharing. However, knowledge and information in the real life are usually uncertain, thereby leading to the uncertainty and inconsistency of the ontology.

The uncertain ontology means that the correctness of the ontology is probabilistic. There are mainly three sources of uncertain ontologies: (1) subjective uncertainty of experts when they build the ontologies, (2) the uncertainty from original ontologies when the ontology is integrated from several sub-ontologies, and (3) the uncertainty from (semi-) automatically ontology learning tools.

The uncertainty of ontology may lead to the inconsistency of the ontology. An inconsistent ontology means that an error or a conflict exists in this ontology, as a result some concepts in the ontology cannot be interpreted correctly. The inconsistency of ontologies may come from mis-presentation, polysemy, migration from another formalism, and integration from multiple sources. The inconsistency will result in wrong answers during ontology reasoning, and also result in false semantic understanding and knowledge representation.

Since there are a great deal of inconsistent and uncertain ontologies in the Semantic Web, and it is always very hard to ensure the quality of ontologies, dealing with inconsistency and uncertainty of ontologies has been recognized as an important problem in the recent decades. Although revising or repairing the imperfect ontology is an effective strategy, the revision cost is relatively high and it is hard to implement. Accordingly, we propose a reasoning method for inconsistent and uncertain ontologies, which can be used to get the query result most probable to be correct without revising the original ontologies, thus more convenient and easy to conduct.

The primary contributions of this paper are summarized as follows:

1. It is achievable to obtain the query results most probable to be correct for uncertain and inconsistent ontology, which exempts the user's high cost in revising the existing ontologies.

2. The reasoning method adopts an incrementally selection function, which is capable of selecting elements related to a specific query. Thus, the elements necessary for the reasoning could be obtained fast and the query results could be returned efficiently.

3. The query results achieved by our method include not only "True" and "False" answers, but also the specific reasoning route (path) and certainty degree of each answer. In this way, the user may obtain more useful information to facilitate his selection of the most credible query result according to the certainty degree of each result.

4. The empirical study of the proposed method is reported. The prototype system constructs an ontology for testing and the evaluation results validate the usability and promise of our approach.

The rest of the paper is organized as follows: Section 2 reviews the related work. The relevant terms and definitions are given in Section 3. Section 4 describes the reasoning method for inconsistent and uncertain ontologies and algorithm RMIU. The system and implementation are presented in Section 5. Finally, the conclusions and future work are depicted in Section 6.

II. Related Work

Normally, there are two main strategies to deal with inconsistent ontologies.

One is to resolve the error whenever an inconsistency is encountered. The other is to "live with" the inconsistency and to apply a non-standard reasoning method to obtain meaningful answers in the presence of inconsistencies.

In this paper, we focus on the latter approach. Considering the revision cost of repairing the inconsistent ontologies, the second strategy is more suitable for the application in the

Manuscript received December 26, 2010; revised January 31, 2011.

Bo Liu is with the NEC Labs China, Beijing, 100084 China (phone: +86-10-62705962; e-mail: liu_bo@nec.cn).

Jianqiang Li is with the NEC Labs China, Beijing, 100084 China (e-mail: li_jianqiang@nec.cn).

Yu Zhao is with the NEC Labs China, Beijing, 100084 China (e-mail: zhao_yu@nec.cn).

ISBN: 978-988-18210-3-4
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

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 [2].

Some researchers have been dedicated to this topic. Z.S. Huang, et al. [2] proposed a linear extension strategy to reason with inconsistent ontologies, but they didn't deal with the situation when the extended subset was inconsistent. Beziau [3] introduced paraconsistent logics which allow theories that are inconsistent but non-trivial. Marquis and Porquet [4] presented a framework for reasoning with inconsistency by introducing a family of paraconsistent inference relations. S.X. Wang [5] depicted a reasoning algorithm for inconsistent OWL ontologies with the aim to support the design of software design patterns.

In the area of reasoning with probabilities in knowledge representation, [6] presented a probabilistic extension of the Ontology Language OWL which relied on Bayesian Networks for reasoning. Fuzzy extensions of OWL have been proposed in [7, 8]. [9] defined a confidence function to handle the uncertain ontologies. [10] applied possibility extension on the description logic to achieve reasoning for uncertain ontology. This method removes elements with low certainty degree by using a decrement function, but they don't deal with the "False" answer of queries. [11] defined a vague ontology for semantic information retrieval, which took uncertainty issue of ontologies into consideration, though it didn't provide a specific solution to solve the issue.

To sum up, the conventional reasoning methods are difficult to obtain correct query results for the ontologies that are both inconsistent and uncertain, and some methods have low processing efficiency. Therefore, there is a need for a new reasoning method capable of considering the uncertainty and inconsistency issues of the ontologies simultaneously and with relatively high processing efficiency.

In this paper, we introduce confidence factor to measure how confident we are of the correctness of the axioms in an ontology. A possibility extension method is utilized to obtain the confidence factor for each element, then an incrementally selection function is conducted to select the elements which are necessary to get the query results step-by-step. Both "true" answer and "false" answer are checked, and the reasoning route and certainty degree of each answer are calculated to get the final query answers. Finally, the ranked query answers are provided to the user to select more conveniently the most credible result.

III. Term and Definition

A. Description Logic

Description Logics (DL) are a family of well-studied set-description languages which have been in use to formalize knowledge for over two decades [12]. They have a well-defined model theoretic semantics, which allows for the automation of a number of reasoning services.

DL is equipped with a formal, logic-based semantics. A distinguished feature is the emphasis on reasoning as a central service: reasoning allows one to infer implicitly represented knowledge from the knowledge that is explicitly contained in the knowledge base [13].

A DL knowledge base $\Sigma = (T, A)$ consists a set T (TBox) and A (ABox). TBox has the form $C \subseteq D$ where C and D are concept descriptions. The ABox contains concept assertions of the form a : C where C is a concept and a is an individual name, and role assertions of the form <a,b> : R, where R is a role, and a and b are individual names.

ALC [14] is a simple yet relatively expressive DL with conjunction ($C \cup D$), disjunction ($C \cap D$), negation ($\neg C$) and universal ($\forall r.C$) and existential quantification ($\exists r.C$). The interpretation function is extended to the different language constructs as follows [12]:

$$(C \cap D)^I = C^I \cap D^I$$
$$(C \cup D)^I = C^I \cup D^I$$
$$(\neg C)^I = U \setminus C^I$$
$$(\exists R.C)^I = \{d \in U \mid \exists e \in U : (d,e) \in R^I \text{ and } e \in C^I\}$$
$$(\forall R.C)^I = \{d \in U \mid \forall e \in U : (d,e) \in R^I \text{ implies } e \in C^I\}$$

A query φ given an ontology $\sum$ can be expressed as an evaluation of the consequence relation $\sum \models \varphi$. There are two answers to that query: 'Yes' ($\sum \models \varphi$) or 'No' ($\sum \not\models \varphi$). A 'yes' answer means that φ is a logical consequence of $\sum$. A 'no' answer means that φ cannot be deduced from $\sum$. The negation of φ is expressed as $\neg\varphi$, so if the negation of φ can be deduced from $\sum$, it is expressed as $\sum \models \neg\varphi$.

In this paper, we focus on ontologies which are represented in the description logic *ALC*. Our approach, however, can be trivially extended to more expressive DLs.

B. Inconsistency of Ontology

First, we will give out some definitions.

Definition 1: A concept C is ***unsatisfiable*** w.r.t. an ontology O iff $C^I = \Phi$ for all models I of O. That means the unsatisfiable concept is interpreted as empty set.

Definition 2: A TBox T is ***inconsistent*** if there is a concept name in T, which is unsatisfiable.

Definition 3: The ***unsatisfiable concept sets of ontology O and concept A*** are subsets of the ontology O in which A is unsatisfiable.

The inconsistency of ontology has transitivity, i.e. one inconsistent axiom may cause many other axioms to become inconsistent as well.

C. Uncertainty of Ontology

An uncertain ontology means that the correctness of the ontology is probabilistic. To capture the certainty degree of ontology elements, we introduce the notion of confidence factor.

Definition 4: The ***confidence factor (CF)*** is a rating annotation which indicates how confident we are of the correctness of the axioms in an ontology. The higher the CF value is, the higher probability that the axiom is correct.

CF is a number between 0 and 1, presented as: $CF : N \rightarrow [0,1]$

in which, N means the set of all possible ontology elements.

The value of CF could be obtained from the experts when they build the ontology, or calculated by some predefined algorithms. There are already some methods to obtain the CF

ISBN: 978-988-18210-3-4
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

value, so the calculation method is not the focus of this paper.

IV. Reasoning with Inconsistent and Uncertain Ontologies

In this section, we will illustrate the reasoning method for the inconsistent and uncertain ontologies. First the key steps, Possibility extension, Selection function and Reasoning route, are introduced, then the algorithm RMIU is given out with a specific example.

A. Possibility Extension

Possibilistic logic [15] or possibility theory offers a convenient tool for handling uncertain or prioritized formulas and coping with inconsistency. When we obtain an ontology using ontology learning techniques, the axioms of the ontology are often attached with confidence degrees and the learned ontology may be inconsistent. In this case, possibilistic logic provides a flexible framework to interpret the confidence values and to reason with the inconsistent ontology under uncertainty.

In this paper, we present a possibilistic extension of description logics based on the definition in [10].

Possibilistic logic is a weighted logic where each classical logic formula is associated with a number in (0, 1]. The confidence factor CF maps each element in the ontology to a number between 0 and 1, presented as: $CF : N \to [0,1]$.

$CF(\varphi)$ represents the confidence factor of formula φ.

A *possibilistic formula* is a pair (φ, c), in which φ is a logic formula and c is the certainty degree of φ. A possibilistic knowledge base is the set of possibilistic formulas, expressed as: $B = \{(\varphi_i, c_i) : i = 1,...,n\}$.

A *possibilistic axiom* is a pair (φ, c), in which φ is an axiom and c is a weight ($c \in [0,1]$). A *possibilistic TBox*/*ABox* is a finite set of possibilistic axioms (φ, c), where φ is a TBox/ABox axoim. A possibilistic DL knowledge base B = (T, A) contains a possibilistic TBox T and a possibilistic ABox A. If T^* is denoted the classical DL axioms associated with T, A^* is denoted the classical DL axioms associated with A, then $T^* = \{\varphi_i : (\varphi_i, c_i) \in T\}$, $A^* = \{\varphi_i : (\varphi_i, c_i) \in A\}$. The classical base B^* of a possibilistic DL knowledge base is $B^* = (T^*, A^*)$. A possibilistic DL knowledge base B is inconsistent iff B^* is inconsistent [10].

B. Selection Function

An incrementally selection function is utilized to determine which subsets of an inconsistent ontology should be considered in its reasoning process. Here we take [2] as a reference, and define a selection function which will be used in algorithm RMIU.

Definition 5: A ***selection function*** s is a mapping s, which selects a subset of ontology $\sum$ relevant to query φ at the step k $\geq$ 0, i.e., $s(\sum, \varphi, k) \subseteq \sum$, $s(\sum, \varphi, k) \subseteq s(\sum, \varphi, k+1)$.

In this function, the initial set is an empty set, i.e., $s(\sum, \varphi, 0) = \Phi$. The incrementally selection function has the advantage that they don't have to return all subsets for consideration at the same time. If a query $\sum \models \varphi$ can be answered after considering some subset of the ontology for some value of k, the processing cost is lower and the efficiency is higher. The user could determine whether need to consider other subsets with higher values of k.

The implementation of selection function is based on syntactic relevance.

Definition 6: Suppose there are two elements ϕ and ψ in a given ontology. If there is a common name (e.g. instance name, concept name, relationship name) existing in both ϕ and ψ, it is named that ϕ and ψ are ***directly relevant*** [2].

Definition 7: Suppose there are two elements ϕ and ϕ' in a given ontology. If there are a set of elements satisfying, $\psi_0, \ldots, \psi_k \in \sum$, and, ϕ and ψ_0 are directly relevant, ψ_0 and ψ_1 are directly relevant, …, ψ_k and ϕ' are directly relevant, then ϕ and ϕ' are referred to as ***K-relevant***, or the relevance degree between ϕ and ϕ' is K [2].

The syntactic relevance is utilized to define the selection function s to extend the query ' $\sum \models \varphi$?' as follows: We start with the query formula φ as a starting point for the selection function, i.e., $s(\sum, \varphi, 0) = \Phi$

Then the selection function selects the formulas ψ 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. That is,

$s(\sum, \varphi, 1) = \{\psi \in \sum \mid \varphi \text{ and } \psi \text{ are directly relevant}\}$

With the increase of k, s selects more elements that are relevant to the current working set. For k>1,

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

We use the relevance-based selection function to fast obtain the subset of the inconsistent ontology that is necessary to get the query result.

C. Reasoning Route

To better record the reasoning process of inconsistent ontologies, we define Reasoning route as follows:

Definition 8: ***Reasoning route*** is a collection R, which records all the ontology elements that are necessary to reason out the query φ, that is, if any element in the collection R is removed, it is impossible to obtain an answer for the query φ. It denotes $R \models \varphi$ and $R' \subset R, R' \not\models \varphi$.

After obtaining the reasoning route of a query answer, we can further calculate the certainty degree of the answer.

Definition 9: The ***Certainty Degree of an Answer (CDA)*** for a query means a score, for indicating the certainty or credibility degree of the answer in the ontology. The higher the CDA is, the more probable the answer is correct. For example, CDA can be represented as a digital from 0 to 1, namely, $CDA : A \to [0,1]$, wherein A denotes the collection of all the answers for the query.

In this paper, the CDA of a query is calculated by multiplying the confidence factors of each element in the reasoning route of this answer.

D. Algorithm RMIU

Our proposed reasoning method for inconsistent and uncertain ontology includes the following steps:

1) input an ontology and a query;

2) perform possibility extension on the ontology to calculate a Confidence Factor (CF) value for each element in

ISBN: 978-988-18210-3-4
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

the ontology;

3) select a set of elements relevant to the query from the ontology by using the selection function;

4) check whether it can reason out any query result from the set of selected elements;

5) if it is determined that the set of selected elements cannot get any query result, increment the relevant degree of the selection function and repeat the steps 3) and 4), or if it can get query results, record the query results and corresponding reasoning paths and certainty degree of the answers, as well as increment the relevant degree of the selection function and repeat the steps 3) and 4);

6) when the set of selected elements already contain all the elements in the ontology or the selection function cannot select any more element relevant to the query, rank the recorded query results in terms of the certainty degree, and output all of the recorded query results and their corresponding reasoning routes and certainty degree.

The algorithm RMIU (Reasoning Method for Inconsistent and Uncertain ontology) is described below.

The input of algorithm RMIU is an ontology B and a query φ, the aim of the algorithm is to determine whether any result of the query φ can be obtained from B: $B \models \varphi ?$. Ontology B is defined as B=(T, A), T={Ψ_i, i=1,2,..,n} and A={c_j, j=1,2,..,m}, T denoting a set of axioms, A denoting a set of assertions.

The output of the algorithm is the query results set A which include two parts: positive answer (PA) and negative answer (NA), expressed as: A={PA, NA}. PA stores the answers when the query is "True" (i.e. $B \models \varphi$), NA stores the answers when the query is "False" (i.e. $B \models \neg\varphi$).

Algorithm RMIU(B, φ): Reasoning Method for Inconsistent and Uncertain ontology

Input: Inconsistent and Uncertain ontology B, query φ

Output: The query result set A of ontology B with query φ

begin

While B is inconsistent and uncertain **do** // Check whether B is inconsistent and uncertain, if yes, start the cycle

{ $B^* \leftarrow PossibilityExtension(B)$ // B* is the possibility extension of B

For (int k=1; C; k++) // C is the predefined threshold of k

{

$\Sigma' = s(B^*, \varphi, k)$ // selection function *s*

If ($\Sigma' \models \varphi$) // if the query answer is true

{

$PRP \leftarrow GetPosReasoningPath(\Sigma', \varphi)$ // get positive reasoning path

$CD \leftarrow GetCertaintyDegree(\Sigma', PRP)$ // get certainty degree of the answer

$PA \leftarrow PA + GetQueryResult(\Sigma', PRP, CD)$ // positive Answer Set PA

$A \leftarrow PA$ // add PA to A

}

If ($\Sigma' \models \neg\varphi$) // if the query answer is false

{

$NRP \leftarrow GetNegReasoningPath(\Sigma', \varphi)$ // get negative reasoning path

$CD \leftarrow GetCertaintyDegree(\Sigma', NRP)$ // get certainty degree of the answer

$NA \leftarrow NA + GetQueryResult(\Sigma', NRP, CD)$ // negative Answer Set NA

$A \leftarrow NA$ // add NA to A

}

If ($\Sigma' = B^*$ *or* $\Sigma = \Sigma'$) // closure condition

Break;

Else $\Sigma \leftarrow \Sigma'$

}

If ((PA is empty) and (NA is empty))

{ $A \leftarrow \Phi$ //A is empty set

Return A //return the empty set A }

Else { $A \leftarrow rank(PA, NA)$ // rank the query results by certainty degree

Return A //return the Query Result Set A }

}

End While

end

When B is inconsistent and uncertain, the function *PossibilityExtension( )* produces a possibility extension ontology B*. The possibility extension function assigns each element in the ontology a confidence factor. That is, $B = (T, A) \rightarrow B^* = (T^*, A^*)$. T* denotes a set of possibilistic axioms, i.e. T*={(Ψ_i,α_i), i=1,2,..,n}. A* denotes a set of possibilistic assertions, A*={(c_j, α_j), j=1,2,..,m}. α represents a confidence factor. Regarding the calculation method of the confidence factor, it can use any well-known techniques in this field, like [10].

Next, the Incrementally Selection Function *s* selects the elements having a relevance degree *k* with the query φ. *k* has an initial value of 1 and plus 1 each time when incrementing the selection.

In each cycle, the reasoner first checks whether the current working set Σ' can get a "True" answer of the query φ. If so, *GetPosReasoningPath()* and *GetCertaintyDegree()* functions are performed to calculate and record the reasoning path of the answer and its certainty degree. The reasoning path includes all the elements in the ontology that are necessary to reason out the answer, the certainty degree is calculated by multiplying the CFs of all the elements along the reasoning path. Then the positive answers with the corresponding reasoning path and certainty degree are recorded in PA by function *GetQueryResult( )*.

To the contrary, if there is no "True" answer, the reasoner checks whether any "False" answer of the query φ can be reasoned out. Then, the function *GetNegReasoningPath( ), GetCertaintyDegree( )* and *GetQueryResult( )* are conducted to calculate and record the reasoning path of the negative answer and its certainty degree.

If the set Σ' is equal to the ontology B* (which means the current set has chosen all the elements in the ontology, and thus no new element can be added in), or if Σ' is equal to the ontology Σ (which means this round of selection function does not select new element to join in, that is, there will be no more element relevant to Σ'), the cycle is terminated; if not, give the set Σ' to Σ and start next cycle.

Finally, if the query φ has an answer ("True" or "False"), *rank(PA,NA)* function ranks all the query results according to the certainty degree of each answer, the algorithm outputs the ranked query answer set A, represented as:

ISBN: 978-988-18210-3-4
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

A={{answer ID: True (or False), path: {…}, certainty degree},…}

If A is empty, it means the ontology B is not enough to make a determination. We name the query answer "Undetermined".

To better explain the principle of algorithm RMIU, a specific example is given below. B is an inconsistent ontology, which includes 11 axioms: ax_1-ax_{11}, in which A-E, G, H, K, Q mean concepts, a and q mean instances.

$$B=\left\{\begin{array}{c} A\subseteq B, A\subset C, A\subset D\cap Q, A(a), B\subseteq G, \\ C\subset B, D\subset E, E\subset B, G\subset H\cap K, H\subset A, Q(q)\end{array}\right\},$$

by passing the Possibility Extension function,

$$B^*=\left\{\begin{array}{l}(A\subseteq B,0.7),(A\subset C,0.8),(A\subset D\cap Q,0.9),\\ (A(a),0.5),(B\subseteq G,0.3),\\ (C\subset B,0.6),(D\subset E,0.8),(E\subset B,0.7),\\ (G\subset H\cap K,0.5),(H\subset A,0.6),(Q(q),0.5)\end{array}\right\}$$

Suppose the input query φ is: "$A\subset B$?".

The initial k is 1 and the selection function is as follows:

$$\Sigma' = s(B^*,\varphi,1)$$

$$=\left\{\begin{array}{l}(A\subseteq B,0.7),(A\subset C,0.8),(A\subset D\cap Q,0.9),(A(a),0.5),\\ (B\subseteq G,0.3),(C\subset B,0.6),(E\subset B,0.7),(H\subset A,0.6)\end{array}\right\}$$

Then check whether $\Sigma'|=\varphi$. Two reasoning paths are found as follows:

Answer set A = {{answer 1: True, path: $\{(A\subseteq B,0.7)\}$, 0.7}, {answer 2: True, path: $\{(A\subset C,0.8),(C\subset B,0.6)\}$, 0.48}}

Check whether $\Sigma'|=\neg\varphi$. The answer is 'No'.

Continue to check whether Σ'=B* or $\Sigma'=\Sigma$. It is found neither of the two conditions can be satisfied, then k = k+1, i.e. k=2, and the set Σ' is given to Σ.

$$\Sigma' = s(B^*,\varphi,2)$$

$$=\left\{\begin{array}{l}(A\subseteq B,0.7),(A\subset C,0.8),(A\subset D\cap Q,0.9),(A(a),0.5),\\ (B\subseteq G,0.3),(C\subset B,0.6),(E\subset B,0.7),(H\subset A,0.6)\\ (D\subset E,0.8),(G\subset H\cap K,0.5),(Q(q),0.5)\end{array}\right\}$$

Then check whether $\Sigma'|=\varphi$. One reasoning path is found, and this path is added to the answer set A:

Answer set A=

$$\left\{\begin{array}{l}\{\text{answer 1: True, path: }\{(A\subseteq B,0.7)\},0.7\},\\ \{\text{answer 2: True, path: }\{(A\subset C,0.8),(C\subset B,0.6)\},0.48\},\\ \{\text{answer 3: True, path: }\{(A\subset D\cap Q,0.9),(D\subset E,0.8),\\ (E\subset B,0.7)\},0.5\}\end{array}\right\}$$

Check whether $\Sigma'|=\neg\varphi$. One reasoning path is found:

{answer 4: False, path:
$\{(B\subseteq G,0.3),(G\subset H\cap K,0.5),(H\subset A,0.6)\}$, 0.09}.

Continue to check whether there is Σ'=B* or $\Sigma'=\Sigma$. The former condition is satisfied. Then, since the answer set "has" answers, the query results need to be ranked. The final output can be obtained as follows:

Query results $A=$

$$\left\{\begin{array}{l}\{\text{answer 1: True, path: }\{(A\subseteq B,0.7)\},0.7\},\\ \{\text{answer 3: True, path: }\{(A\subset D\cap Q,0.9),(D\subset E,0.8),\\ (E\subset B,0.7)\},0.5\},\\ \{\text{answer 2: True, path: }\{(A\subset C,0.8),(C\subset B,0.6)\},0.48\},\\ \{\text{answer 4: False, path: }\{(B\subseteq G,0.3),(G\subset H\cap K,0.5),\\ (H\subset A,0.6)\},0.09\}\end{array}\right\}$$

There are three "True" answers and one "False" answer. The user could select the most proper answer according to the certainty degree and also his personal knowledge.

V. System and Implementation

A. System Architecture

We have implemented a prototype as a reasoner of inconsistent and uncertain ontologies. Figure 1 presents the architecture of the prototype system. Ontology storage stores all the ontologies with OWL format. To construct an inconsistent and uncertain ontology for testing, we use an ontology learning tool (like Text2Onto [16]) to build an ontology from Reuters News. Possibility extension modular performs the possibility extension for each element of the ontology, and stores the possibilistic ontology in possibilistic ontology storage.

The key modulars of the prototype are incremental selection modular, query result checking and recording modular, reasoning path recording modular, certainty degree calculation modular and query result ranking modular. They realize the main function of algorithm RMIU.

User interface inputs the query and outputs the query answers. The user could select the answer most probable to be correct according to the certainty degree of each answer, and also review the reasoning path to know how to get this result. In this way, the user may obtain more useful information to facilitate his selection.

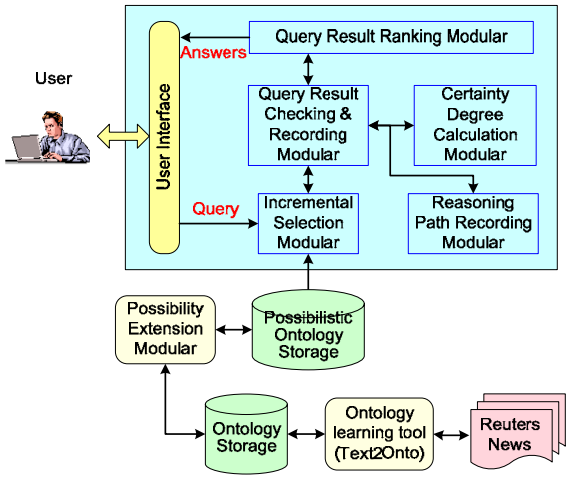

Fig. 1. System architecture of the prototype.

B. Experiment and Evaluation

In order to validate our methods presented in the previous sections, we generated a document set from Reuters News, using 500 documents with the topic code "science and technology". This corpus served as input to the ontology extraction step. For this purpose we decided to use the freely available Text2Onto tool, developed at the AIFB, Karlsruhe, Germany. The learned ontology consisted of the following elements as shown in Table 1.

TABLE I
The Learned Ontology from Reuters News

Learned Element	Number
concepts	450
instances	89
subconcept-of relations	248
disjoint-concepts relations	352
instance-of relations	76

ISBN: 978-988-18210-3-4
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

Then, we applied an evaluation function and algorithm [17] to calculate the CF value of the learned ontology. After possibility extension, the learned ontology is stored into possibilistic ontology storage for further reasoning.

For each instance c of concept C in this ontology, we make a positive instance query like 'is c a C?', and a negative instance query like 'is c a B?', in which B is another concept selected randomly. From the subconcept-of relations, we randomly select 100 relations to construct queries. For each subconcept-of relation like ' $A \subseteq B$ ', we make a positive query ' $A \subseteq B$?', and a negative query like ' $A \subseteq C$?'. After reasoning, the query results are shown in table 2.

To measure the accuracy of the query results, we compare the answers with their intuitive answers which are supposed by a human. For a query, there may exist three answers: True, False, Undetermined. If the answer is the same as the intuitive answer, it is recorded as a correct answer. Of the 378 queries, our reasoner returns 201 'True', 142 'False', and 35 'Undetermined', in which 4 answers are different from the intuitive answers. The total accuracy is 99%, which proves the usability and effectiveness of our method.

TABLE II
THE QUERY RESULTS AND ACCURACY

Answer	True	False	Undetermined
Num. of answers	201	142	35
Num. of error	1	1	2
Accuracy	99.5%	99.3%	94.3%
Num. of answers	201	142	35
Total accuracy	99%		

In the above experiment, we predefine the threshold of k (i.e., C in algorithm RMIU) as 4. That is to say, the selection function will stop incrementally selecting more elements when k=4. Generally speaking, the value of C influences the execution time and the number of query answers. With the increase of threshold C, more elements are selected for reasoning, more query answers may be found, and more processing time is needed. But if the selection function reaches the closure condition ($\Sigma' = B^*$ or $\Sigma = \Sigma'$) before k increases to C, then the threshold C couldn't influence the final results any more. So how to choose the "right" threshold C will depend on the application scenario and the scale of ontology, as it essentially means finding a trade-off between the execution time and the number of query answers.

Figure 2 shows the relation between C and average processing time of the tested queries. When C<5, the time cost is raising with the increase of C. When C≥5, the closure condition is reached, so the processing time remains in 6 seconds.

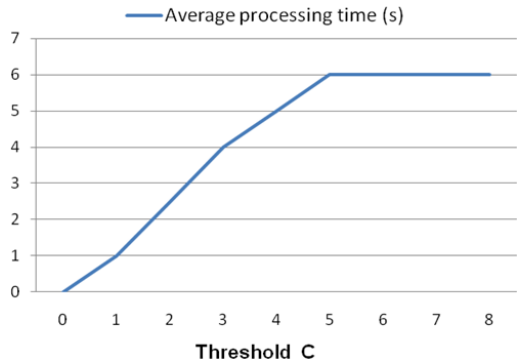

Fig. 2. The relation between threshold C and average processing time.

VI. Conclusions and Future Work

To ensure the quality and correctness of the ontology, this paper proposes a query-specific reasoning method for inconsistent and uncertain ontologies. Without modifying the original ontology, we aim at developing a non-standard reasoner which is able to obtain a meaningful answer with the most possibility to be correct. The reasoning route and certainty degree of each answer are also calculated, thus the user may obtain more useful information to facilitate his selection of the most credible query result. The prototype system validates the feasibility and effectiveness of our method. In future, the efficiency of algorithm RMIU will be improved and more experiments need to be conducted to validate the algorithm in many different application scenarios.

References

[1] T.B. Lee, "Semantic Web Road Map", *W3C Design Issues*, 1998. Available: http://www.w3.org/DesignIssues/Semantic.html
[2] Z.S. Huang, V.F. Harmelen, A.T. Teije, "Reasoning with Inconsistent Ontologies", in *Proceedings of IJCAI'05*, 2005, pp. 254-259.
[3] J.-Y. Beziau, "What is Paraconsistent Logic", *Frontiers of paraconsistent logic*, Research Studies Press, pp. 95–111, 2000.
[4] P. Marquis, N. Porquet, "Resource-bounded Paraconsistent Inference", *Annals of Mathematics and Artificial Intelligence*, pp. 349–384, 2003.
[5] S.X. Wang, "Reasoning with Inconsistent OWL Ontologies for Software Reuse", *World Congress on Software Engineering*, vol. 2, pp.113-116, 2009.
[6] Z. Ding, Y. Peng, "A Probabilistic Extension to Ontology Language OWL", in *Proceedings of the 37th Hawaii International Conference on System Sciences*, Track 4, Vol. 4, IEEE CS Press, 2004.
[7] U. Straccia, "Towards a Fuzzy Description Logic for the Semantic Web" (preliminary report), in *Proceedings of the Second European Semantic Web Conference*, 2005, pp. 67-181.
[8] H.B. Kong, G. Xue, X.L. He, S.W. Yao, "A Proposal to Handle Inconsistent Ontology with Fuzzy OWL", in CSIE, Vol. 1, pp.599-603, 2009, *WRI World Congress on Computer Science and Information Engineering*, 2009.
[9] S.C. Lam, J.Z. Pan, D. Sleeman, W. Vasconcelos, "Ontology Inconsistency Handling: Ranking and Rewriting Axioms", *Technical Report AUCS/TR0603*, 2006.
[10] G.L. Qi, J.Z. Pan, Q. Ji, "Extending Description Logics with Uncertainty Reasoning in Possibilistic Logic", in *Proceedings of the 9th European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty*, CSQARU 2007, LNAI 4724, pp. 828-839, 2007.
[11] S. Calegari, E. Sanchez, "A Fuzzy Ontology-approach to Improve Semantic Information Retrieval", in *Proceedings of the Third ISWC Workshop on Uncertainty Reasoning for the Semantic Web*, URSW'07, 2007, pp. 6-11.
[12] S. Schlobach, Z.S. Huang, "Inconsistent Ontology Diagnosis and Repair", 2007. Available: http://wasp.cs.vu.nl/sekt/dion/sekt363.pdf
[13] F. Baader, D. Calvanese, D.L. McGuinness, et al., "The Description Logic Handbook: Theory, Implementation, and Applications", Cambridge University Press, 2003.
[14] M.S. Schauss, G. Smolka, "Attribute Concept Descriptions with Complements", *Artificial Intelligence*, Vol. 48, pp. 1–26, 1991.
[15] D. Dubois, J. Lang, H. Prade, "Possibilistic Logic", *Handbook of Logic in Artificial Intelligence and Logic Programming*, pp. 439–513. Oxford University Press, Oxford, 1994.
[16] Text2Onto. Available: http://code.google.com/p/text2onto/
[17] P. Haase, J. Völker, "Ontology Learning and Reasoning - Dealing with Uncertainty and Inconsistency", *Lecture Notes in Computer Science*, Vol. 5327, pp. 366-384, 2008.

ISBN: 978-988-18210-3-4
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)