Processing Incomplete Query Specifications in a Context-Dependent Reasoning Framework

Neli P. ZLATAREVA
Department of Computer Science,
Central Connecticut State University
1615 Stanley Street,
New Britain, CT 06050, USA

ABSTRACT

Search is the most prominent web service, which is about to change dramatically with the transition to the Semantic Web. Semantic Web applications are expected to deal with complex conjunctive queries, and not always such queries can be completely and precisely defined. Current Semantic Web reasoners built upon Description Logics have limited processing power in such environments. We discuss some of their limitations, and show how an alternative logical framework utilizing **context-dependent rules** can be extended to handle incomplete or imprecise query specifications.

Keywords: Semantic Web, Ontology Representation, Reasoning under Uncertainty and Incompleteness, Argumentative Reasoning.

1. INTRODUCTION

World Wide Web (WWW) is a huge repository of information which people helped by search engines and web crawlers can access and use in a variety of ways. To make this repository amenable to automated processing, it must be annotated and explicated using ontologies. These are formal specifications of limited web domains with well-defined semantics, which can be traced by automated reasoners to allow web agents to not only access information but also make use of it and even act on human behalf. Consider, for example, the following scenario. A student is looking for a graduate program that allows her to graduate within a year, fits her undergraduate specialization, and has GRE requirement that she satisfies. She also has preferences related to the type and location of the school, financial aid, etc. Using current web services, the student will browse a (limited) number of universities and search through their web sites, which may be considerably different and thus hard to compare. Now, assume that university catalogs on the web are uniformly organized as ontologies of courses, requirements and services, and all those ontologies are linked via semantic bridges to allow entities from different ontologies to be mapped and compared easily. Assume further that there exists a web service intended to find the "best fit" for the student by automatically navigating through the university domain, collecting and evaluating information about all relevant programs. The search should be carried out under the assumption that the student's initial query may not be specific enough to precisely express her intentions in which case the web service should be able to (i) search in the presence of incompleteness or inconsistency, and (ii) generate query-specific questions to guide the student to further detail her query. The final result of the web search should be one or several best programs that the student can choose from helped by a detailed explanation as to why the selected program(s) is (are) a good fit.

Implementation of this scenario requires a flexible representation framework that allows for concepts with different degree of significance or certainty to be represented and processed. For example, courses are often described in terms of "required" and "recommended" pre-requisites, while programs may allow for multiple implementation paths depending on student background. Current ontology languages such as RDF and RDFS are not expressive enough to allow for data to be associated with a degree of certainty, because they target simple typed ontologies. The Web Ontology Language, OWL [1], which builds upon RDF and RDFS, provides more expressive representation, but its inference capabilities are limited to those of Description Logics (DL) [2], upon which it is built. This limits its ability to represent and process imprecise, incomplete, and possibly inconsistent data that our hypothetical scenario allows. And finally, current ontology reasoners supporting OWL ontologies, such as Racer [3] and Fact++ [4], are capable of computing unsatisfiable, subsumped, equivalent and/or disjoint classes and types for individuals, but they do not provide explanation or justification for their reasoning. The importance of such service is being recognized recently by the Semantic Web community as highly desirable, because a broad audience of users is expected to rely on Semantic Web services [5, 6, 7].

In [8], we have outlined one representation framework utilizing **context-dependent rules**, which we advocated was a good candidate for Semantic Web applications because of its expressiveness, adaptability and computational efficiency.

Further in [9] we have explored the ability of context-dependent rules to handle exceptions, defaults and inconsistencies. In this article, we show how processing power of context-dependent rules can be extended to handle incomplete or imprecise conjunctive queries. In Section 2, we discuss why such queries cannot be adequately processed by current DL-based reasoners and point out how some of their limitations can be tackled if reasoning contexts are used to automatically generate query-specific questions to guide the user towards précising her query specification. In Section 3, we describe the augmented syntax of context-dependent rules (see [8] for detailed presentation of context-dependent reasoning framework) to allow for explicit recording of derivation paths. Section 4 outlines a query interpretation technique for processing incomplete query specifications. Section 5 presents an extended example to illustrate the proposed framework.

2. PRELIMINARIES AND MOTIVATION

Description Logics (DL) are decidable fragments of first-order logic intended to provide a well-defined model-theoretic semantics for earlier representation schemes based on frames and semantic networks [2]. They are defined in terms of classes (unary predicates) representing domain concepts, and roles (binary predicates) representing relations between classes. Complex classes and roles can be built from atomic ones by means of conjunction, disjunction, negation, existential restriction and value restriction constructors.

Domain descriptions expressed as DL ontologies are defined by two finite and mutually disjoint sets, called the **Tbox**, and the **Abox**. The Tbox contains concept inclusion axioms of the form $C \subseteq D$, where **C** and **D** are classes. The Abox contains facts about individual objects such as a : C, where **a** is an individual name and **C** is a class, as well as relations between individual objects such as <a,b> : R, where **a** and **b** are individual names and R is a role.

Given ontology Σ and query δ, $\Sigma \models \delta$ iff every model of Σ is also a model of δ. Likewise, $\Sigma \models \neg\delta$ iff every model of Σ is also a model of $\neg\delta$. If δ is not completely and accurately defined, the following two cases are possible:

- Case 1: Both, δ and $\neg\delta$, are derivable.
- Case 2: Neither δ, nor $\neg\delta$ is derivable.

Current ontology reasoners recognize the former case as logical inconsistency and require all inconsistencies to be repaired before further processing. Inconsistencies, however, may be an integral part of ontological knowledge, and repairing them may not be possible or even desirable. Inconsistencies are common in merged or embedded ontologies, where ontology engineer has no authority to modify original ontologies and thus is unable to repair the inconsistency. In such case, reasoning may have to be carried out in the presence of an inconsistency. Work in this direction is reported in [10, 11].

Case 2 can be caused by either incompleteness in the domain representation, in which case the problem should be addressed prior to web service utilization, or by incompleteness in query specification. Addressing the later was somewhat ignored by the Semantic Web community under the assumption that (i) completeness is not a critical requirement in Semantic Web applications [12], and (ii) the user is responsible for precisely defining his query. While the former is dictated by the necessary tradeoff between completeness and efficiency, the later (as our example scenario suggests) may not always be possible and may have to be addressed by the web service by providing the user with some help in identifying relevant details to complete his query. Employing simple "question -- answer" techniques as utilized in conventional knowledge engineering will be highly inefficient because of unforeseen variety of queries. However, "guessing" user intentions may be helped by generating possible extensions of his query and testing those extensions for relevance. Consider, for example, the following statement:

"Networking class requires Data Structures class, but knowledge on Web Technologies and Computer Architecture is expected."

Notice that relations between Data Structures and Networking on one hand, and Web Technologies and Networking / Computer Architecture and Networking on the other hand, are different. The former can be stated as

<Networking, Data-Structures>: Prerequisite (Required).

The latter can be stated as

<Networking,Web-Technologies>:Prerequisite (Recommended)
<Networking, Computer-Architecture> : Prerequisite (Recommended).

While the student cannot enroll in the Networking class without the required prerequisite, he can enroll without one or both of recommended prerequisites. That is, the role, Prerequisite, can have different semantic meaning depending on the associated type, Required or Recommended, where the former is imperative, and the latter is supportive.

Current ontology languages based on DL do not allow for types of roles which makes it impossible to precisely represent the above relations. Ignoring the type, these relations will be either overstated ("Networking class requires Data Structures, Web Technology, and Computer Architecture classes") or understated ("Networking class requires Data Structures class"). That is, if the user asks if she can enroll in a networking class given that she has already completed data structures and computer architecture classes, the answer will be "NO" in the first case, and "YES" in the second case. While the first answer is clearly incorrect, the second answer is not exactly precise with respect to the original statement.

One of the approaches to tackle this issue is to extend DL with non-monotonic features [13, 14]. But, as stated in [13] "Identifying a non-monotonic DL that is of sufficient expressivity and computationally well-behaved is a non-trivial task. In particular, the resulting formalisms often suffer from one or more of the following problems: (i) they have limitations in expressivity such as treating objects that are named by an individual constant different from unnamed ones; (ii) they are

computationally very hard and easily become undecidable; and (iii) they are often conceived as being difficult to understand." These problems are largely caused by the fact that non-monotonic logics work under the Closed World Assumption (CWA), which states that everything not explicitly declared to be true is false. An unwanted consequence of the CWA is that assumptions must be tested for consistency with the current knowledge. As the Semantic Web operates under the Open World Assumption (OWA) (i.e. everything not explicitly declared true is unknown), this is clearly impossible. In [9], we have shown how defaults and inconsistencies can be handled in a non-monotonic framework utilizing context-dependent rules, under the OWA, i.e. without the need for consistency check. This, in turn, provides a high degree of tolerance in processing incomplete data as we show further in this article.

3. CONTEXT-DEPENDENT RULES: AN OVERVIEW

The general form of context-dependent rules is the following:

$<name_i>$ $(T_1,...,T_n)(P_1,...,P_m) \rightarrow A^U$, where

- $<name_i>$ is a reference to the rule. It is part of the justification for the rule's conclusion, A, to allow for more convenient identification of the reasoning path leading to that conclusion.
- $T_1,..,T_n$ define the *minimal context* required to support the plausibility of A. They are called T-premises for A.
- $P_1,...,P_m$ define additional evidence which, if present, will extend the minimal context and enforce the truth of A. They are called P-premises. If the set of P-premises is empty, then A is derived with maximum certainty with respect to that rule.

Context-dependent rules facilitate two types of relations between concepts:

- Relations that necessary hold, such as <Networking, Data-Structures> : Prerequisite (Required).
- Relations that possibly hold, such as <Networking, Web-Technologies> : Prerequisite (Recommended).

The conclusion of context-dependent rule, **A**, is the following data structure representing the justification (or evidence) for conclusion **A**:

A^{LV}: $(T_1,...,T_n)(P_1,...,P_m)(<name_i>)$, where:

- LV∈ {T, T*, U} defines the truthfulness of **A**. **T** means "necessarily true", **T*** means evidentially true, and **U** may mean unknown / uncertain / plausible depending on the context.
- $T_1,...,T_n$, are statements defining the minimal support for A. They must be logically or evidentially true statements, and $\neg A \notin \{T_1,...,T_n\}$. We call them the **T-set** of **A**.
- $P_1,...,P_m$ are statements that may provide additional support for **A** if they become true later. We call them the **P-set** for **A**.
- $<name_i>$ identifies the rule by which A was derived. The complete reasoning path to A can be easily defined by computing the transitive closure of the justifications for all statements in the T-set.

To illustrate the notion of the **context**, consider again the statement: "Networking class requires Data Structures class, but knowledge on Web Technologies and Computer Architecture is expected." This statement translates into the following context-dependent rule:

$<name_i>$(Data-Structures)(Web-Technologies, Computer-Architecture) $\rightarrow$ $Networking^U$

The conclusion, **Networking**, can be derived in one of the following contexts:

- $Networking^U$: (Data-Structures) (Web Technologies, Computer-Architecture) ($<name_i>$), i.e. the T-premise holds, but none of the P-premises hold. This defines the **minimal context**, where the degree of truthfulness of the conclusion is nominal.
- $Networking^{T*}$:(Data-Structures, Web Technologies, Computer-Architecture) ( ) ($<name_i>$), i.e. the T-premise and both P-premises hold. This defines the **maximal context**, where the truthfulness of the conclusion is the highest with respect to that rule.
- $Networking^U$: (Data-Structures, Web-Technologies) (Computer-Architecture) ($<name_i>$), i.e. the T-premise and one of the P-premises hold (either Web-Technologies, or Computer-Architecture). The T-premise and the satisfied P-premises define the **context** in which the conclusion holds, and the truthfulness of the conclusion is evaluated with respect to this context.

4. PROCESSING OF INCOMPLETE QUERIES

Definition A set of context-dependents rules describing minimal contexts for their respective conclusions is called the **R-set** (rule set).

Definition A set of statements with their associated justifications describing individual objects in the domain of interest is called the **B-set** (belief set).

It is important to note that the B-set may contain inconsistent statements such as A^T : $(T_1,...,T_i)$ ( ) $(name_i)$ and $\neg A^T$: $(T_j,...,T_k)$ ( ) $(name_j)$. As stated above, some inconsistencies may be impossible to resolve or ignore, in which case they must be defined as part of ontological knowledge. They are recorded as new statements of the form $C_A{}^U$: (A, ¬A) (C_A) $(name_i, name_j)$, which results in the following revision of both B- and R-sets required to ensure the soundness of ontological knowledge.

Revision of the R-set

For each C_X^U: (X, ¬X) (C_X) ($name_i$, $name_j$) ∈ B-set, the following transformations take place:

$<name_i>$ (X, T_1,...,T_i)( ) ➔ $A^T \Rightarrow$
$<name_i\text{-rev}>$ (X, T_1,...,T_i)(C_X) ➔ A^U
$<name_j>$ (¬X, T_j,...,T_n)( ) ➔ $A^T \Rightarrow$
$<name_j\text{-rev}>$ (¬X, T_j,...,T_n)(C_X) ➔ A^U
$<name_i>$ (X, T_1,...,T_i)(P_1,...,P_m) ➔ $A^U \Rightarrow$
$<name_i\text{-rev}>$ (X, T_1,...,T_i)(P_1,...,P_m, C_X) ➔ A^U
$<name_j>$ (¬X, T_j,...,T_n)(P_1,...,P_m) ➔ $A^U \Rightarrow$
$<name_j\text{-rev}>$ (¬X, T_j,...,T_n)(P_1,...,P_m, C_X) ➔ A^U

Notice that uncertainty associated with the conclusions of the revised rules increases. Moreover, these conclusions cannot be further matched to T-premises of other rules, thus efficiently blocking the propagation of the inconsistency.

Revision of the B-set

For each C_X^U: (X, ¬X) (C_X) ($name_i$, $name_j$) ∈ B-set, the following transformations take place:

A^T : (X, T_1,...,T_i) ( ) ($name_i$) ⇒
A^U : (X, T_1,...,T_i) (C_X) ($name_i$, $name_j$)
A^T : (¬X, T_j,...,T_n) ( ) ($name_i$) ⇒
A^U : (¬X, T_j,...,T_n) (C_X) ($name_i$, $name_j$)
A^U : (X, T_1,...,T_i) (P_1,...,P_m) ($name_i$) ⇒
A^U: (X, T_1,..,T_i)(P_1,...,P_m, C_X) ($name_i$, $name_j$)
A^U: (¬X, T_j,...,T_n) (P_1,...,P_m) ($name_i$) ⇒
A^U: (¬X,T_j,..,T_n)(P_1,..,P_m, C_X)($name_i$, $name_j$)

Reasoning with context-dependent rules can be carried out in forward or backward chaining manner depending on the specific task. The forward chaining algorithm is the main reasoning tool in the described framework and it is discussed in [9]. Here we present a backward chaining algorithm which kicks off when forward chaining halts as a result of potential incompleteness in the B-set.

Backward chaining algorithm for finding the maximal support for a statement

Given domain description Σ = <B-set, R-set>, query δ, such that Σ |≠ δ and Σ |≠ ¬δ, δ^U: (T_1,...,T_i) (P_1,...,P_m) ($name_i$) ∈ B-set or/and $\neg\delta^U$: (T_j,...,T_p) (P_n,...,P_s) ($name_j$) ∈ B-set, and $|P_1,...,P_m| \cap |P_n,...,P_s| = \varnothing$, for each $P_i \in |P_1,...,P_m| \cap |P_n,...,P_s|$ do:

1. Find the minimum support for P_i to make it evidentially true. This may require the following revisions of B- and R-sets:
   P_i^U : (T_1,...,T_i) (P_1,...,P_m) ($name_i$) ⇒
   ⇒P_i^{T*} : (T_1,...,T_i, P_1,...,P_m) ( ) ($name_i$-rev), where
   < $name_i$-rev> (T_1,..,T_i, P_1,..,P_m) ( ) ➔ P_i^{T*}
2. Apply forward chaining with the revised R-set to gather additional evidence for either δ or ¬δ.
3. Repeat steps 1 and 2 until either δ or ¬δ is justified as evidentially true.
4. Return justifications for both δ and ¬δ to the user for evaluation.

An extended example is shown next to illustrate the presented algorithm in the context of the overall search for an answer.

5. EXAMPLE

This is a modified version of the domain description presented in [9] (the original example was inspired by the well-known "koala" ontology [15]).

"Most people are not nocturnal species, unless they are party lovers or live in the rain forest. People who leave in the rain forest are lemur researchers. Nocturnals are not hardworking unless they are lemur researchers. Students are people, and PhD students are students with BS degrees. Most people are hardworking, especially those living in the rain forest. Lemur researchers who live in the rain forest are nocturnal, so are party lovers. People who live in the rain forest typically are not party lovers. Most students who live at universities are party lovers. PhD students are typically hardworking, especially those who are lemur researchers. Hardworking people are successful. Most party lovers are not successful, especially those who are not hardworking."

Most of the concepts in this description are not precisely defined. This is why the domain as described cannot be translated exactly in OWL. However, the following is a semantically acceptable approximation of it.

1a. Person ⊆ ¬Nocturnal
1b. Person ∩ PartyLover ⊆ Nocturnal
1c. Person ∩ ∃hasHabitat.RF ⊆ Nocturnal
2. Person ∩ ∃hasHabitat.RF ⊆ LemurReseracher
3. Nocturnal ∩ ¬LemurResearcher ⊆ ¬ Hardworking
4. Student ⊆ Person
5. PhDStudent ⊆ Student ∩ ∃hasDegree.BS
6. Person ∩ ∃hasHabitat.RF ⊆ Hardworking
7. LemurResearcher ∩ ∃hasHabitat.RF ⊆ Nocturnal
8. PartyLover ⊆ Nocturnal
9. Person ∩ ∃hasHabitat.RF ⊆ ¬PartyLover
10. ∃hasHabitat.University ∩ Student ⊆ PartyLover
11. PhDStudent ∩ LemurResearcher ⊆ Hardworking
12. Hardworking ⊆ Successful
13. PartyLover ∩ ¬Hardworking ⊆ ¬Successful

A DL reasoner such as RACER will find this set of concept definitions inconsistent and no queries will be answered until all inconsistencies are repaired. An obvious inconsistency here is that party lovers are both nocturnal and not nocturnal. The problem is due to the fact that statement 1b defines an exception to statement 1a. We can repair this inconsistency by specializing 1a as follows:

1a*. Person ∩ ¬PartyLover ⊆ ¬Nocturnal

If **¬PartyLover** were the only exception to 1a. this would be an easy fix. However, in case of multiple exceptions (1c. here is

yet another exception to 1a. and many other exceptions, some of which not known in advance, are possible) we need a more flexible representation to represent and process defaults and some types of inconsistencies that may follow from there.

Here is the set of context-dependent rules that matches the concept inclusion axioms above, but is semantically more accurate description of the example domain.

<1> (Person) (¬PartyLover, ¬hasHabitat.RF) ➔¬Nocturnal^{U}
<2> (Person, hasHabitat.RF) ( ) ➔ LemurResearcher^{T}
<3> (Nocturnal) (¬LemurResearcher) ➔¬Hardworking^{U}
<4> (Student) ( ) ➔ Person^{T}
<5a> (PhDStudent) ( ) ➔ Student^{T}
<5b> (PhDStudent) ( ) ➔ hasDegree.BS^{T}
<6> (Person) (hasHabitat.RF) ➔Hardworking^{U}
<7> (LemurResearcher) (hasHabitat.RainForest) ➔Nocturnal^{U}
<8> (PartyLover) ( ) ➔ Nocturnal^{T}
<9> (Person) (hasHabitat.RF) ➔ ¬ PartyLover^{U}
<10> (Student) (hasHabitat.University) ➔ PartyLover^{U}
<11> (PhDStudent) (LemurResearcher) ➔ Hardworking^{U}
<12> (Hardworking) ( ) ➔ Successful^{T}
<13> (PartyLover) (¬Hardworking) ➔ ¬Successful^{U}

Consider the following query (notice that it cannot be answered in a DL-based framework because it does not fully describe the instance we are trying to derive):

$$\text{PhDStudent} \cap \text{hasHabitat.RF} \subseteq \text{Successful}$$

Processing of this query in the presented framework is carried out as follows:

Step 0: Define initial, possibly incomplete B-set.

PhDStudent^{T}: ( ) ( ) ( ), hasHabitat.RF^{T}: ( ) ( ) ( ),
Successful^{U}: ( ) ( ) ( ), ¬Successful^{U}: ( ) ( ) ( )

Apply rules whose T-premises match necessarily or evidentially true statements from the B-set and augment the later with newly derived conclusions, i.e. **B-set** = **B-set** ∪ Cons(**B-set, R-set**).

Step 1:
Student^{T}: (PhDStudent) ( ) (5a)
hasDegree.BS^{T}: (PhDStudent) ( ) (5b)
Hardworking^{U}: (PhDStudent)(LemurResearcher) (11)

Step 2:
Person^{T}: (Student) ( ) (4)
PartyLover^{U} : (Student)(hasHabitat.University) (10)

Step 3
¬Nocturnal^{U}: (Person) (¬PartyLover, ¬hasHabitat.RF) (1)
LemurResearcher^{T}: (Person, hasHabitat.RF) ( ) (2)
Hardworking^{T*} : (Person, hasHabitat.RF) ( ) (6)
¬PartyLover^{T*}: (Person, hasHabitat.RF) ( ) (9)

Notice that **LemurResearcher** was derived as a true statement, which causes rule 11 to be revised. Because rule 11 has already been applied in a different context, its conclusion must be updated to reflect the new evidence as follows:

Hardworking^{T*}: (PhDStudent, LemurResearcher)( ) (11rev)

Hardworking is now supported as a true statement by two independent justifications, namely rules 6 and 11rev. Notice that **¬PartyLover** was derived as a true statement which overrides **PartyLover** derived as a plausible statement by rule 10, and also causes rule 1 to be revised as well as its conclusion which now becomes:

¬Nocturnal^{U}: (Person,¬PartyLover)(¬hasHabit.RF) (1rev)

Step 4
Nocturnal^{T*}: (LemurResearcher, hasHabitat.RF)( ) (7-rev)

Step 5
Successful^{T*}: (Hardworking) ( ) (12)
¬Hardworking^{U}:(Nocturnal)(¬LemurResearch) (3)

Here **Successful** is supported as evidentially true by the following two justifications:

Successful^{T*}: (PhDStudent, LemurResearcher, Hardworking) ( ) (12, 11rev)
Successful^{T*}: (Person,hasHabitat.RF, Hardworking)( ) (12,6)

We were not able to derive any support for **¬Successful**, and therefore the initial query is answered "Yes".

Consider now the following query:

$$\text{PhDStudent} \cap \text{hasHabitat.University} \subseteq \text{Successful}$$

Step 0 (initial B-set)
PhDStudent^{T}: ( ) ( ) ( ), hasHabitat.University^{T}: ( ) ( ) ( ),
Successful^{U}: ( ) ( ) ( ), ¬Successful^{U}: ( ) ( ) ( )

Step 1
Student^{T}: (PhDStudent) ( ) (5a)
hasDegree.BS^{T}: (PhDStudent) ( ) (5b)
Hardworking^{U} : (PhDStudent) (LemurResearcher) (11)

Step 2
Person^{T}: (Student) ( ) (4)
¬PartyLover^{U} : (Person) (hasHabitat.RF) (9)
PartyLover^{T*} : (Student, hasHabitat.University) ( ) (10)

Step 3
¬Nocturnal^{U}:(Person)(¬PartyLover, ¬hasHabitat.RF) (1)
Hardworking^{U}: (Person) (hasHabitat.RF) ( ) (6)
Nocturnal^{T}: (PartyLover) ( ) (8)
¬Successful^{U}: (PartyLover) (¬Hardworking) (13)

Step 4
¬Hardworking^{U}: (Nocturnal) (¬LemurResearcher) (3)

No further evidence for **Successful** or **¬Successful** can be derived. To complete the evidence for one of them, the backward chaining algorithm presented in the previous section must identify a stronger justification for **¬Hardworking** or find a justification of Hardworking. Currently, **¬Hardworking** is weakly supported by rules 6 and 11. If **¬LemurResearcher** can be derived or supported as a true statement, it will provide additional evidence for **¬Hardworking**. Context specific question about **LemurResearcher** will confirm or reject **¬Hardworking**. Let the answer be **LemurResearcher**. Then, rules 7 and 11 must be revised to reflect this new evidence resulting in the following changes in the B-set:

Hardworking^{T*}: (PhDStudent, LemurResearcher) ( ) (11rev)
Nocturnal^{U}: (LemurResearcher) (hasHabitat.RF) (7rev)

Thus, Successful^{T*}: (Hardworking) ( ) (12)

Notice that although there is evidence supporting **¬Successful**, the evidence for **Successful** overrides it, because the later is justified as evidentially true (vs. plausible for **¬Successful**).

Assume now that the answer to the question was **¬LemurResearcher**. Then, rules 3 and 13, respectively, will be revised to reflect the new evidence resulting in the following justifications:

¬Hardworking^{T*}: (Nocturnal, ¬LemurResearcher) ( ) (3rev)
¬Successful^{T*}: (PartyLover, ¬Hardworking) ( ) (13rev)

Thus**, ¬Successful** will now be justified as a true statement with respect to the given context.

6. CONCLUSION

We advocated in this paper that many Semantic Web applications will have to deal with complex conjunctive queries, and that such queries should not be expected to be completely and accurately specified. We discussed why conjunctive queries are hard for Description Logic reasoners and how the context-dependent reasoning framework utilizing context-dependent rules can be extended to handle them. An example was presented to illustrate how the proposed framework identifies, maintains and interprets contexts to answer incomplete query specifications.

Acknowledgement. This work was partially supported by a CSU-AAUP research grant.

7. REFERENCES

[1] http://www.w3.org/TR/owl2-overview/
[2] F. Baader, F., D. Calvanese, D. McGuinness, D. Nardi, and P. Patel-Schneider (eds). **The Description Logic Handbook – Theory, Implementation and Applications.** Cambridge University Press, 2003.
[3] http://www.racer-systems.com
[4] http://owl.man.ac.uk/factplusplus/
[5] X. Deng, Explanation and Diagnosis Services for Unsatisfiability and Inconsistency in Description Logics. Ph.D. Thesis, Concordia University, Montreal, 2010.
[6] M. Horridge, B. Parsia, and U. Sattler -- Laconic and Precise Justifications in OWL. In **Proc. ISWC '08 Proceedings of the 7th International Conference on The Semantic Web (ISWC'08)**, 2008.
[7] S.Gomez, I. Chesnevar, and G. Simari – An Argumentative Approach to Reasoning with Inconsistent Ontologies. In **Proc. Knowledge Representation Ontology Workshop KROW'2008**, Sydney, Australia.
[8] N. Zlatareva – Context - dependent Reasoning for the Semantic Web. **Journal of Systemics, Cybernetics and Informatics**, vol.9, number 4, 2011, IIIS Press.
[9] N. Zlatareva – Managing Exceptions, Defaults, and Inconsistencies in Semantic Web Ontologies. In **Proc.14th International Conference on Artificial Intelligence and Soft Computing**, Crete, Greece, June, 2011.
[10] Liu, B., J. Li, and Y.Zhao – A Query-specific Reasoning Method for Inconsistent and Uncertain Ontology, In **Proc. International Multi Conference of Engineers and Computer Scientists (IMECS'2011)**, vol.1, March 16-18, 2011, Hong Kong.
[11] Z. Huang, van Harmelen, F., and ten Teije, A. Reasoning with Inconsistent Ontologies: Framework, Prototype, and Experiment. **Semantic Web Technologies: Trends and Research in Ontology-Based Systems** (eds. J. Davies, R. Studer, P. Warren), John Wiley & Sons, 2006.
[12] G. Stoilos, B. C. Grau and I. Horrocks -- How Incomplete is your Semantic Web Reasoner? In **Proc. 24th AAAI**, 2010.
[13] P. Bonatti, C. Lutz and F. Walter – Description Logics with Circumscription, In **Proc. Principles of Knowledge Representation and Reasoning KR'2006**.
[14] L. Giordano, V. Gliozzi, N. Olivetti, and G. L. Pozzato– Reasoning about Typicality in Preferential Description Logics. In **Proc. 11th European Conference on Logics in Artificial Intelligence JELIA'2008**, 2008.
[15] http://protege.stanford.edu/plugins/owl/owl-library/koala.owl