SEMANTIC CONFLICT DETECTION IN META-DATA – A RULE BASED APPROACH

by

KARTHIKEYAN GIRILOGANATHAN

(Under the Direction of I. BUDAK ARPINAR)

ABSTRACT

The Web has become a source of reference for information on many subjects. Also the ability to extract semantic meta-data from Web resources has increased tremendously in recent years. Effective use of this meta-data by the users can be affected by conflicts among the meta-data. In this context, we propose a new semi-automatic process using rules to detect conflicts. This meta-data can be represented in either of RDF(S), DAML or OWL and the rules are represented in RuleML. Furthermore, our technique can identify conflicts among the data at different granularities using a Relationship Ontology to simplify complex meta-data. We also describe a prototype implementation and an evaluation of this approach on a real-world dataset extracted from various Web resources.

INDEX WORDS: Semantic Web, Trust, Conflict, Rule, RuleML, RDF, RDFS, DAML, OWL, and Logic

DETECTING CONFLICTS IN SEMANTIC META-DATA – A RULE BASED APPROACH

by

KARTHIKEYAN GIRILOGANATHAN

B.E, Anna University, India, 2000

A Thesis Submitted to the Graduate Faculty of The University of Georgia in Partial Fulfillment of the Requirements for the Degree

MASTER OF COMPUTER SCIENCE

ATHENS, GEORGIA

2004

SEMANTIC CONFLICT DETECTION IN META-DATA – A RULE BASED APPROACH

by

KARTHIKEYAN GIRILOGANATHAN

Major Professor:	I. BUDAK ARPINAR
Committee:	AMIT SHETH KHALED RASHEED

Electronic Version Approved:

Maureen Grasso
Dean of the Graduate School
The University of Georgia
May 2004

DEDICATION

I dedicate this work to my parents and brother who sacrificed a lot to get me here. I also dedicate this to all my friends who have given me moral support when I went through difficult times. I also dedicate this to the scientific community that strives to make this world a better place for all.

ACKNOWLEDGEMENTS

I would like to thank the members of the LSDIS Lab, University of Georgia for their valuable feedbacks and insights. I would like to thank my advisor Dr. Budak Arpinar whose vision made this work possible. I also thank the members of my committee, Dr. Amit Sheth and Dr. Khaled Rasheed for their intellectual and technical guidance. I would like to express my thanks to Boanarges Aleman-Meza and Chris Halaschek for providing me the testbed ontology and helpful comments. I would also like to thank all the scientists in the semantic Web community, whose contributions enabled this work.

TABLE OF CONTENTS

Page

LIST OF FIGURES

Page

CHAPTER 1

INTRODUCTION

Today a massive amount of data is available on the Internet, as well as in private organizational databases. The volume of this data is increasing continuously. However, despite the abundance of information, most of it cannot be used effectively for decision-making purposes causing knowledge starvation. The focus of contemporary data and information retrieval systems has been to provide efficient support for the querying and retrieval of data [1, 4]. Due to the increasing move from data to knowledge, and the increasing popularity of the vision of the semantic Web, there is significant interest and ongoing work, in automatically extracting and representing the metadata as semantic annotations to documents and services on the Web. The Semantic Web aims to represent information in the World Wide Web for it to be processed by machines not just for display purposes, but also for automation, integration, and reuse across applications. Given these developments, the stage is now set for the next generation of technologies in information retrieval, which will facilitate getting actionable knowledge and information from massive data sources.

The consumer of such actionable knowledge could be a human user or an application. A human user should be made aware when he is dealing with contradicting information because this can have significant impact in a decision-making process. When an application has to deal with conflicting data it should have logic built into it to make a decision choice either automatically or through a user input. Our work focuses on identifying conflicts in the semantic

meta-data to facilitate a user or an application to reach proper conclusions. The resolution of such conflicts is beyond the scope of this work.

Researchers have developed the semantic Web languages like RDF [22], RDFS [13], DAML+OIL [23] and OWL [24], which provide for knowledge representation, querying and inferencing. Underlying all these languages is the basic concept of *triples*. That is, subject, predicate, and object, as in "*Anna motherOf John*". Therein, our discussion and definitions will be based on this basic unit of information or knowledge. In fact, using information extraction techniques on structured and non-structured documents we are able to convert the knowledge into one of these semantic Web languages such as RDF in PISTA (Passenger Identification, Screening, and Threat Analysis) application [2] and its successor SemDIS (Semantic Discovery: Discovering Complex Relationships in Semantic Web)[3]. SemDIS is a scalable system that finds semantic associations between two entities from a massive amount of knowledge extracted from public sources (e.g., relations between two persons). These semantic associations are then ranked based on relevance, trust and other parameters [3].

Conflicts can occur between RDF statements (i.e., triples), or between sets of RDF statements. If conflicts occur between two RDF statements we identify the conflicts through rules that specify if these two statements can coexist or not. These rules are defined by domain experts considering the semantics of entities and relations involved in these statements. Thus, they can be part of the ontologies as universal and agreed constraints (e.g., *maxCardinality*, *disjoint* etc.) or externally specified if they are context dependent and not universally agreed upon. If the conflicts occur between sets of RDF statements, we compare the sets of statements (i.e., complex relations) after reducing them to a single statement through a simplification process. For this simplification process we use a Relationship Ontology (RO) for explicit

specification of relations among relations. Finally, a conflict query on the knowledgebase that is built using extracted data from Web resources provides the statements that are in conflict together with the cause of conflict. By the cause of conflict we refer to the RDF statements that violated the rule.

Thus, our contributions in this work can be summarized as follows:

- A formalization of conflicts and their classification for the semantic meta-data on Web.
- A rule-based approach based on RuleML [12] to define and identify conflicts automatically.
- A Relationship Ontology to equate granularities of complex meta-data for conflict checking.

The rest of this thesis is organized as follows: Chapter 2 provides the motivating factors behind this work. Chapter 3 provides the definition and classification of conflicts. Chapter 4 describes the using rules for simplification and conflict identification. Chapter 5 discusses the concept of Relationship Ontology and how it enables simplification and conflict detection. Chapter 6 presents the system architecture and initial experimental results. Chapter 7 compares our approach to related work. Chapter 8 summarizes our contributions and future research directions.

CHAPTER 2

MOTIVATION AND BACKGROUND

This work has been motivated by the focus of semantic Web research for representing the data in a machine processable format and therefore a more efficient analysis of the data. The result of such an analysis will yield actionable information (with associated sources and supporting evidence) to a user or an application. The main idea of identifying conflicts is one such analysis which would ensure that the retrieved information is dependable (i.e., trusted).

2.1 SemDIS

This work aims provide a complimentary capability for the SemDIS[1] project which is being developed to query and analyze massive amount of meta-data collected from various Web resources. In SemDIS, a user interacts with a populated ontology (SWETO) [38] through a knowledge discovery-driven approach that combines search and inferencing, enabling more complex analysis and deeper insight. With this infrastructure in place, tools and algorithms have been developed to automatically identify and rank complex relationships between entities in this semantically annotated data. Figure 1 highlights the key components in the SemDIS architecture.

[1] This project is funded by National Science Foundation under Grant No. IIS – 0325464 titled "SemDIS: discovering Complex Relationships in Semantic Web". Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not reflect the views of the National Science Foundation.

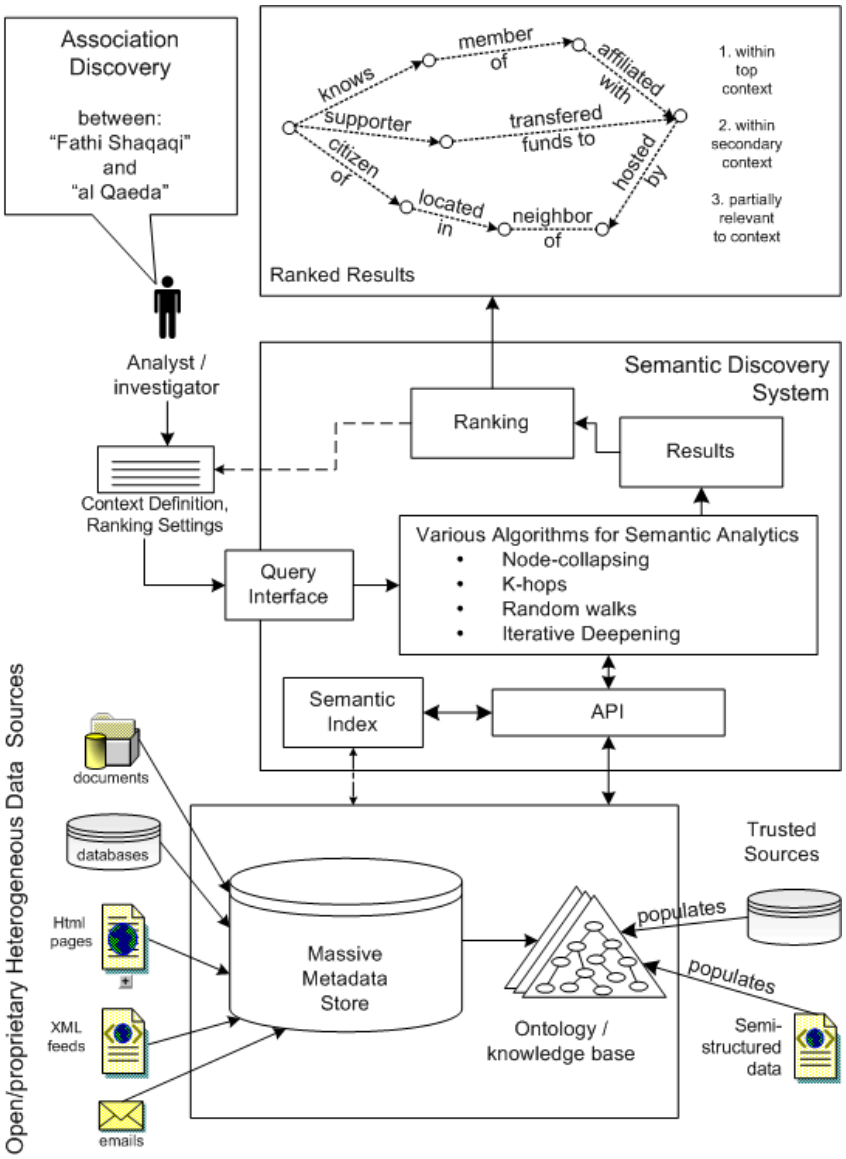


Figure 1: SemDIS System Architecture

As part of the SemDIS, a Semantic Web Evaluation Ontology (SWETO) [38] has been developed and populated with data collected from Web to generate a real world knowledge base. The conflict identification techniques outlined in this thesis are tested on this ontology and knowledgebase which included over 6000 entities and more than 11,000 explicit relations among them at the time of the tests.

2.2 Logic And Rules

Analysis of the data as outlined above requires reasoning capability. This can be implemented using database techniques enriched with heuristics. Another approach would be to use logic based techniques. The semantic Web logic layer highlights the fact about using rules in achieving this goal.

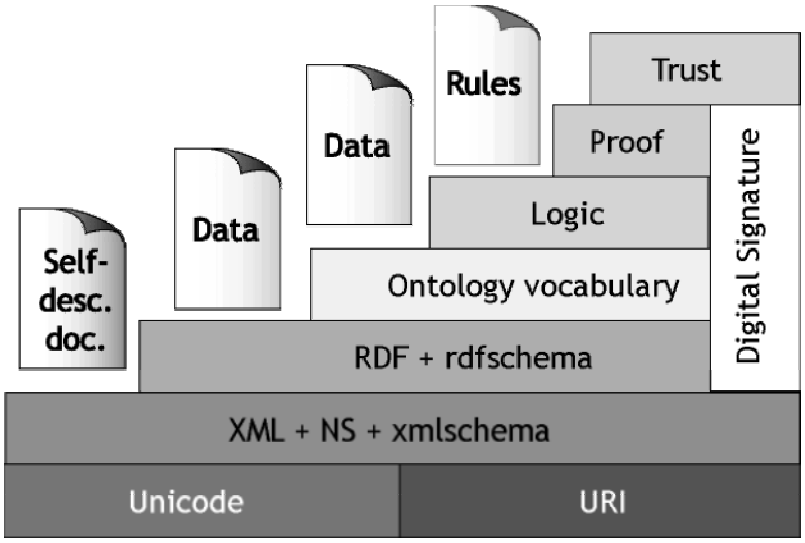


Figure 2: Semantic Web Stack

For this thesis we have taken the logic-based approach and used rules to identify conflicts. The layers above the RDF schema are geared towards inferencing. Markup languages (e.g., RuleML[12], SWRL (Semantic Web Rule Language)[32]) have been developed to specify rules that help in inferencing on semantic meta-data. Futhermore, *inconsistency checking* has been stated as an important part of the requirements for the OWL language. The OWL design document justifies the requirement as follows:

> "The Web is decentralized, allowing anyone to say anything. As a result, different viewpoints may be contradictory, or even false information may be provided. In order to prevent agents from combining incompatible data or from taking consistent data and

evolving it into an inconsistent state, it is important that inconsistencies can be detected automatically." (http://www.w3.org/TR/webont-req/#goal-inconsistency).

Our work can be perceived as a consistency checking approach in a limited form.

CHAPTER 3

DEFINITION AND CLASSIFICATION OF CONFLICTS

Semantic metadata can be described as the content from unstructured and structured documents enriched with semantic annotations to enable a disparate collection of content items to be explored and analyzed as a single, interconnected repository [3]. The content we are using is the semantic metadata from the Web. The Web is an unmonitored publishing environment where anyone can say anything they want. A tool that generates metadata using the Web is responsible for making sure that the repository does not contain contradicting or conflicting information which will affect the credibility and trustworthiness of the repository.

Before presenting conflict definitions, the terminology used in the definitions and the succeeding chapters is presented in the table below.

t	A single triple
T	A set of triples
S	A function denoting the process of *simplification*
s	The result of *simplification* (S(T)→s), could be a single triple or again a set of triples
U	Constraints expressed in an ontology, e.g., the property 'biologicalMother' is unique
E	Constraints supplied by an expert, e.g., person(x) can never do action(y)

Definition 1: Two sets of triples T1 and T2 are said to be in *conflict* if their *simplification*s S(T1)→s1 and S(T2)→s2 are mutually non-agreeable.

Definition 2: Two *simplification*s s1 and s2 are mutually *non agreeable* if taken together they are in violation of U or E.

A *simplification* s is the result of any function S that reduces the complexity of a set of triples but still preserves the meaning. The definitions take into consideration the granularity of information (a set of triples or a single triple). Some conflicts may occur at the level of triples (i.e., subject, predicate, and object). Yet other conflicts may be between complex relations instead of triples where a complex relation may span several triples. Thus an initial step in conflict identification is to reduce the granularity level of information to be compared o the same level. This is done by a technique which we name *simplification*.

The definitions of the function S, the constraints U, and E enable flexibility in conflict definitions by users (e.g., domain experts) because, they are mainly subjective in their nature. We have used them in the context of conflict detection but someone else can use them for inferring from the existing knowledge. Also a choice of constraints from U, or E can be based upon the domain of interest, purpose of the analysis etc. We do not use the term 'mutually exclusive' in conflict definitions, because we want to allow for levels of disagreement. For example, triples about a person that state that he is a champion in both gymnastics and wrestling are mutually non-agreeable. It is intuitively non-agreeable for us that a person can excel at both sports. Therefore by signaling a certain level of disagreement it is possible *hidden* inconsistencies in the data can be detected.

Intuitively, a set of information on a given topic can be reduced to simpler knowledge until there is no further reduction possible. The resulting knowledge of this process of reduction is what we call *simplification*. For example, given a set of facts about a person's characteristics

we can draw a conclusion about whether s/he is trustworthy of to what degree. That is, an estimate of trustworthiness by simplification. We use the idea of simplification to identify conflicts among complex relations by reducing them to triples. In terms of RDF we consider three types of simplifications:

1) An RDF triple is a *simplification* because it is the most basic piece of knowledge.
2) We might be able to compose relations [11] to a single relation between a subject and an object. Let E denote the set of entities and P denote the set of relations in a set of statements of an RDF graph: $E = \{e_1, e_2 ... e_n\}$, $P = \{p_1, p_2 ... p_m\}$. Let P^+ be the power set of P. Then, $P^+ = \{(p_1), (p_2) ... (p_m), (p_1, p_2) ... (p_1, ..., p_m)\}$. Let C be a subset of P^+ consisting of only groups of relations that can be composed to a single relation, that is, $C = \{(p_1, p_k), ..., (p_a, p_b, p_c, ...)\}$. Let R be the set of relations obtained by substituting the composed relation for the composable relations, then $R = \{r_1, r_2 ... r_n\}$, where $r_1, r_2 ... r_n$ are results of the composition. The statement $(e_i\ r_k\ e_j)$ is a *simplification* if $r_k \in R$ *and* $e_i, e_j \in E$.

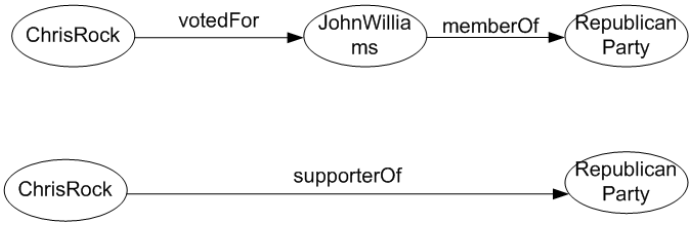


Figure 3: Example for composition

In the example shown in Figure 3 the statement "ChrisRock supporterOf RepublicanParty" is a simplification because the relation 'supporterOf' is a result of composition of the relations 'votedFor' and 'memberOf'. For this type of simplification to work, there needs to be a mechanism wherein we can specify that a relation is the composition of several given

relations. We propose to use the concept of a Relationship Ontology, discussed in Chapter 4, for this purpose.

3) There could be background knowledge based simplifications of the form $statement_1 \wedge statement_2 \wedge \ldots statement_n \rightarrow statement_t$. In this case $statement_t$ is a simplification. This type of simplification will depend on expert knowledge.

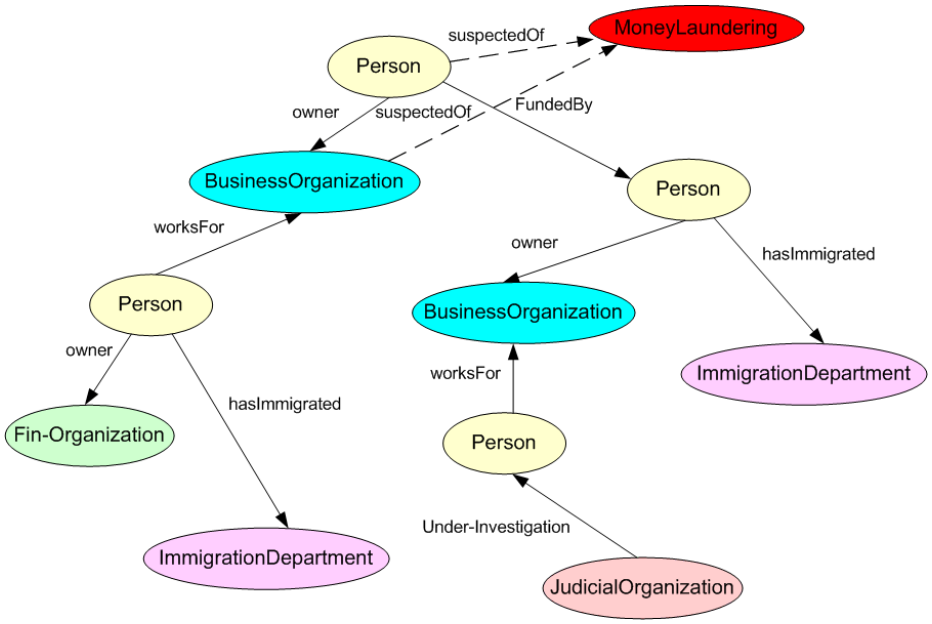
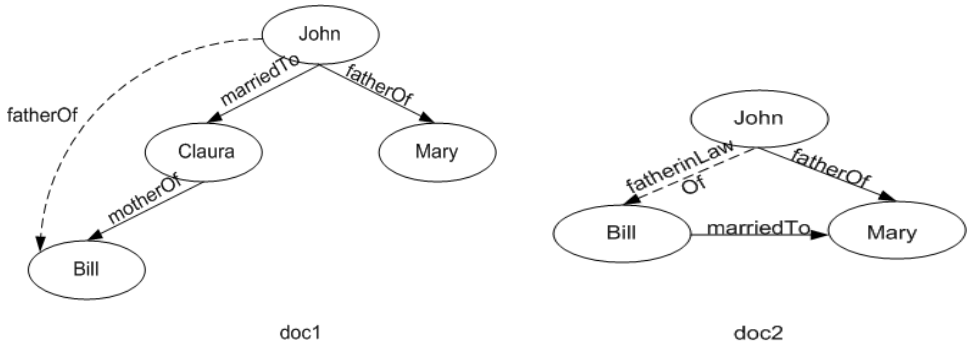


Figure 4: Statement simplification example

A typical money laundering scenario is shown in Figure 4. This sub-graph (set of triples) tries to capture from the knowledge base, instances of an immigrant making multiple deposits in a financial organization and working for a business organization that is owned by somebody well known to the owner (who is an immigrant) of another business organization that employs peoples under investigation by a judicial organization such as the FBI. The dotted lines show some possible simplifications that can be done on this set of triples. This simplification is possible only through an expert's knowledge involving these subjects. Note that this type of

simplification is different than the relationship composition in the previous item where a series of nodes are assembled and the end points do not change in the composition. However here the simplification results in a totally new statement with potentially new nodes (e.g., 'MoneyLaundering' is not part of initial set of nodes). The idea of simplification and its relation to logic are further discussed in section 4.1.2 when we discuss about propositional logic.



(a) (b)

Figure 5: RDF sub-graphs to illustrate simplification and conflict

In order to illustrate the concept of conflict in terms of simplification, consider the two sets of statements of Figure 5. Based on our definition of simplification:

- every statement is a simplification,
- each relation that can be composed (or implied from explicit statements) results in another simplification

By composing the relations, *marriedTo* and *motherOf* into the relation *fatherOf* we get the simplification "John *fatherOf* Bill" (dotted line in Figure 5a). The resulting simplification "John *fatherOf* Bill" and the existing simplification "John *fatherInLawOf* Bill" are mutually non-agreeable. Thereby they are considered to be in conflict. This enables a refinement and validation

of the meta-data. For example, it could be the case that for legal purposes it needs to be clear whether the relationship is *fatherOf*, *fatherInLawOf*, or something else like *stepFatherOf*.

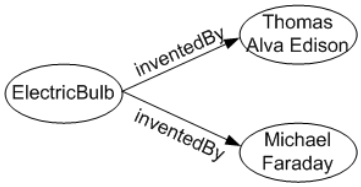
3.1 CONFLICT TYPES

Conflicts can be classified based on the type of assertion that the simplifications violate. In the following sections the use of prefixes rdf, rdfs, daml, owl refers to the respective namespaces. When we use some existing constraints from these semantic Web languages we use appropriate prefixes.

1) Property assertion conflicts

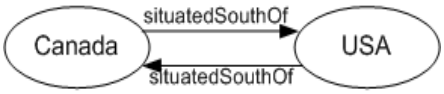
These conflicts occur when the constraints placed on a property *p* are violated. These constraints are defined with namespaces of these semantic Web languages for metadata.

- If *p* has a 'daml:uniqueProperty' or 'owl:functionalproperty' constraint, then the simplifications (e_1 *p* e_2) and (e_1 *p* e_3) are in conflict, e.g.



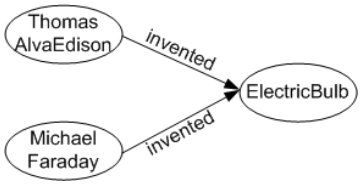
  If a property has 'daml:uniqueProperty' or 'owl:functionalproperty' constraint, it cannot connect a single subject to two different objects. In the example 'inventedBy' is specified as a 'daml:uniqueProperty' or 'owl:functionalProperty'. So the two triples (ElectricBulb inventedBy ThomasAlvaEdison) and (ElectricBulb inventedBy MichaelFaraday) are in conflict.

- If *p* is 'asymmetric', then the simplifications (e_1 *p* e_2) and (e_2 *p* e_1) are in conflict, e.g.



If a property has 'asymmetric' constraint it cannot connect a subject to an object and vice versa (i.e., in both the forward and the reverse directions). In the example 'situatedSouthOf' is specified as a 'asymmetric' property. So the triples (Canada situatedSouthOf USA) and (USA situatedSouthOf Canada) are in conflict.

- If *p* has a 'daml:unambiguous' or 'owl:inverseFunctional-Property' restriction, then the simplifications (e_1 *p* e_2) and (e_3 *p* e_2) are in conflict, e.g.

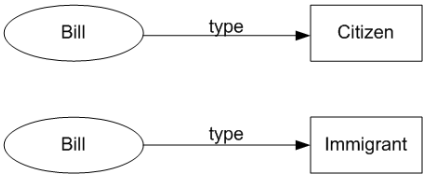


If a property has 'daml:unambiguous' or 'owl:inverseFunctional-Property' restriction then it cannot connect two different subjects to a single object. In the example 'invented' is specified as 'daml:unambiguous' or 'owl:inverseFunctional-Property'. So the triples (ThomasAlvaEdison invented ElectricBulb) and (MichaelFaraday invented ElectricBulb) are in conflict.

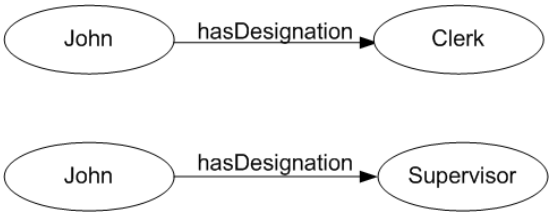
2) Class assertion conflicts

These conflicts occur when constraints placed on classes are violated. We consider here the type of assertions possible using DAML and OWL.

- If classes c1 and c2 are defined as 'daml:disjoint' or 'owl:disjoint', then "x *subclassOf* c1" and "x *subclassOf* c2" signal a conflict. Similarly, the relations *type or isA* used with the same entity over disjoint classes signals a conflict. For example, if class 'Citizen' and 'Immigrant' are disjoint then "Bill *type* Citizen" and "Bill *type* Immigrant" are in conflict.

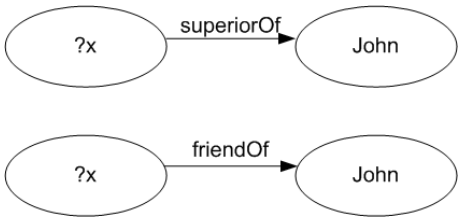


- If a class 'Employee' has a OWL or DAML restriction 'maxCardinality' of '1' on a relation '*hasDesignation*', and John is an instance of the employee class, then "John *hasDesignation* clerk" together with "John *hasDesignation* supervisor" signals a conflict.



3) Statement assertion conflicts

Here we make the assertion that under specific conditions the given statements are in conflict. These are based upon background expert knowledge. This is to be differentiated from the previous conflicts where there were violations of assertions on relations and classes. For example, assume that we want to say that a person cannot be a superior and a friend at the same time to "John".

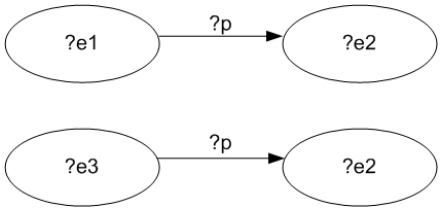


Thus the statements "x *superiorOf* John" and "x *friendOf* John" are in conflict. We use a '?' mark on 'x' to show that 'x' can be replaced by an instance from the knowledgebase. For expressing this kind of conflict, we define rules in RuleML (explained in Chapter 4).

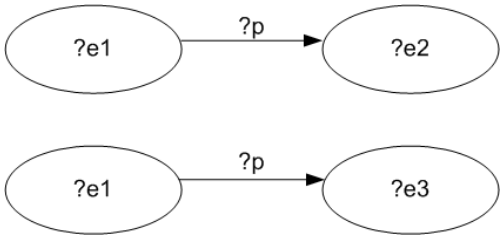
4) Non-assertional conflicts

These are subjective conflicts, that is, there are no explicit constraints defined on the relations or statements involved but the information maybe in conflict. In this case, the conflict is subjective (or context dependent). For example, consider the statements "Jim *isA* gymnastics champion" and "Jim *isA* wrestling champion". An ontology may allow both statements to exist. But subjectively a wrestler cannot be expected to be gymnastics champion or vice versa. These types of conflicts would be given the least priority when the results are presented to the user. An incomplete list of this type of conflicts includes the following:

- The simplifications (e_1 p e_2) and (e_3 p e_2) are in subjective conflict because the subject is different but is related to the object through the same relation.



- The simplifications (e_1 p e_2) and (e_1 p e_3) are in subjective conflict because the subject is related through same relation to different objects.



Note that this type of conflicts is different than the previous one that is still captured through assertions. However, this last type of conflicts does not depend on any defined assertions. This is evident from the figures where the entities (e1,e2,e3) and the property (p) are prefixed with a '?' mark to denote that they can be replaced with any instance from the knowledgebase.

We have defined four types of conflicts based on the type of assertion that the simplifications violate. When the assertions have been defined, the process of identifying the conflicts is mechanical in nature needing no human intervention. Conflict types 1,2 and 4 can be detected automatically based on the ontology used. The type of assertion that conflict type 3 violates is not expressible as part of an ontology. We require human intervention to enter these constraints and the mechanism we provide is discussed in Chapter 5.

CHAPTER 4

RULES FOR IDENTIFYING CONFLICTS

This section describes the role of rules and the applicability of rules in identifying conflicts.

4.1 Background

A background discussion of what rules are and how they are related to logic and semantic Web will be helpful to understand the significance of using rules in identifying conflicts. This will give a picture of how and why using rules is justified.

4.1.1 Knowledge Representation

The notion of knowledge representation can best be understood in terms of five distinct roles [33] it plays, each crucial to the task at hand:

1. A knowledge representation (KR) is most fundamentally a *surrogate*, a substitute for the thing itself, used to enable an entity to determine consequences by thinking rather than acting, i.e., by reasoning about the world rather than taking action in it.

Any intelligent entity that wishes to reason about its world encounters an important, inescapable fact: reasoning is a process that goes on internally, while most things it wishes to reason about exist only externally. A program (or person) engaged in planning the assembly of a bicycle, for instance, may have to reason about entities like wheels, chains, sprockets, handle bars, etc., yet such things exist only in the external world.

2. It is a set of *ontological commitments*, i.e., an answer to the question: In what terms should I think about the world?

Selecting a representation means making a set of ontological commitments. The commitments are in effect a strong pair of glasses that determine what we can see, bringing some part of the world into sharp focus, at the expense of blurring other parts. A KR *is* a set of ontological commitments. It is *unavoidably* so because of the inevitable imperfections of representations. It is *usefully* so because judicious selection of commitments provides the opportunity to focus attention on aspects of the world we believe to be relevant.

3. It is a *fragmentary theory of intelligent reasoning*, expressed in terms of three components: (i) the representation's fundamental conception of intelligent reasoning; (ii) the set of inferences the representation *sanctions*; and (iii) the set of inferences it *recommends*.

Where the sanctioned inferences indicate what can be inferred at all, the recommended inferences are concerned with what should be inferred. Where the ontology tells us how to see, the recommended inferences suggest how to reason.

4. It is a medium for pragmatically efficient computation, i.e., the computational environment in which thinking is accomplished.

From a purely mechanistic view, reasoning in machines is a computational process. Simply put, to use a representation we must compute with it. As a result, questions about computational efficiency are inevitably central to the notion of representation. There is always a trade off between expressive power and computational efficiency. We ignore computational considerations at our peril, but we can also be overly concerned with them, producing representations that are fast but inadequate for real use.

5. It is a *medium of human expression*, i.e., a language in which we say things about the world.

In our case, since we are dealing with documents that conform to the semantic Web language formats, the knowledge representation is in the form of an ontology. As per the description of the standards such as RDF, the schema and the instances are part of the ontology. For our system we assume that we can extract instance data from the Web and convert it to RDF documents using the vocabulary (schema) of the ontology. In fact this is being done in a very efficient way by the commercial product, 'Freedom' by Semagix Inc [].

In light of the above definition of Knowledge Representation, logic can be referred as a type of knowledge representation technique. A brief discussion of the different types of logic will give a proper perspective of using rules.

4.1.2 Propositional Logic

Propositional logic can be defined as a system of symbolic logic using symbols to stand for whole propositions and logical connectives [34]. A proposition is a statement like 'all men are mortal'. The letters P and Q stand for such propositions in the following discussion.

Inference is defined as generating new knowledge from existing knowledge. We will discuss inference rules for propositional logic which is what we want to highlight in the context of this paper. Our detection of conflicting statements is an inference based on the defining constraints as expressed in Chapter 3. Similarly our idea of using simplification can be extended to include all the inference allowed under the chosen format of RDF/RDFS/OWL. The following are common inference rules under Propositional logic which motivate us to develop rules for simplification:

$$\text{Modus Ponens: } \frac{P, P \rightarrow Q}{Q}$$

If proposition P is true and given that Q is true whenever P is true, we conclude that Q is true.

AND Introduction: $\frac{P_1, P_2 \cdots P_n}{P_1 \wedge P_2 \wedge \cdots P_n}$

If propositions P_1, P2,… Pn are true then we can claim that the complex sentence P1 AND P2 AND … AND Pn is true.

AND Elimination, OR Introduction and NOT Elimination are other inference rules supported by propositional logic.

Given a set of such sanctioned inference rules we can infer new knowledge from existing knowledge. We also need to provide a proof of how such an inference was made. A proof is a list of statements that are either part of the knowledge base or can be inferred from the knowledgebase along with the inference rule applied at each stage:

Given: $S, (S \vee P) \rightarrow (Q \wedge R),\ X$

To show: $Q \wedge X$

Proof:

S	Given
$S \vee P$	OR introduction
$(S \vee P) \rightarrow (Q \wedge R)$	Given
$Q \wedge R$	Modus ponens
Q	AND elimination
X	Given
$Q \wedge X$	AND introduction

We also follow the same approach when presenting the result to the user when conflicts have been identified. We call this the *derivation tree*. In the case of simplification this tree will show

what rules caused the simplification and which statements are parts of the summary. In the case of conflicts this tree will identify the rules that triggered this conflict and the statements that satisfied those rules.

4.1.3 Predicate Logic

In proposition logic the internal structure of a proposition itself is never analyzed. The proposition is the lowest unit of representation. Predicate logic addresses this issue. Predicate logic [34] is an extension of propositional logic with separate symbols for predicates, subjects and quantifiers. The following are some examples of sentences in predicate logic:

- Sentences

  friends(Alison, Richard) likes(Alison, Richard)

- Sentences with quantifiers

  $\exists X$ bird(X) $\Lambda \neg$flies(X)
  There exists some bird that does not fly

  $\forall X$ (person(X) $\rightarrow \exists$Yloves(X,Y))
  Every person has something that they love

In addition to the rules of propositional logic, predicate logic also has its own set of inference rules like Modus Tollens, chain argument, disjunctive argument, conjunctive argument, reduction ad absurdum etc. Explaining each of them will be a digression. So we will talk about the aspect of predicate logic that we modeled this work on. We can see from the predicate logic sentences, how close they are to RDF statements (i.e., subject, predicate, and object). So it stands to reason that we can use some ideas from the results on predicate logic. There are two main proof procedures to express the inferences made based on predicate logic. They are Unification

and Resolution. Unification is finding substitutions of terms for variables to make two or more expressions identical. Resolution is a refutation proof procedure. This means that when a sentence has to be proved add the negative of that sentence to the knowledgebase. Then use the inference rule that 'if a sentence and a negative of the sentence are found, reduce it to an empty sentence'. If such an empty sentence is derived then we have proved our statement. We can use Resolution and Unification for answering our queries. For example:

Knowledgebase:

P(Tim)

P(Sarah)

Query:

P(x)

Proof:

$\neg$P(X)

$\neg$P(Tim) $\Rightarrow \cdot$ (modus tollens)

$\neg$P(Sarah) $\Rightarrow \cdot$ (modus tollens)

Solution:

X – Tim, Sarah

The knowledgebase has the information that some predicate 'P' is true for the values 'Tim' and 'Sarah'. The query is to find all values of 'x' that make the predicate 'P' true. The proof begins

with ‘Resolution’ by adding the ‘NOT’ before the query. Then it uses ‘Unification’ and substitutes the values ‘Tim’ and ‘Sarah’ for ‘x’. By the inference rule ‘Modus Tollens’ if a statement and its negation exist, then it can be reduced to an empty set. The values of ‘x’ that lead to an empty set are the solutions to our query, in this case ‘Tim’ and ‘Sarah’.

In our system, the query will be conflict(x,y) and the result will be the solutions for x and y where x and y will be the ids for the RDF statements that are in conflict. Thus our solution will be the result of such a logical query. An observation that needs to be made at this point is that the predicate logic statement $\forall X(man(X) \Rightarrow mortal(X))$ is in fact a rule from which we make the conclusion that if man(Socrates) then mortal(Socrates). A rule, in predicate logic is of the form

If p1, p2, p3….

Then q1, q2, q3…

where the ps and qs are sentences. The ps are called the premises and the qs are called the consequents. If the number of consequent is reduced to one then it is called Horn rule/logic. Rules are used for the following reasons and are classified accordingly as shown in the table below.

Derivation/Production Rules	To derive implicit facts from explicit facts.
Integrity Constraint Rules	To check consistency.
Reaction(Action Rules)	To take an action when certain conditions are met.
Facts (Rules without premises)	To state actual information.

In our work we use two types of rules which are both Horn type rules. They are Conflict rules and Simplification rules. A short primer on RuleML will help at this point because we make use of it as the means to specify rules.

4.2 RuleML – A Brief Introduction

The following illustration (Figure 6) from the RuleML home page (http://www.ruleml.org/) illustrates how rules are represented in RuleML [12]. This theoretic discussion behind the design rationale [12] of RuleML is beyond the scope of this work.

Sample rule used in Figure 6: A person owns an object if that person buys the object from a merchant and the person keeps the object.

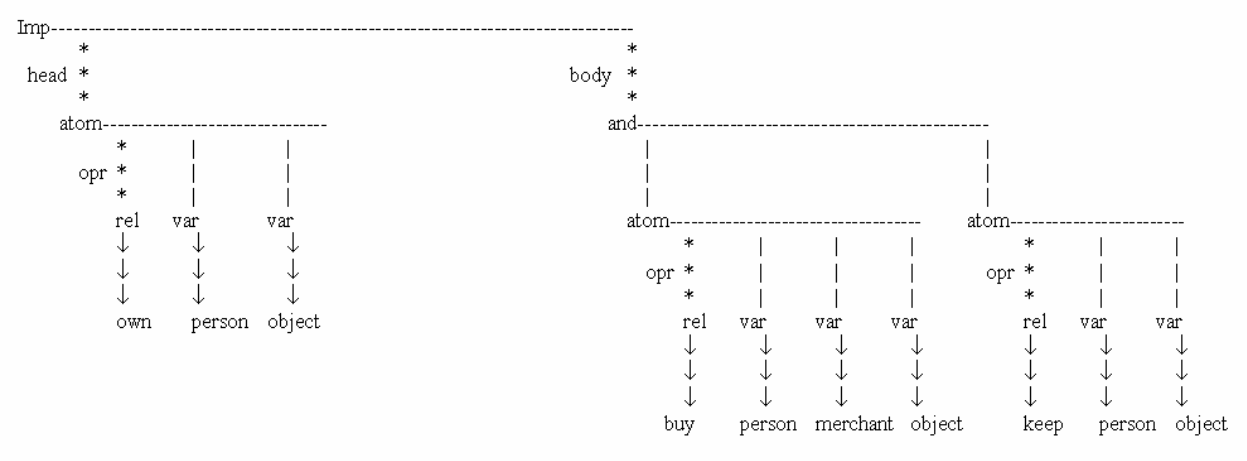


Figure 6: RuleML Order Labeled Tree

RuleML uses XML syntax to represent this rule:

```
<imp>
  <_head>
    <atom>
      <_opr>
         <rel>own</rel>
      </_opr>
      <var>person</var>
      <var>object</var>
    </atom>
  </_head>
  <_body>
    <!-- explicit 'and' -->
    <and>
      <atom>
       <_opr>
             <rel>buy</rel>
        </_opr>
        <var>person</var>
        <var>merchant</var>
        <var>object</var>
      </atom>
      <atom>
         <_opr>
               <rel>keep</rel>
          </_opr>
          <var>person</var>
           <var>object</var>
        </atom>
    </and>
  </_body>
</imp>
```

An implication “imp” represents the rule. It consists of a “head” , the “then” part and the “body”, which contains the “if condition (s)”. The “atom” in the “head” represents the result of the rule. The “atom” (s) in the body represent the conditions that need to be satisfied for the head to be true. This is the extent to which we have used RuleML in our work as will be illustrated with examples in the following sections.

4.3 Conflict Rules

This section deals with use of rules in identifying conflicts given a collection of semantic meta-data based on the classification of conflicts defined in Chapter 3. The basic idea is to convert the assertions to the form of rules and to signal a conflict if these rules are violated. The RuleML [12] initiative addresses the design issue of rule markup for the semantic Web. We use RuleML as an intermediate step in identifying conflicts, and translate assertions in RDF(S), DAML+OIL, or OWL to RuleML rules. We express assertions on statements that cannot be expressed in RDF(S), DAML+OIL or OWL which is discussed in previous section as rules. For simplicity we show the rule in a intuitive If-Then format below and then the RuleML format later:

if statement(x) and statement(y) and
subject(x,a) and relation(x,*rel1*) and object(x,b) and
subject(y,a) and relation(y,*rel2*) and object(y,b) and disjoint(*rel1*,*rel2*)
then conflict(x,y);

Here ‘statement’, ‘subject’, ‘relation’, ‘object’ and ‘disjoint’ are the prerequisites and ‘conflict’ is the conclusion. The rule above indicates that two statements x and y are in conflict if the subject *a,* and object *b* they address are identical and the relations (i.e., properties) *rel1*, and *rel2* are defined as *disjoint*. The x and y stand for identification for the statements. Reifying the RDF

statements into triples and assigning an id to each triple can achieve this. An important observation at this point is that when two triples are in conflict, we syntactically represent the conflict as a statement about two statements. This is not directly supported in semantic Web languages currently according to our knowledge. We overcome this limitation by making the term ‘statement’ into a predicate and programmatically assigning an id to each RDF statement. For example a statement like ‘statement (HarryPotter *type* Book)’ is expressed as

Statement(*x*)

Subject(*x*, HarryPotter)

Property(*x*, type)

Object(*x*, Book)

where *x* is an id generated during reification. The same rule expressed in RuleML would be:

```
<?xml version="1.0"
encoding="UTF8" ?>
   <rulebase>
  <imp>
    <_head>
     <atom>
      <_opr>
<rel>conflict</rel>
      </_opr>
      <var>x</var>
      <var>y</var>
     </atom>
    </_head>
  <_body>
   <and>
     <atom>
      <_opr>
 <rel>statement</rel>
      </_opr>
      <var>x</var>
     </atom>
     <atom>
      <_opr>
 <rel>subject</rel>
      </_opr>
      <var>x</var>
      <var>a</var>
     </atom>
     <atom>
      <_opr>
<rel>relation</rel>
      </_opr>
      <var>x</var>
      <var>rel1</var>
     </atom>
     <atom>
      <_opr>
<rel>object</rel>
      </_opr>
      <var>x</var>
      <var>b</var>
     </atom>
     <atom>
      <_opr>
<rel>statement</rel>
      </_opr>
      <var>y</var>
     </atom>
     <atom>
      <_opr>
<rel>subject</rel>
      </_opr>
      <var>y</var>
      <var>a</var>
     </atom>
     <atom>
      <_opr>
<rel>relation</rel>
      </_opr>
      <var>y</var>
      <var>rel2</var>
     </atom>
     <atom>
      <_opr>
<rel>object</rel>
      </_opr>
      <var>y</var>
      <var>b</var>
     </atom>
     <atom>
      <_opr>
<rel>disjoint</rel>
      </_opr>
      <var>rel1</var>
      <var>rel2</var>
     </atom>
   </and>
  </_body>
 </imp>
</rulebase>
```

4.4 Simplification Rules

A rule-based specification is also used for simplification of complex relations:

if statement(*x*) and statement(*y*) and

subject(*x*,*a*) and relation(*x*,*rel1*) and object(*x*,*b*) and

subject(*y*,*a*) and relation(*y*,*rel2*) and object(*y*,*b*)

then newStatement(a,rel3,b)

This rule means that "if statements *x* and *y* have relations *rel*1 and *rel*2 between subject *a*, and object *b*", then we can add statement "*a rel*3 *b*" to the knowledgebase. This rule indicates *rel*1

and *rel*2 can be composed to *rel*3. A composition rule of this type could be based on expert knowledge. Furthermore, the relations *rel*1 and *rel*2 may not be composable relations, where each successor relationships are connected through the same nodes; yet, a totally new relationship can be established. An example is the addition of relation "*dedicatedTo*" in Figure 4. This simplification rule expressed as RuleML would be

```
<?xml version="1.0"
encoding="UTF8" ?>
 <rulebase>
  <imp>
   <_head>
    <atom>
     <_opr>
 <rel>newStatement</rel>
     </_opr>
     <var>a</var>
     <var>rel3</var>
     <var>b</var>
    </atom>
   </_head>
   <_body>
    <and>
     <atom>
      <_opr>
 <rel>statement</rel>
      </_opr>
      <var>x</var>
     </atom>
      <atom>
       <_opr>
      <rel>subject</rel>
       </_opr>
       <var>x</var>
       <var>a</var>
      </atom>
      <atom>
       <_opr>
      <rel>relation</rel>
       </_opr>
       <var>x</var>
       <var>rel1</var>
      </atom>
      <atom>
       <_opr>
      <rel>object</rel>
       </_opr>
       <var>x</var>
       <var>b</var>
      </atom>
      <atom>
       <_opr>
      <rel>statement</rel>
       </_opr>
       <var>y</var>
      </atom>
       <atom>
        <_opr>
       <rel>subject</rel>
        </_opr>
        <var>y</var>
        <var>a</var>
       </atom>
       <atom>
        <_opr>
       <rel>relation</rel>
        </_opr>
        <var>y</var>
        <var>rel2</var>
       </atom>
       <atom>
        <_opr>
       <rel>object</rel>
        </_opr>
        <var>y</var>
        <var>b</var>
       </atom>
    </and>
   </_body>
  </imp>
 </rulebase>
```

Here again we wish to stress the fact that our idea of using simplification can be extended to include all the inference allowed under the chosen format RDF(S)/DAML+OIL/OWL. We have considered simplification only for simplification when it can actually be used for generating new

knowledge based on existing knowledge. Note that more advanced forms of simplification can be achieved by simplifying arbitrary templates of complex relations. Yet this type of simplification is out of scope of this thesis. Simplification is further discussed in the section on Relationship Ontology. To be precise with terminology, we can say that a rule is a simplification rule if the ‘consequent’ or result of the rule is a new statement and a rule is a conflict rule if the ‘consequent’ or result of the rule is a conflict decision.

CHAPTER 5

RELATIONSHIP ONTOLOGY

We saw in the previous section how our work is influenced by the concepts used in logic. Here we would like to show how the concept of Relationship Ontology is inspired by results of Frame Logic and Description Logic.

5.1 Background

Frame Logic [35] is similar to the object-oriented paradigm. The information in the knowledgebase is grouped around objects as opposed to being grouped around relations as in predicate logic. We will give a brief description of the way information is organized in Frame Logic. The information is organized in three layers or levels, the Object Base, the Database Facts and General Class information as follows:

The Object Base (classes, subclasses and objects):

```
Empl::person              (Empl is subclassOf  person)
Student::person
Faculty::empl
Child(person)::person     ( Child(person) a class defined as function without a name      )
John:student              (John is of type student)
John:empl
```

Note that there is no common root “class” that every other class derives from. Classes are also objects and can be instances of other classes. Note the use of ‘Child(person)’. It stands for a class without a name, predecessor of the ‘anonymous classes’ in the semantic Web languages.

Database Facts:

```
Bob [   name → ”Bob”;
        age → 40;
        affiliation → cs1[     dname → “CS”;
        mngr → bob;
        assistants --» { john, sally } ] ]
```

These are the actual instances of data. Here Bob is an object with name(“Bob”), age(40) and affiliation(cs1).

General Class Information:

```
Faculty [       boss ⇒ (faculty, manager ) ;
                papers =» article;                ]
```

This is the place where you define what properties a class can have and what values the properties can have. This is similar to ‘class restrictions’ and ‘property restrictions’ in semantic Web languages. We can also have rules and queries in Frame Logic which will not be relevant to the discussion at hand. Frame ontology introduces the concept of classes, objects and properties. Description Logic [36] uses these concepts but makes a clear distinction between the layer for class description (TBox or terminology box) and the layer for actual instance data (ABox or Assertion box) (Figure 7).

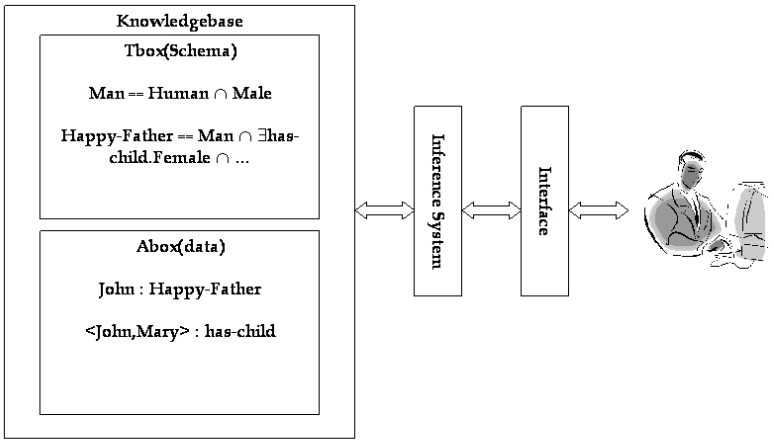


Figure 7: Description Logic Based System

This separation helps in making a distinction between reasoning for the TBox (Satisfiability, Subsumption, Equivalence, and Disjointness) reasoning for the ABox (Consistency). Description of each of the reasoning tasks is out of scope of this work. We would like to point out that our view of conflict checking falls into 'Consistency' checking on the ABox. Our idea of Relationship Ontology is equivalent to splitting the TBox into two, one for classes and another for relations (like 'has-child' in Figure 7) and do reasoning on the relations box as well. Now we will discuss the Relationship Ontology in connection to the semantic Web for which it is intended.

5.2 The Concept Of Relationship Ontology

There is an emerging consensus that the relation between entities as well as the nature of these relations are at the heart of the semantic Web [48]. Our introduction of this terminology 'Relationship Ontology' and concept is a step forward in that direction.

Conflict identification by simplification and rules has made extensive use of relations about relations, e.g., *disjoint, unique, etc*. Thereby, we envision a framework that allows an explicit definition of relations about relations through Relationship Ontology (RO). As contrary to traditional ontologies where entities and concepts are treated as first-class objects and relations as second-class the relations are treated as first-class objects in the RO. Hence, inheritance, similarity, part-of and other relations among relations can be specified by domain experts. For example, composable relations as exemplified in Figure 3 find their place in this ontology.

Ontologies primarily have two layers, the schema (vocabulary) layer and the instances (assertion) layer. The schema layer can be further divided into ontology meta-layer and application specific schema [16]. The latter consists of the application specific classes and

properties (relations). The relations about relations are placed in this layer, and we call it the Relations Schema. Figure 8 illustrates an ontology that makes use of RDF(S) to define relations about relations.

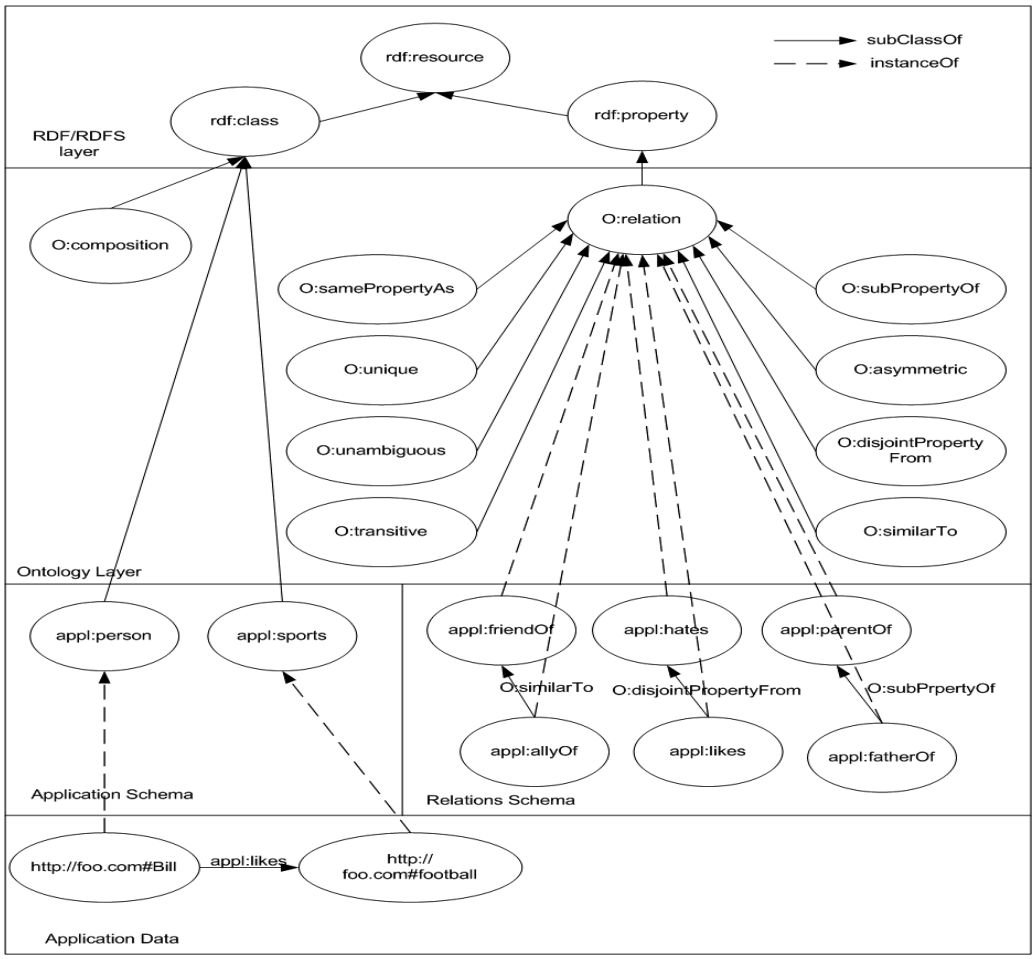


Figure 8: An Object Model Highlighting the Relations Schema

The Relations Schema layer can be a different ontology that can evolve separately for specific applications or domains. If it is specified as a separate ontology we name the Relations Schema layer as RO.

For the RO we use the property constraints from RDFS, DAML and Frame Ontology [25] and some additional constraints when they are not supported by these frameworks. Thus, the constraints in the RO include:

- daml:samePropertyAs, owl:equivalent-Property
- daml:unique, owl:functionalProperty
- daml:unambiguous, owl:inverseFunctionalProperty
- daml:transitive, owl:transitive-Property
- rdfs:subPropertyOf,
- Composition [16],
- Asymmetric [16],
- disjointPropertyFrom,
- similarTo.

The relation 'disjointPropertyFrom' is intended to express the fact that two relations cannot be true at the same time when both relations have the same subject and object, e.g. "likes *disjointPropertyFrom* hates". 'similarTo' relation is intended to specify that two relations that are not defined as equivalent (i.e., 'samePropertyAs') are in fact considered similar, e.g. "allyOf *similarTo* friendOf".

Other constraints or axioms under one of the major categories [16] such as relational algebra, (exhaustive) partitions, axioms for sub-relations, part-whole reasoning, etc. have not been included in RO. We have initially incorporated axioms that we consider important for simplification, and conflict identification.

CHAPTER 6

SYSTEM ARCHITECTURE AND EXPERIMENTS

The idea of semantic conflict detection is tested by implementing a prototypical system. In the prototype, conflict identification involves the following steps:

1) Meta-data is extracted from source documents into RDF documents.
2) These documents are serialized, and triples (i.e., RDF data) are placed in the knowledgebase.
3) By relying on the RO, the implicit simplifications are enumerated and added as triples into the knowledgebase.
4) The assertions/constraints are translated from the (user) ontology into rules which are placed in the rule-base.
5) A rule engine identifies conflicting statements by querying the knowledgebase.

6.1 System Architecture

The system architecture consists of three major components (see Figure 9). The *simplification* module uses information from the (user-provided) input ontology, RO and the rule-base to simplify the statements available to the desired granularity level. The simplification rules are based on expert knowledge and added through a user interface or by adding RuleML files. The output of the *simplification* module is stored as facts. The *conflict identification* module analyzes the constraints available in ontology provided by the user as well as the RO in order to generate conflict identifying rules.

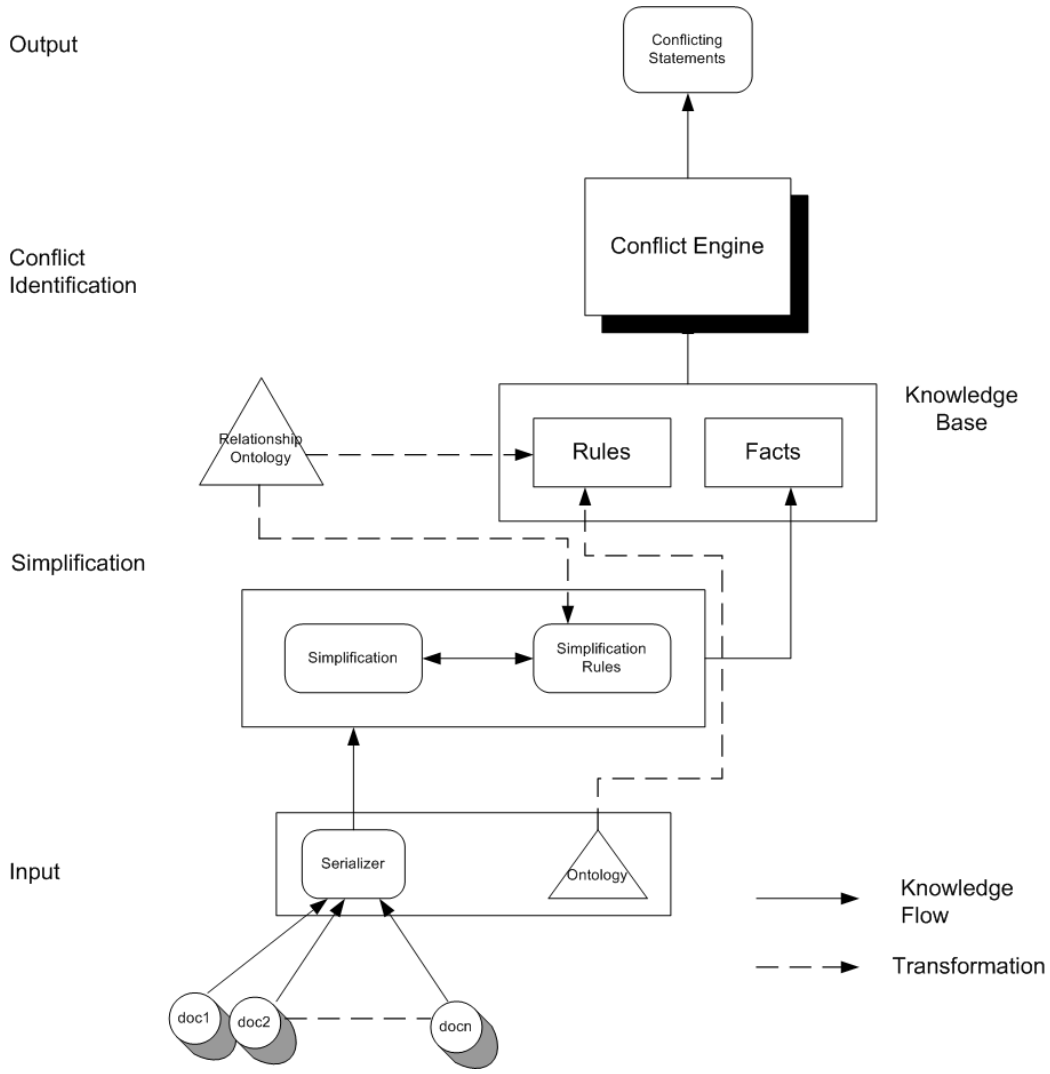


Figure 9: Overview of System Architecture

We use pre-defined rule templates to achieve this translation. Whenever there is a constraint on a relation the corresponding template is invoked and populated with the relation. This is then added to the rule-base. For example, we have a pre-defined template for *disjoint* as follows:

if statement(x) and statement(y) and

subject(x,a) and relation(x,*rel1*) and object(x,b) and

subject(y,a) and relation(y,*rel2*) and object(y,b) and

disjoint(*rel1*,*rel2*) then conflict(x,y)

When there is a constraint *disjoint(http://foo.com/test#likes, http://foo.com/test#hates)* the values of *rel1* and *rel2* are replaced with '*http://foo.com/test#likes*' and '*http://foo.com/test#hates*' respectively, and the rule that is placed in the rule-base will be:

if statement(x) and statement(y) and

subject(x,a) and relation(x, http://foo.com/test#likes) and object(x,b) and

subject(y,a) and relation(y, http://foo.com/test#hates) and object(y,b) and

disjoint(http://foo.com/test#likes, http://foo.com/test#hates) then conflict(x,y).

A full list of the constraints and their RuleML representations are given in Appendix A.

The *conflict engine* uses rules and facts to identify the conflicts and generates a list of the conflicting pairs of statements as output. In fact, the conflicts can have a degree and be ranked accordingly. We have not studied this issue in this thesis work. The screenshots (Figures 10 and 11) of the GUI illustrate how the knowledgebase can be queried for conflicts. We have used Semantic Web Technology Evaluation Ontology(SWETO), Version 2.0 [26] developed at the LSDIS lab at the University of Georgia and in these examples.

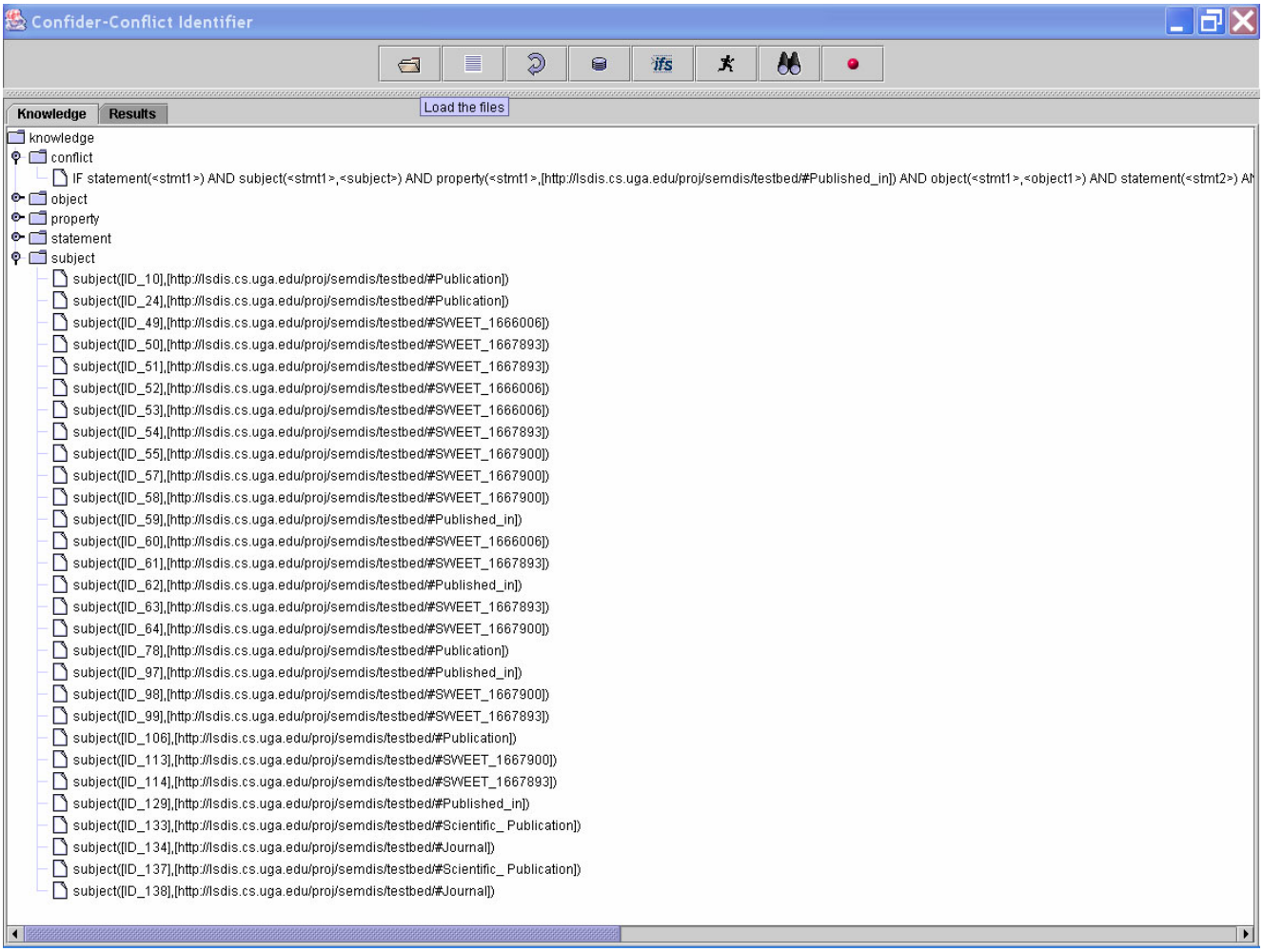


Figure 10: Knowledgebase with Facts and Rules

Figure 10 illustrates some example statements with subject, property (predicate) and object. For example the statement with id 'ID_49' has subject

'http://lsdis.cs.uga.edu/proj/semdis/testbed/#SWEET_1666006'

The conflict node in figure 10 contains the rules that will help identify conflicts. For example we define the property

'http://lsdis.cs.uga.edu/proj/semdis/testbed/#Published_In'

as unique in the ontology and the rule in the conflict node represents this constraint. This specifies a paper cannot be in more than one journal or conference.

The Relationship Ontology (RO) can be edited prior to running the queries for conflict checking. This is done by associating properties of the ontology provided by the user with the relations defined already in the RO. For example, to define a relation as unique in the RO, the following information can be entered "(unique, x)" where x is the relation and 'unique' is already present in the RO. Similarly, in order to define two relations as "disjoint" the information "(disjoint, x, y)" can be entered, where x and y are the relations and 'disjoint' is already part of the RO. The relations entered must belong to the same namespace as that of the input documents for the RO to have an effect on the conflict detection results. Note that these rules about relation will be mapped to RuleML eventually.

The "conflict" query is evaluated using a backward-reasoning algorithm implemented using the Mandarax API [27, 31]. Mandarax is an open source java class library for deduction rules. It provides an infrastructure for defining, managing and querying rule bases. Mandarax is pure OO [31], not a translation of a prolog interpreter from c to java. The design is flexible and open, making use of well-known design patterns such as factory, adapter, singleton, strategy and others. Mandarax is based on backward reasoning. This fits perfectly in a computing landscape based on a pull model (e.g., a transaction initiated from a web site). Mandarax includes a comprehensive library of pre-defined predicates and functions. Mandarax contains a reference implementation of an inference engine. This engine is very flexible: unification algorithm, loop checking algorithm and selection policy can be configured. Oryx [14], an extension containing a visual editor, a JSP tag lib to deploy applications and a catalog like meta-data concept is available.

The result of executing the query returns the ids of statement pairs those are in conflict (51, 55 in Figure 11). The right frame in Figure 11 shows the components of the concerned

statements and the particular conflict rule that caused these two statements to be in conflict. This tree is called the derivation for this result. This allows providing an answer of "why" a pair of statements is in conflict. For example the statements with ids 'ID_51' and 'ID_55' are in conflict because they have the same subject

'http://lsdis.cs.uga.edu/proj/semdis/testbed/#SWEET_1667893'

connected to different objects,

'http://lsdis.cs.uga.edu/proj/semdis/testbed/#SWEET_1666006'

and

'http://lsdis.cs.uga.edu/proj/semdis/testbed/#SWEET_1666007'

through the same property

'http://lsdis.cs.uga.edu/proj/semdis/testbed/#Published_In'

which has a conflict rule associated with it. 'SWEET_1667893' is the resource Id of a publication. 'SWEET_1666006' and 'SWEET_1666007' are the resource ids of journals.

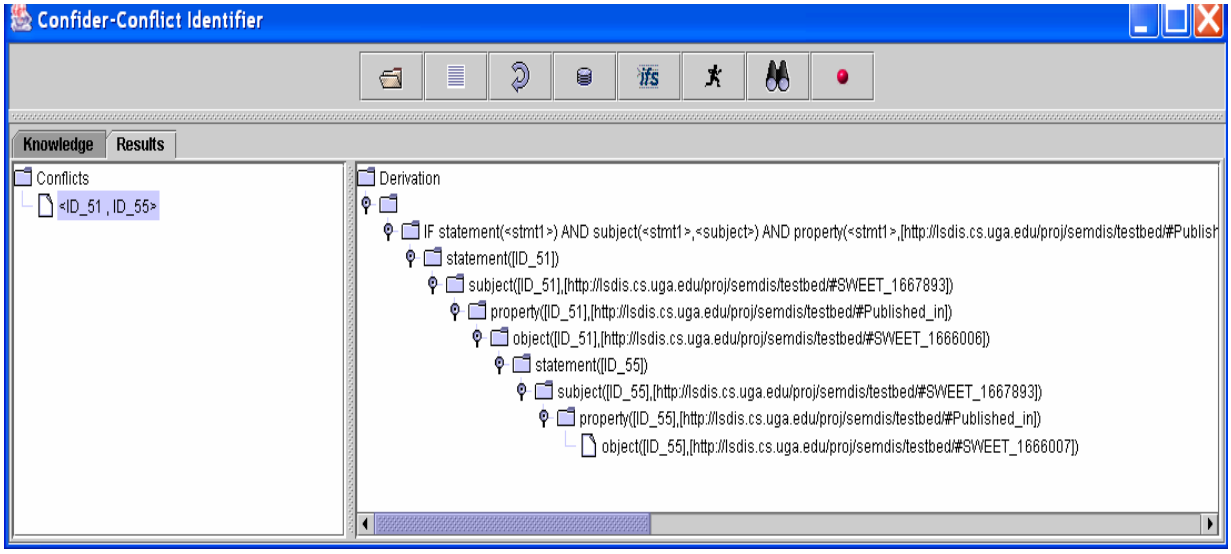


Figure 11: Execution of a Conflict Query on the Knowledgebase

As discussed earlier, a simplification is needed in some cases. The simplification process is executed as a query and the resulting statements are stored back in the knowledgebase. The derivation is stored along with the simplification. The derivation in this case would be the simplification rule used together with the statements that were brought together into this simplification. Thus when there is a conflict involving a simplification, it is possible to provide detail of how that simplification was achieved using the associated derivation.

6.2 System Performance

The following experiments were done to evaluate the performance of the system. In the first case the number of triples was kept constant and the number of conflicts was increased.

The graph (figure 12) shows the time in the y axis and the number of conflicts in the x axis. The time increases linearly with increase in the number of conflicts. This suggests that with a fixed number of triples, the system is scalable with respect to the number of conflicts found within those triples.

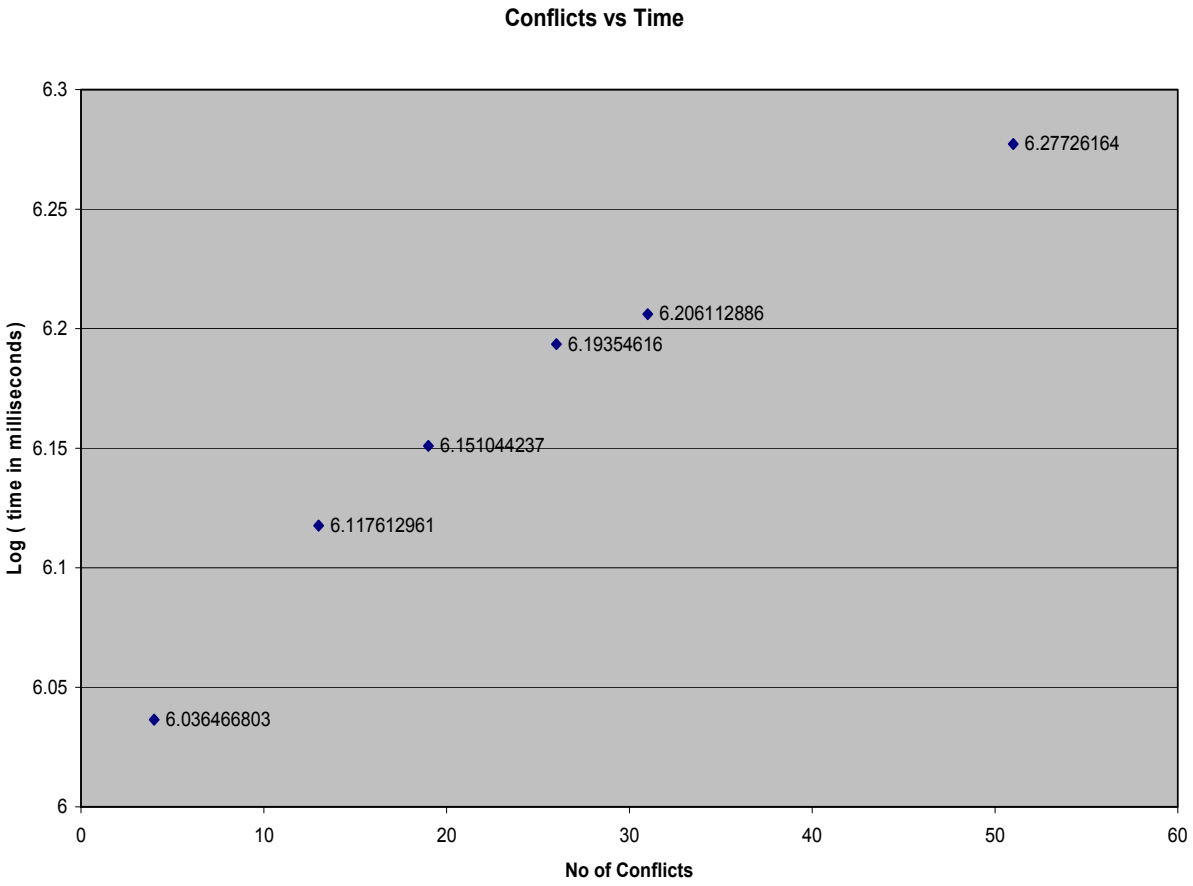


Figure 12: Performance with increase in number of conflicts (500 triples)

In the second case the number of conflicts was kept constant and the number of triples was increased. The graph (Figure 13) shows the time in the y axis and the number of triples in the x axis. The time required increases linearly but the rate of increase goes down as the number of triples increases. This can be explained based on the way the inference engine of Mandarax [49] has been implemented. The engine has a parameter 'maxsteps' that specifies the maximum number of derivation steps that it should perform before it gives up. The value of this parameter determines the depth of the tree that the rule engine uses. When the number of triples is increased the time taken to construct this tree increases. There is a threshold where the tree is saturated and so the time taken to detect the conflicts almost becomes a constant. When the number of triples is more than what the tree can handle the inference engine is not able to detect the conflicts.

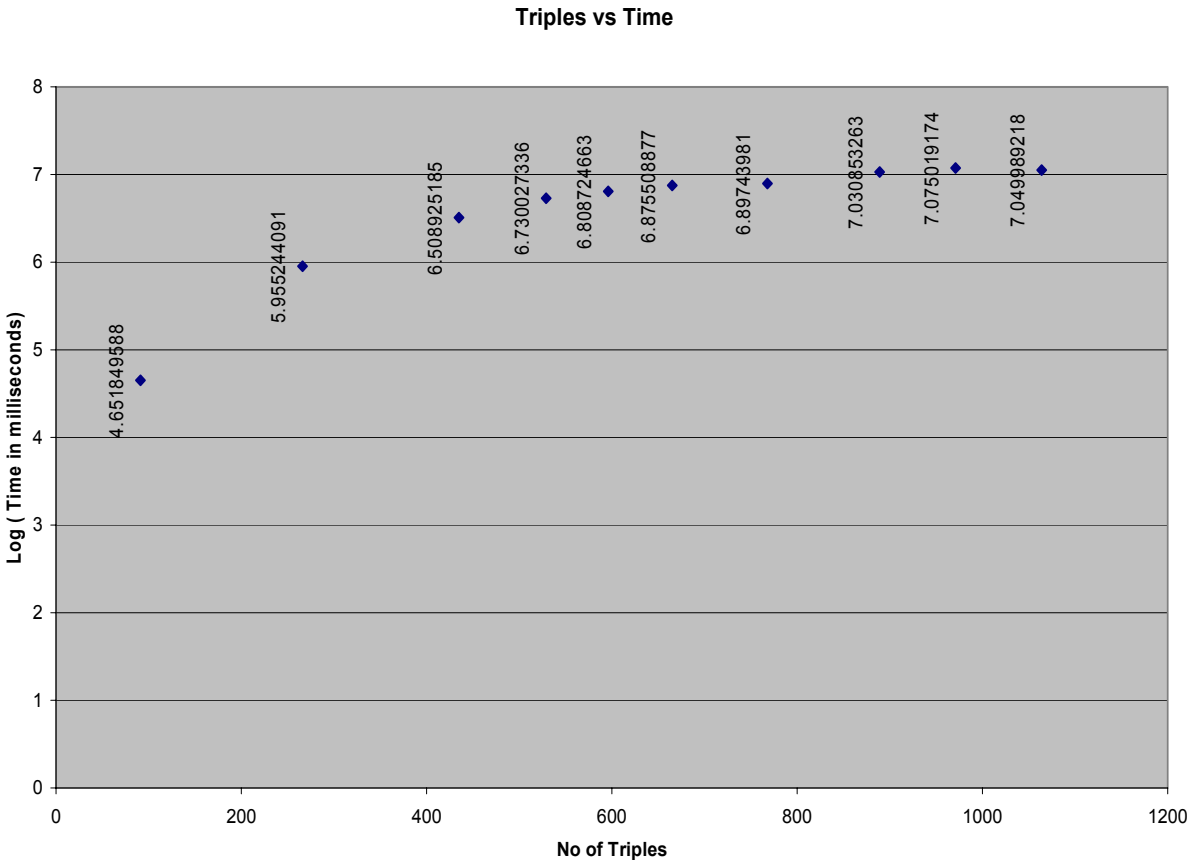


Figure 13: Performance with increase in number of triples (10 conflicts)

For the case of multiple rules, each of them is evaluated individually and the results are accumulated. With a large set of facts and a relatively limited number of rules this methodology will be efficient. However, when the number of rules increases there will be scalability issues because each rule has to be evaluated over the entire set of facts. We would like to further address the scalability issue in a future work.

When a triple is represented as a fact we use four predicates (statement, subject, property, and object). This increases the amount of memory required to hold the knowledgebase in memory. This may also limit the scalability of the approach. One design choice that could have helped is to represent the triple as a single predicate ‘triple (id, subject, property, object)’. We

made a choice to use binary predicates (predicates with two parameters) which resemble triples closely.

CHAPTER 7

RELATED WORK

Work in the field of electronic commerce has lead to several important ideas about conflicts. They deal with the priority of one business rule over other in the case that both are applicable. They are more concerned with resolving the conflicts than identifying it. An example would be the prioritized conflict handling from IBM[17]. They introduce a term called *overrides* to indicate which rule has priority over the other. Furthermore, trust management is discussed comprehensively in the context of semantic Web in [28, 29]. Our work can be used to verify trustworthiness of semantic meta-data by checking if there are contradictions (i.e. , conflicts) available in the data.

Rules have been employed to define the behavior of agents [27]. The agents exploit the rules described on semantic information, and they keep themselves updated by running queries based on those rules on the underlying data that changes periodically. The method we propose can help such an agent to discern conflicting information.

It must be made clear that conflict discussed in this work is distinct from the term ‘semantic conflict’ in some literature (e.g., [8]). In the literature semantic Conflict refers to the usage of the same term with different meanings resulting in ambiguity in understanding the information. For example one source may use the term ‘rate’ as charges after taxes and another source may use the term ‘rate’ as charges before taxes. In our discussion of conflicts we do not consider this kind of ambiguity oriented conflicts.

Considerable work has been done to categorize properties or relations into hierarchies [18] (e.g., taxonomies). They satisfy just one property of the Relationship Ontology, which is subPropertyOf. For our purposes, we need information about how one property is related to another. Also, an ontology is different from taxonomy in having named relations and not just a hierarchy. Similarly our Relationship Ontology is different from property hierarchies by having named relations between relations (properties).

The UMLS project [49] is a long-term NLM (National Library of Medicine) research and development effort designed to facilitate the retrieval and integration of information from multiple machine-readable biomedical information sources. The sources of interest include: descriptions of the biomedical literature, clinical records, factual databanks, knowledge-based systems, and directories of people and organizations. This project uses 'attributes' in a 'Metathesaurus' to add information about the relations. This is similar to our idea of Relationship Ontology.

From a conceptual point of view our approach is similar to TRIPLE [31], F-Logic [31] where the whole RDF model is re-represented with the RDF triples as the basic element. TRIPLE is also part of the RuleML initiative. These languages are designed for various types of inference. Our work is geared towards finding conflicts. Also these are efforts towards representing RDF in a way that logic such as Prolog can be used for evaluation [30]. Our work is more of an evaluation strategy than a representation technique.

Recent efforts like Semantic Web Rule Language (SWRL) [32] have tried to realize the logic layer of the Semantic Web by combining RuleML and OWL where rules of RuleML are written using vocabulary from OWL. Our work does not try to bridge the gap between rules and

Semantic Web Languages. Rather it is an effort in expressing OWL constraints as rules that can be evaluated using a rule engine.

The quality of an ontology can be evaluated based on the completeness of the schema and the trustworthiness and consistency of the data populated based on the ontology. Our work can help in maintaining the quality by identifying potential conflicts and manually resolving the conflicts.

CHAPTER 8

CONCLUSION AND FUTURE WORK

In this work, we have defined conflicts. We discussed the different types of conflicts and ways to identify them given a collection of semantic meta-data. We have also shown the use of RuleML rules to express conflicts and simplifications. We also presented a Relationship Ontology that can evolve independently and enable simplification of complex relations. With this, conflicts can be detected at different granularity level than the RDF statements. A system and corresponding API have been developed to check RDF(S)/DAML/OWL documents for conflicts. A prototype implementation demonstrates the use of this API and the encouraging performance of the approach.

Our future work directions include developing:

- Developing a more scalable conflict identification techniques for large amounts of semantic meta-data and conflict rules,
- Developing a ranking criterion for the conflicts,
- Investigating other rule evaluation methods to improve performance.
- Experimenting with ways of representing an RDF triple in predicate form to compare performance.
- Building a mechanism for expressing, evaluating, and adjusting trust dynamically based on conflict detection.

- Investigation of using a P2P network for identifying conflicts. A brief outline of our thought process in this direction follows.

A P2P system can help in reducing the complexity of conflict checking by delegating some steps to individual peers. This delegation becomes significant when individual peers are able to contribute some rules and expert knowledge into the conflict detection process. Thus a conflict that cannot be detected with the available information at any given peer will become more obvious as the information is processed at different peers. The following example highlights the significance of a P2P environment over centralized processing for conflict identification. Figure 14 illustrates how multiple peers in concert can help in identifying conflicts that is not obvious to a single peer.

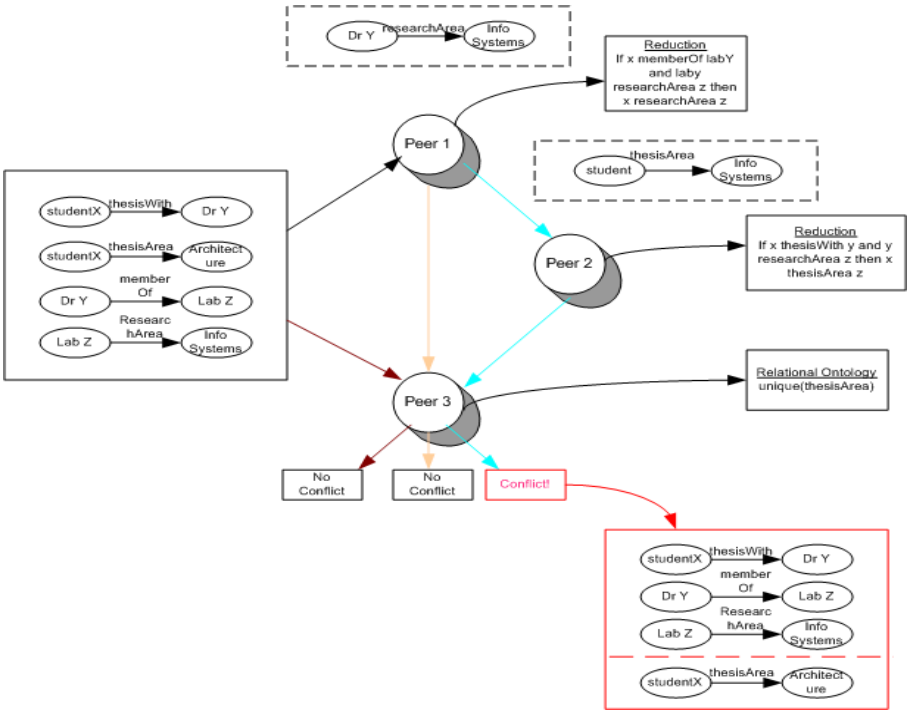


Figure 14: Semantic Conflict Identification in a Peer-to-Peer Network

Peer 3 makes the decision about conflict. It has information that the relation ‘thesisArea’ has to be unique. From the available pool of statements this information is not enough to identify the

conflict. Peer 1 has a simplification rule which uses the information that the relation ‘memberOf’ a lab has to have same ‘researchArea’ as the lab itself and derives a statement which is added to the knowledgebase (shown as dotted rectangle). Peer 2 has a simplification rule, and uses it to add a statement to the knowledgebase. A conflict cannot be detected yet. After these steps, when the pool of statements reaches Peer 3 it is able to identify the conflict from the statement that was derived at Peer 2. Note that if the processing at Peer 3 had happened before Peer 2 we would never have identified the conflict. Thus the challenge is to establish an interaction pattern between the peers for conflict identification.

REFERENCES

[1] Kemafor Anyanwu and Amit Sheth, The rho Operator: Enabling Querying for Semantic Associations on the Semantic Web. SIGMOD Record (Special issue on Amicalola Workshop), 31 (4), pp. 42-47, December 2002.

[2] Amit Sheth, Boanerges Aleman-Meza, I. Budak Arpinar, Clemens Bertram, Yashodhan Warke, Cartic Ramakrishnan, Chris Halaschek, Kemafor Anyanwu, David Avant, F. Sena Arpinar, and Krys Kochut, Semantic Association Identification and Knowledge Discovery for National Security Applications. Special Issue of Journal of Database Management on Database Technology for Enhancing National Security, Eds: L. Zhou and W. Kim, 2003.

[3] Boanerges Aleman-Meza, Chris Halaschek, I. Budak Arpinar, and Amit Sheth, Context-Aware Semantic Association Ranking. Proceedings of the First International Workshop on Semantic Web and Databases, Berlin, Germany, September 7-8, 2003; pp. 33-50.

[4] Mullai T. Shanmuhan, SEMANTA: An Ontology-Driven Semantic Link Analysis Framework. Master's Thesis, Computer Science Department, University of Georgia, 2003.

[5] Lynn Lampert. Can you Trust the Web? Evaluating What You Find. Adapted from UC Berkeley training guide. (http://library.csun.edu/llampert/MOD/TrustWeb.ppt.)

[6] Evaluating Information Found on the Internet. A thoughtful guide to evaluating Web and other Internet resources for scholarly purposes, from John Hopkins University Library. (http://www.library.jhu.edu/elp/useit/evaluate/).

[7] Yolanda Gil, and Varun Ratnakar, Trusting Information Sources One Citizen at a Time. In Proceedings of the First International Semantic Web Conference (ISWC), Sardinia, Italy, June 9-12, 2002.

[8] Hongjun Lu, Weiguo Fan, and Chen Hian Goh, Discovering and Reconciling Semantic Conflicts: A Data Mining Perspective. In the Proceedings of the 7th IFIP 2.6 Working Conference on Data Semantics (DS-7), Leysin, Switzerland, 1997.

[9] Dan Connolly, Frank van Harmelen, Ian Horrocks, Deborah McGuinness, Peter F. Patel-Schneider, and Lynn Andrea Stein, Annotated DAML+OIL Ontology Markup. W3C Note 18 December 2001.

[10] Dan Connolly, Frank van Harmelen, Ian Horrocks, Deborah McGuinness, Peter F. Patel-Schneider, Lynn Andrea Stein, Annotated DAML +OIL Ontology Markup, W3C Note, 18 December 2001 (http://www.w3.org/TR/daml+oil-walkthru/).

[11] S. Staab, and A. Maedche, Axioms are Objects too - Ontology Engineering beyond the Modeling of Concepts and Relations. Workshop on Ontologies and Problem-Solving Methods, ECAI, Berlin 2000 .

[12] Harold Boley, Said Tabet, and Gerd Wagner, Design Rationale of RuleML: A Markup Language for Semantic Web Rules. In Proceedings of SWWS'01, The first Semantic Web Working Symposium, Stanford University, California, USA, July 30 - August 1, 2001.

[13] Dan Brickley, and R.V.Guha. RDF Vocabulary Description Language 1.0: RDF Schema, W3C Working Draft, 10 October 2003.

[14] Jens B. Dietrich. JBDietrich Knowledge Management Software, (http://www.jbdietrich.com).

[15] Richard Fikes, and Deborah McGuinness. An Axiomatic Semantics for RDF, RDF-S, and DAML+OIL, W3C Note 18 December 2001.

[16] Steffen Staab, Michael Erdmann, Alexander Maedche, and Stefan Decker, An Extensible Approach for Modeling Ontologies in RDF(S). In Proceedings of ECDL 2000 Workshop on the Semantic Web, Lisbon, Portugal, 11-22, 2000.

[17] Benjamin N. Grosof, Courteous Logic Programs: Prioritized Conflict Handling for Rules. IBM Research Report RC 20836, Dec. 30, 1997, revised from May 8 1997.

[18] R. Guha, and Rob McCool, TAP-A Semantic Web platform. The Eleventh International World Wide Web Conference, Honolulu, Hawaii. May 2002.

[19] Leah Graham, Panagiotis, and Takis Metaxas, Of course it's true; I saw it on the internet: critical thinking in the internet era. In Communications of the ACM, Volume 46, Issue 5, May 2003.

[20] Kemafor Anyanwu, and Amit P. Sheth. rho-Queries: Enabling Querying for Semantic Associations on the Semantic Web. The Twelfth International World Wide Web Conference, Budapest, Hungary. May 2003.

[21] Jennifer Golbeck, James Hendler, and Bijan Parsia. The Trust Networks on the Semantic Web. The Twelfth International World Wide Web Conference, Budapest, Hungary. May 2003.

[22] Ora Lassila, and Ralph R. Swick. Resource Description Framework (RDF) Model and Syntax Specification. W3C Recommendation 22 February 1999.

[23] Dan Connolly, Frank van Harmelen, Ian Horrocks, Deborah L. McGuinness, Peter F. Patel-Schneider, and Lynn Andrea Stein. DAML+OIL Reference Description, W3C Note 18 December 2001.

[24] Peter F. Patel-Schneider, Patrick Hayes, and Ian Horrocks. OWL Web Ontology Language Semantics and Abstract Syntax. W3C Candidate Recommendation, 18 August 2003.

[25] François Gerbaux, and Tom Gruber. Theory Frame Ontology. http://www-ksl.stanford.edu/knowledge-sharing/ontologies/html/frame-ontology/.

[26] Semantic Web Technology Evaluation Ontology (SWETO), v2.0, 2004-02-13, http://lsdis.cs.uga.edu/Projects/SemDis/Sweto/testbed_v2_0.owl

[27] J. Dietrich, A. Kozlenkov, M. Schroeder, and G. Wagner, Rule-Based Agents for the Semantic Web, Preprint submitted to Elsevier Science, 8 May 2003.

[28] Jennifer Golbeck, James Hendler, and Bijan Parsia, Trust Networks on the Semantic Web, WWW2003, May 20-26, 2003, Budapest, Hungary.

[29] Tim Finin and Anupam Joshi, Agents, Trust, and Information Access on the Semantic Web, SIGMOD Record, Volume 31, Number 4, December 2002.

[30] Li Ding, Lina Zhou, and Tim Finin. Trust Based Knowledge Outsourcing for Semantic Web Agents, 2003 IEEE/WIC International Conference on Web Intelligence (WI 2003), October 2003, Beijing.

[31] Jens Dietrich. The Mandarax Manual, 2003 (http://mandarax.sourceforge.net/docs/mandarax.pdf)

[32] Ian Horrocks, Peter F. Patel-Schneider, Harold Boley, Said Tabet, Benjamin Grosof, and Mike Dean. SWRL: A Semantic Web Rule Language Combining OWL and RuleML Version 0.5 of 19 November 2003 (http://www.daml.org/2003/11/swrl/).

[33] R. Davis, H. Shrobe, and P. Szolovits. What is a Knowledge Representation? AI Magazine, 14(1):17-33, 1993.

[34] Ginsberg, M. Essentials of Artificial Intelligence, Morgan Kaufmann, 1993.

[35] Michael Kifer, Georg Lausen, and James Wu. Logical Foundations of Object-Oriented and Frame – Based languages, Journal of ACM, 1995.

[36] F. Baader, and W. Nutt. Basic Description Logics. In the Description Logic Handbook, edited by F. Baader, D. Calvanese, D.L. McGuinness, D. Nardi, P.F. Patel-Schneider, Cambridge University Press, 2002, pages 47-100.

[37] Michael Sintek, and Stefan Decker. TRIPLE-A Query, Inference, and Transformation Language for the Semantic Web. International Semantic Web Conference (ISWC), Sardinia, June 2002.

[38] Boanerges Aleman-Meza, Chris Halaschek, Amit Sheth, I. Budak Arpinar, and Gowtham Sannapareddy. SWETO: Large-Scale Semantic Web Test-bed, International Workshop on Ontology in Action, Banff, Canada, June 20-24, 2004 (submitted)

[39] Chris Halaschek, Boanerges Aleman-Meza, I. Budak Arpinar, Amit Sheth. Discovering and Ranking Semantic Associations over a Large RDF Metabase, 30th Int. Conf. on Very Large Data Bases, August 30 - September 03, 2004, Toronto, Canada. Demonstration Paper (submitted).

[40] OWL Web Ontology Language Use Cases and Requirements, W3C Recommendation 10 February 2004.

[41] S.Staab: Emergent Semantics. IEEE Intelligent Systems 17(1), 2002, pp. 78-86

[42] V. Kashup and C. Behrens. The Emergent Semantic Web: A Consensus Approach for Deriving Semantic Knowledge on the Web, Proceedings of the International Semantic Web Working Symposium, July 2001, Stanford, USA.

[43] A. Sheth, C. Bertram, D. Avant, B. Hammond, K. Kochut, and Y. Warke. (2002). Managing semantic content for the Web. IEEE Internet Computing, 6(4), 2002. pp 80-87

[44] R. Mihalcea, and S. I. Mihalcea. Word Semantics for Information Retrieval: Moving One Step Closer to theSemantic Web. ICTAI 2001: 280-287.

[45] P. Resnik. Semantic Similarity in a Taxonomy: An Information-Based Measure and its Application to Problems of Ambiguity in Natural Language, Journal of Artificial Intelligence Research, 1999.

[46] V. Kashyap, and A. P. Sheth, Semantic and schematic similarities between database objects: A context –based approach. VLDB Journal, 5(4):276—304, 1996.

[47] M. Rodriguez, and M. Egenhofer. Determining Semantic Similarity among Entity Classes from Different Ontologies, IEEE Transactions on Knowledge and Data Engineering, Vol. 15, No. 2, March/April 2003.

[48] Relationships at the Heart of Semantic Web: Modeling, Discovering, and Exploiting Complex Semantic Relationships, A.Sheth, I. B. Arpinar, and Vipul Kashyap, Book Chapter in Enhancing the Power of the Internet: Studies in

Fuzziness and Soft Computing, M. Nikravesh, B. Azvin, R. Yager, and L. Zadeh, Eds., Springer-Verlag, 2003.

[49] National Library Of Medicine, Unified Medical Language System (UMLS), (http://www.nlm.nih.gov/research/umls/).

APPENDIX A

CONSTRAINTS AND RuleML REPRESENTATION

This section gives the representation of constraints that help in identifying conflicts in a 'IF-THEN' form and the RuleML form.

DISJOINT

IF-THEN FORM

if statement(*x*) and statement(*y*) and
subject(*x*,*a*) and relation(*x*,*rel1*) and object(*x*,*b*) and
subject(*y*,*a*) and relation(*y*,*rel2*) and object(*y*,*b*) and disjoint(*rel1*,*rel2*) then conflict(*x*,*y*);

RuleML FORM

```
<?xml version="1.0" encoding="UTF8" ?>
  <rulebase>
        <imp>
              <_head>
                    <atom>
                          <_opr>
                                <rel>conflict</rel>
                          </_opr>
                          <var>x</var>
                          <var>y</var>
                    </atom>
              </_head>
              <_body>
                    <and>
                          <atom>
                                <_opr>
                                      <rel>statement</rel>
                                </_opr>
                                <var>x</var>
                          </atom>
                          <atom>
                                <_opr>
```

```
                <rel>subject</rel>
        </_opr>
        <var>x</var>
        <var>a</var>
</atom>
<atom>
        <_opr>
                <rel>relation</rel>
        </_opr>
        <var>x</var>
        <var>rel1</var>
</atom>
<atom>
        <_opr>
                <rel>object</rel>
        </_opr>
        <var>x</var>
        <var>b</var>
</atom>
<atom>
        <_opr>
                <rel>statement</rel>
        </_opr>
        <var>y</var>
</atom>
<atom>
        <_opr>
                <rel>subject</rel>
        </_opr>
        <var>y</var>
        <var>a</var>
</atom>
<atom>
        <_opr>
                <rel>relation</rel>
        </_opr>
        <var>y</var>
        <var>rel2</var>
</atom>
<atom>
        <_opr>
                <rel>object</rel>
        </_opr>
        <var>y</var>
        <var>b</var>
</atom>
```

```
                        <atom>
                                <_opr>
                                        <rel>disjoint</rel>
                                </_opr>
                                <var>rel1</var>
                                <var>rel2</var>
                        </atom>
                </and>
        </_body>
</imp>
</rulebase>
```

UNIQUE/FUNCTIONAL

IF-THEN FORM

if statement(x) and statement(y) and
subject(x,a) and relation(x,rel1) and object(x,b) and
subject(y,a) and relation(y,rel1) and object(y,c) and notEqual(b,c) then conflict(x,y);

RuleML FORM

```
<?xml version="1.0" encoding="UTF8" ?>
        <rulebase>
                <imp>
                        <_head>
                                <atom>
                                        <_opr>
                                                <rel>conflict</rel>
                                        </_opr>
                                        <var>x</var>
                                        <var>y</var>
                                </atom>
                        </_head>
                        <_body>
                                <and>
                                        <atom>
                                                <_opr>
                                                        <rel>statement</rel>
                                                </_opr>
                                                <var>x</var>
                                        </atom>
                                        <atom>
                                                <_opr>
                                                        <rel>subject</rel>
                                                </_opr>
```

```
        <var>x</var>
        <var>a</var>
</atom>
<atom>
        <_opr>
                <rel>relation</rel>
        </_opr>
        <var>x</var>
        <var>rel1</var>
</atom>
<atom>
        <_opr>
                <rel>object</rel>
        </_opr>
        <var>x</var>
        <var>b</var>
</atom>
<atom>
        <_opr>
                <rel>statement</rel>
        </_opr>
        <var>y</var>
</atom>
<atom>
        <_opr>
                <rel>subject</rel>
        </_opr>
        <var>y</var>
        <var>a</var>
</atom>
<atom>
        <_opr>
                <rel>relation</rel>
        </_opr>
        <var>y</var>
        <var>rel1</var>
</atom>
<atom>
                <_opr>
                        <rel>object</rel>
                </_opr>
                <var>y</var>
                <var>c</var>
</atom>
<atom>
        <_opr>
```

```
                                        <rel>not equal</rel>
                                    </_opr>
                                    <var>b</var>
                                    <var>c</var>
                                </atom>
                            </and>
                        </_body>
                    </imp>
 </rulebase>
```

UNAMBIGUOUS/INVERSE-FUNCTIONAL

IF-THEN FORM

if statement(x) and statement(y) and
subject(x,a) and relation(x,rel1) and object(x,b) and
subject(y,c) and relation(y,rel1) and object(y,b) and notEqual(a,c) then conflict(x,y)

RuleML FORM

```
 <?xml version="1.0" encoding="UTF8" ?>
        <rulebase>
                <imp>
                        <_head>
                                <atom>
                                        <_opr>
                                                <rel>conflict</rel>
                                        </_opr>
                                        <var>x</var>
                                        <var>y</var>
                                </atom>
                        </_head>
                        <_body>
                                <and>
                                        <atom>
                                                <_opr>
                                                        <rel>statement</rel>
                                                </_opr>
                                                <var>x</var>
                                        </atom>
                                        <atom>
                                                <_opr>
                                                        <rel>subject</rel>
                                                </_opr>
                                                <var>x</var>
                                                <var>a</var>
```

```
</atom>
<atom>
        <_opr>
                <rel>relation</rel>
        </_opr>
        <var>x</var>
        <var>rel1</var>
</atom>
<atom>
        <_opr>
                <rel>object</rel>
        </_opr>
        <var>x</var>
        <var>b</var>
</atom>
<atom>
        <_opr>
                <rel>statement</rel>
        </_opr>
        <var>y</var>
</atom>
<atom>
        <_opr>
                <rel>subject</rel>
        </_opr>
        <var>y</var>
        <var>c</var>
</atom>
<atom>
        <_opr>
                <rel>relation</rel>
        </_opr>
        <var>y</var>
        <var>rel1</var>
</atom>
<atom>
                <_opr>
                        <rel>object</rel>
                </_opr>
                <var>y</var>
                <var>b</var>
</atom>
<atom>
        <_opr>
                <rel>not equal</rel>
        </_opr>
```

```
                                        <var>a</var>
                                        <var>c</var>
                                    </atom>
                                </and>
                            </_body>
                        </imp>
 </rulebase>
```

ASYMMETRIC

IF-THEN FORM

if statement(x) and statement(y) and
subject(x,a) and relation(x,rel1) and object(x,b) and
subject(y,b) and relation(y,rel1) and object(y,a) then conflict(x,y);

RuleML FORM

```
<?xml version="1.0" encoding="UTF8" ?>
<rulebase>
    <imp>
        <_head>
            <atom>
                <_opr>
                    <rel>conflict</rel>
                </_opr>
                <var>x</var>
                <var>y</var>
            </atom>
        </_head>
        <_body>
            <and>
                <atom>
                    <_opr>
                        <rel>statement</rel>
                    </_opr>
                    <var>x</var>
                </atom>
                <atom>
                    <_opr>
                        <rel>subject</rel>
                    </_opr>
                    <var>x</var>
                    <var>a</var>
                </atom>
                <atom>
```

```
                                        <_opr>
                                                <rel>relation</rel>
                                        </_opr>
                                        <var>x</var>
                                        <var>rel1</var>
                                </atom>
                                <atom>
                                        <_opr>
                                                <rel>object</rel>
                                        </_opr>
                                        <var>x</var>
                                        <var>b</var>
                                </atom>
                                <atom>
                                        <_opr>
                                                <rel>statement</rel>
                                        </_opr>
                                        <var>y</var>
                                </atom>
                                <atom>
                                        <_opr>
                                                <rel>subject</rel>
                                        </_opr>
                                        <var>y</var>
                                        <var>b</var>
                                </atom>
                                <atom>
                                        <_opr>
                                                <rel>relation</rel>
                                        </_opr>
                                        <var>y</var>
                                        <var>rel1</var>
                                </atom>
                                <atom>
                                                <_opr>
                                                        <rel>object</rel>
                                                </_opr>
                                                <var>y</var>
                                                <var>a</var>
                                </atom>
                        </and>
                </_body>
        </imp>
</rulebase>
```

COMPOSITION OF RELATIONS

IF-THEN FORM

if statement(x) and statement(y) and
subject(x,a) and relation(x,rel1) and object(x,b) and
subject(y,b) and relation(y,rel2) and object(y,c) then newStatement(a,rel3,c);

RuleML FORM

```
<?xml version="1.0" encoding="UTF8" ?>
<rulebase>
	<imp>
		<_head>
			<atom>
				<_opr>
					<rel>newStatement</rel>
				</_opr>
				<var>a</var>
				<var>rel3</var>
				<var>c</var>
			</atom>
		</_head>
		<_body>
			<and>
				<atom>
					<_opr>
						<rel>statement</rel>
					</_opr>
					<var>x</var>
				</atom>
				<atom>
					<_opr>
						<rel>subject</rel>
					</_opr>
					<var>x</var>
					<var>a</var>
				</atom>
				<atom>
					<_opr>
						<rel>relation</rel>
					</_opr>
					<var>x</var>
					<var>rel1</var>
				</atom>
				<atom>
					<_opr>
```

```
                                        <rel>object</rel>
                                    </_opr>
                                    <var>x</var>
                                    <var>b</var>
                                </atom>
                                <atom>
                                    <_opr>
                                        <rel>statement</rel>
                                    </_opr>
                                    <var>y</var>
                                </atom>
                                <atom>
                                    <_opr>
                                        <rel>subject</rel>
                                    </_opr>
                                    <var>y</var>
                                    <var>b</var>
                                </atom>
                                <atom>
                                    <_opr>
                                        <rel>relation</rel>
                                    </_opr>
                                    <var>y</var>
                                    <var>rel2</var>
                                </atom>
                                <atom>
                                        <_opr>
                                            <rel>object</rel>
                                        </_opr>
                                        <var>y</var>
                                        <var>c</var>
                                </atom>
                            </and>
                        </_body>
                    </imp>
</rulebase>
```

APPENDIX B

PROTOTYPE IMPLEMENTATION – USER GUIDE

The tool can be used to read in RDF(S), DAML, OWL documents. Also rules in the form of RuleML can be imported. The tool converts constraints expressed in any of the formats to a java based representation using Mandarax API and the Oryx API.

PACKAGES NEEDED

Jdk1.4 and above.

From jena 2.0, we need to set classpath to
\lib\jena.jar,\lib\log4j-1.2.7.jar,
\lib\antlr.debug.jar,
\lib\concurrent.jar
\lib\icu4j.jar
\lib\jakarta-oro-2.0.5.jar
\lib\junit.jar
\lib\rdf-api-2001-01-19.jar
\lib\xercesImpl.jar
\lib\xmlParserAPIs.jar

From oryx3.3 we need to set classpath to
\lib\jdom-b8.jar
\lib\jhall.jar
\lib\jndifscontext.jar
\lib\jndiproviderutil.jar
\lib\junit-3.8.1.jar
\lib\log4j-1.2.8.jar
\lib\metouia.jar
\lib\mm.mysql-2.0.6.jar
\lib\pf-joi-full.jar
\lib\sqlx.jar
\lib\mandarax-2.3.1.jar
\lib\oryx-3.3.jar

\lib\oryx-examples-3.3.jar
\lib\oryx-help-3.3.jar

Finally set classpath to Confider.jar

HOW TO RUN IT

To start the GUI at the command prompt type

java confider.gui.startup

HOW TO OPEN FILES

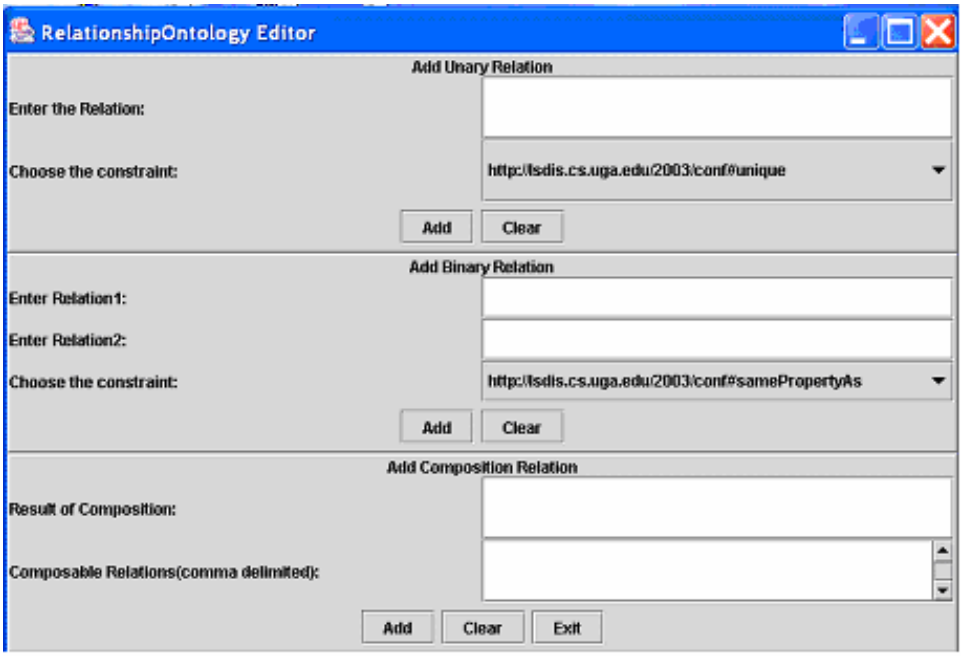
To open RDF,RDFS,DAML,OWL files use the following image icon from the menu

To import RuleML files use the following image icon from the menu

HOW TO EDIT THE RELATIONSHIP ONTOLOGY

To edit the Relationship Ontology click on the following image icon from the menu

The following form will be presented which enables you to add the relations between relations:



The Unary relation panel can be used to add relation about a relation called ‘unique’, ‘unambiguous’, and ‘asymmetric’. The binary relation panel can be used to add relations between two relations like ‘disjointPropertyFrom’, ‘samePropertyAs’, and ‘similarTo’. You can specify the relations (properties) using a fully qualified URL “www.foo.com/test#prop1” or just “prop1” if your target namespace is a single namespace. The Composition relation template lets you add any number of relations that can be composed to a single relation.

HOW TO ADD STATEMENT SIMPLIFICATION RULES

To add statement simplification rules click on the following image icon from the menu

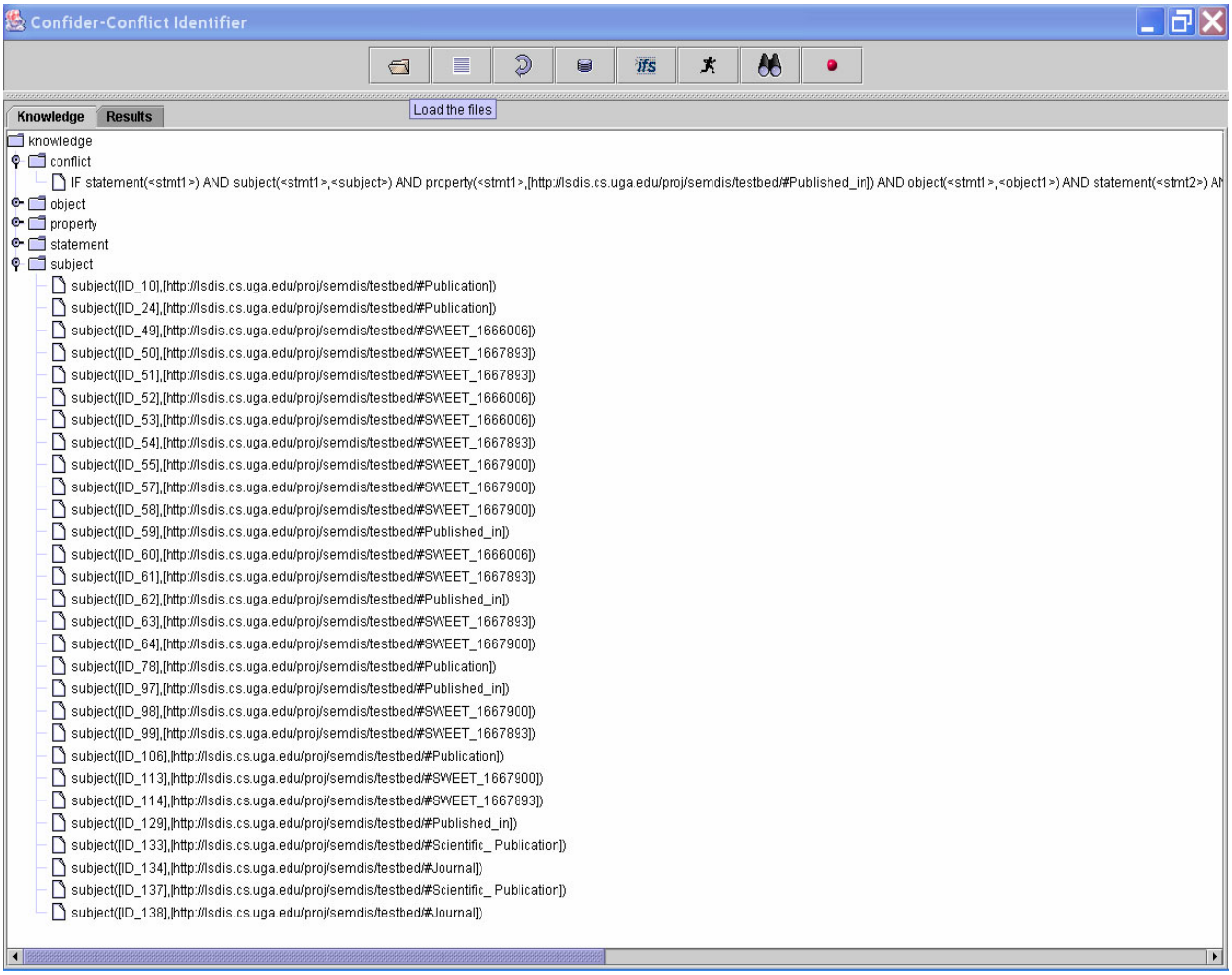
The following screenshot shows how you can use it to specify rules using an example:

The ‘IF’ panel is used to put in the prerequisites of the rule and the ‘THEN’ panel is used to input the result of the rule.

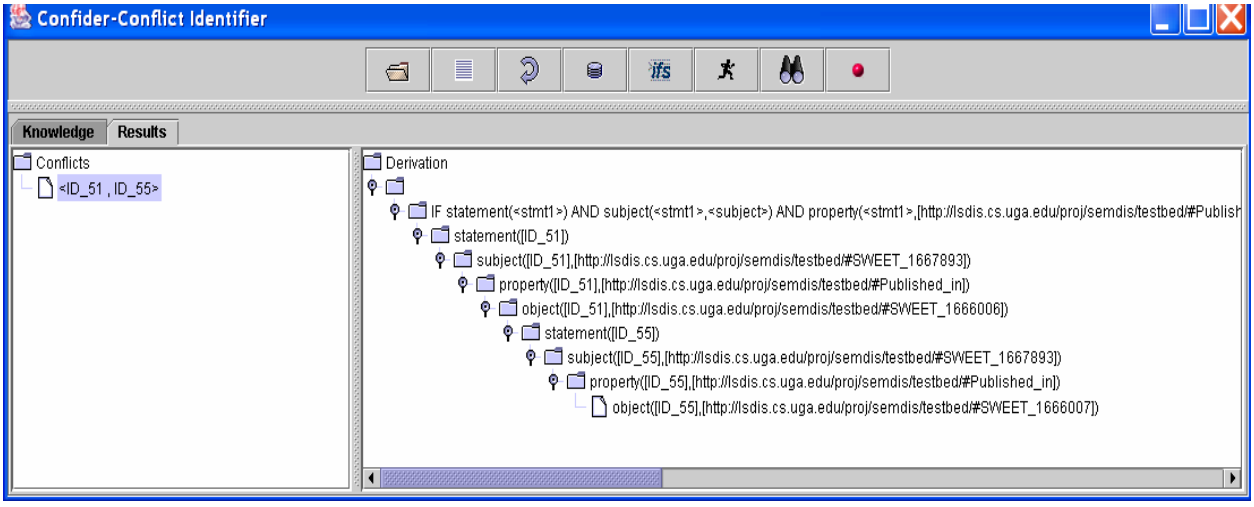
HOW TO VIEW AND INTERPRET THE RESULTS

There are two views in the GUI, the knowledge view and the results view available as tabs. To run the conflict query click on the following image icon available in the menu

You can view the knowledgebase changes as you load files. The following screenshot shows the knowledge view expanded:



Once you have run the query the result view is populated and you can switch over to view the results.

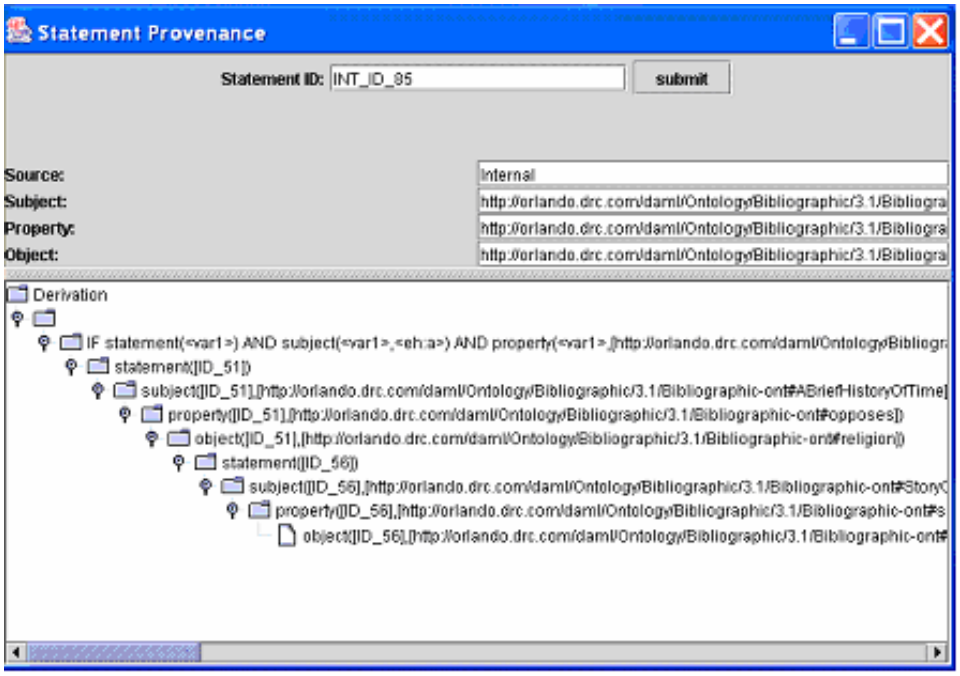


In the figure above, the left pane shows the statements that are in conflict. The right pane shows the rule that triggered this conflict and the associated statements.

HOW TO VIEW INFORMATION ABOUT A STATEMENT

When the conflict query is run, the simplification rules are also executed. The statements that are the results of simplification are given the ids with prefix “INT_ID” denoting interpreted or summarized. To view how that statement was arrived click on the following image icon in the menu

The following screenshot shows the interface that is provided to view information about a statement:



Type in the id and press submit. The subject, property and object of the statement will be shown. The source indicates the document from which the statement was read in or in the case of

simplification it will be ‘internal’. The lower panel gives the simplification rule and the associated statements how this statement was arrived at.