Project Work

Probabilistic Extension For Rule Language

Submitted By: *Sevkan Taskan*
Submitted To: *Prof. Ralf Moeller*
Submitted Subject: *Probabilistic Extension For Rule Language*
Submitted Date: *07/10/2005*

Technische Universität Hamburg-Harburg

I should like to thank my supervisor, Prof. Ralf Moeller for his supervision, direction and encouragement throughout the whole process of my project work. Without his valued criticism and help, I would not have finished the project.

Many Thanks

Sevkan Taskan

Table Of Contents

1 INTRODUCTION

The goal of the project is to design and develop a knowledge system with the probabilistic extension for rule languages. This project checks whether there is a possibility to implement probabilistic extension for rule languages where as these can be pDATALOG and nRQL query language [See: 63] which is already used by Racer System [See: 64].

The scope of the project will figure out in what ways the nRQL can handle or allow an application to use ontologies that are convertible to / compatible with the format specified by the probabilistic Datalog.

As in every research, the first step of this project work is started by collecting the related information about web ontology languages, their semantics, syntaxes and the restrictions [See: 9, 14, 24, 25, 46, 47, 52, 74, and 75]. But the web ontology languages are not only considered but also the logical part which is mostly composed of the probabilistic Datalog is also studied. Before dealing with the probabilistic Datalog, Datalog [See: 36] and the probability theory are also researched.

The overview of the sections can be classified as probabilistic Datalog [See: 1, Chapter 2 in this volume, 6, 12, 30, 35, 36, 58], OWL LITE web ontology language [See: Chapter 3 in this volume, 8, 14, 40, 42, 46, 47, 52, 53, 55, 74, 75], probabilistic Datalog with OWL LITE [See: Chapter 4 in this volume,13, 29, 34], nRQL [See: Chapter 5 in this volume, 63] possibility of implementing probabilistic extension for rule language [See Chapter 6 in this volume].

The OWL LITE web ontology language and probabilistic Datalog with OWL LITE sections are mentioned because of having the opportunity of getting the basic knowledge how a logic language can be implemented into web needs by paying attentions to the steps and procedures that are taken under notation while implementing a language onto another or extending a language by the help of other languages.

Another reason why OWL LITE web ontology language is chosen, is the being an extension of RDF [See: 27, 28, 71, 73], RDF-Schema [See: 27, 28, 46, 73] and the revision of DAML-OIL [See: 16, 27, 28, 35, 36, 72]. Therefore, RacerPRo can also read RDF, RDF-Schema and DAML-OIL languages. The way that was chosen in order to reach the goal of the project, is trying to find out the similarities and differences of the related concepts which can light the idea to find a way of implementing the probabilistic extension under some restrictions and some given models.

The brief information about nRQL, in which the structures, semantics, synopsis and the other related query tools like ABoxes [See: 15, 23, 63 page 53, 64 page 139], TBoxes [23, 63 page 53, 64 page 142] are explained, gives the vague scene for the “probabilistic extension for rule language” part of the project.

These fields of computer science are not explored enough and these concepts are newly known. The aim of the project is to search and figure out the new coming possibilities that will be the combination of the known concepts in order to create more expressive and powerful languages. For the following parts of the project work, we assume that the reader is familiar with given topics.

2 PROBABILISTIC DATALOG

2.1 Abstract

As a first step the usage of probability and also the explanation why probabilities are used in Information Retrieval (IR) [See: 5, 12, 29, and 34], are pointed out. The uncertain information which Information Retrieval deals with is the first concept which must be taken under notation. Also another typical problem of Information Retrieval is how well the query is matched to the data or how relevant the result is for the given query. Also the answers and the questions are focused on the information which can be sometimes certain or uncertain. Therefore, probability theory [See: 51] seems to be the efficient way of solving problems with regard to uncertainty. But nevertheless writing the ways of recovering the problems is not as difficult as implementing into the open world assumptions. The probability takes place under the concept of ranking principle in IR but in this report the concept of probability ranking principle or probabilistic indexing or probabilistic inference is not included.

But the idea of this first paragraph is just to point out the needs of probability and probabilistic theory to quantify the uncertainty. Then these terms can be added to the Datalog to create our probabilistic Datalog. For more detail about the indexing, ranking or inference, the following concepts of Computer Science can be researched: Probability Theory, Bayes' Decision Rule, BII (Binary Independence Indexing), BIR(Binary Independence Retrieval), DIA (Darmstadt Indexing), n-Poisson Indexing [See: 5, 12, 66].

After point out the place of the probability in Information Retrieval [See: 12], the main topic “Probabilistic Datalog (pDatalog) can be discussed in the following paragraphs.

The informal description of Probabilistic Datalog (pDatalog) is a combination of classical Datalog with probability theory where Datalog is a variant of function-free Horn logic.

Interference of probability theory comes to a conclusion which is inconsistency or certain restriction which is occurred at the former version that is based on extensional semantics. In order to gain the efficient result the latter steps in pDatalog requires intensional semantics [See: 69].

Combination of Datalog, probability theory and the intensional semantics with the logical rules are the milestones of probabilistic Datalog. In probabilistic Datalog probabilistic weight is attached to every fact or rule. Therefore, there is a link between the term and the assignment of probabilistic weight which is linked to the fact. The probabilistic weight of the related fact is computed by the means of intensional semantics.

The advantages of probabilistic Datalog:

- Enables the modelling of the new hypermedia retrieval,
- Enables the classical probabilistic models of Information Retrieval (IR),
- Provides powerful inference,
- Can be used as a query language for Information Retrieval or database system.

2.2 Syntax and Semantics

In this part syntaxes and semantics of “Probabilistic Datalog” are described [See: 6, 29, 30, 34] so that it will be easier to compare or map onto nRQL in the following sections.

2.2.1 Syntax

In order to describe the pDatalog syntaxes, the syntaxes of Datalog can be given and the differences they have or the similarities can be shown.

In Datalog the variables are starting with capital letters. Constants which can be alphanumeric strings or numbers are starting with lower-case letters. Predicates which are the alphanumeric strings are starting with lower-case letters. Variables, constants and predicates are the basic elements of Datalog.

A term can be either a variable or a constant (alphanumeric strings or numbers).But in Datalog a ground term can only be a constant.

An atom is one of the syntaxes that consists of an n-ary predicate symbol and the list of arguments that each one is a term where;

Atom = p ($t_{1,}$…,t_n),

N-ary predicate symbol = p,

List of arguments = (t_1,....., t_n),

Term = t_i.

A literal is an atom p (t_1,…….,t_n) or a negated atom ¬ p (t_1,…….,t_n).

A finite list of literals is called clause.

Four types of clauses are declared in the syntax.

1) Ground Clause: This is the clause in which variables can not be found.
2) Negative Clauses: These are the clauses that consist of only negative literals.
3) Positive Clauses: These are the clauses that consist of only positive literals.
4) Unit Clause: This is the clause that consists of only one literal.

Facts are the ground clauses which also have to be unit clauses.

Rules are the clauses with one positive literal. The positive literal part of the rule is called “head” and the list of negative literal of the rule is called “body” or sometimes “subgoals”.

Rules have the form which is composed of the “head” and the subgoals of the “body”. These terms denote literals with variables and constants as arguments. A rule can be seen as a clause.

Example:

father(X, Y) :parent(X, Y) & male(X)

This denotes that father(x, y) is true if both parent(x, y) and male(x) are true for two constants x and y.

father (X, Y) → The head of this rule

parent (X, Y) → One of the body literal (Conjunction)

male (X) → One of the body literal (Conjunction)

Also a fact can be a rule if the fact satisfies the condition which has constants in the head and an empty body.

Example:

parent (jo, mary).

The last syntaxes before passing to the semantics of Probabilistic Datalog are the predicates. The set of predicates is the disjoint sets of IPred and EPred where the “I” stands for the intensional part and “E” stands for the extensional part. All elements of each predicate group denote their own predicate groups in which they are defined by the means of the rules.

Up to this point, the syntaxes of the pure Datalog are described and also some parts which are different from probabilistic Datalog but still the syntaxes of probabilistic Datalog are not discussed totally. As it is mentioned at the introduction part, probabilistic Datalog is an extension of stratified Datalog with the probability concept. The negation literal can be added to the syntax. Therefore the stratified Datalog takes place and allows using negation literals in rule bodies. A probabilistic Datalog program P is composed of two different sets which are P_E and P_I. P_I is the part which is the set of stratified Datalog rules and P_E is the set of probabilistic ground facts with probabilistic weight. First step with syntaxes of Datalog follows the path to the stratified Datalog and ends with probabilistic Datalog.

2.2.2 Semantics

Semantics are the probability distributions over the set of all possible worlds which are the well-founded models of union of deterministic part which is “True” or “False” and the subset of nondeterministic part.

Possible world can contain atoms ….

a…….. ¬ a ……. none (unknown) …….. Both (inconsistent)

An example for possible world semantics [See 34 page 31]:

0.9 docTerm (d1, ir). (Ground facts with probabilistic weights)

P (W_1) = 0.9 :{ docTerm (d1, ir)}
P (W_2) = 0.1 :{}

Possible interpretations [See 34, page 32]:

0.9 docTerm (d1, ir).
0.5 docTerm (d1, db).

I_1:
P (W_1) = 0.45 : {docTerm (d1, ir)}
P (W_2) = 0.45 : {docTerm (d1, ir), docTerm (d1, db)}
P (W_3) = 0.05 : {docTerm (d1, db)}
P (W_3) = 0.05 : {}
I_2:
P (W_1) = 0.5 : {docTerm (d1, ir)}
P (W_2) = 0.4 : {docTerm (d1, ir), docTerm (d1, db)}
P (W_3) = 0.1 : {docTerm (d1, db)}
I_3:
P (W_1) = 0.4 : {docTerm (d1, ir)
P (W_2) = 0.5 : {docTerm (d1, ir), docTerm (d1, db)}
P (W_3) = 0.1 : {}

Semantics are divided into two sets:

1) Intensional Semantics: In this semantic the weight of intensional database fact is shown as the function of weights of the underlying ground facts,

An example for the implementation of intensional semantics [See 34 page 26, 27]:

Event keys and event expressions:

0.9 docTerm (d1, ir). [dT (d1, ir)]
0.5 docTerm (d1, db). [dT (d1, db)]

0.7 link (d2, d1). [l (d2, d1)]

?- docTerm (D, ir) & docTerm (D, db).

gives
d1 [dT (d1, ir) & dT (d1, db)] 0.9 0.5 = 0.45

about (D, T) :- docTerm (D,T).
about (D, T) :- link (D, D1) & about (D1, T)

?- about (D, ir) & about (D, db).

gives
d2 [l (d2, d1) & dT (d1, ir) & l (d2, d1) & dT (d1, db)] 0.7 ·0.9 ·0.5 = 0.315
d1 [dT (d1, ir) & dT (d1, db)] 0.9 ·0.5 = 0.45

Remark: computation of probabilities for event expressions [See 34 page 30]

1. Transformation of expression into disjunctive normal form

2. Application of sieve formula:
c_i - conjunct of event keys

$$P(c_1 \vee \ldots \vee c_n) = \sum_{i=1}^{n} (-1)^{i-1} \sum_{1 \le j1 < \ldots < j1 \le n} P(c_{j1} \wedge \ldots \wedge c_{ji})$$

2)Extensional Semantics: In this semantic the weight of derived fact is shown as the function of the weights of bodies or subgoals.

An example for an Extensional Semantics [See 34 page 25]:

0.9 docTerm (d1, ir). 0.5 docTerm (d1, db). 0.7 link (d2, d1).

about (D, T) :- docTerm (D, T).
about (D, T) :- link (D, D1) & about (D1, T)

q (D) :- about (D, ir) & about (D, db).

P(q(d2))= P(about(d2,ir))P(about(d2,db))= (0.7 0.9)(0.7 0.5)

Remark: extensional semantics only correct for treelike inference structures.

Computation of probabilities in pDatalog is based on event keys and event expressions. By having event keys and event expressions, disjoint or duplicated events are recognized (“intensional semantics”) during the computation of the probabilistic weight. So these concepts in pDatalog prevent to yield irrelevant probabilities.

The scenario goes like that;

→ All facts and instantiated rules are basic events.

→ Each of fact and instantiated rule have assigned a unique key event.

→ Each derived fact is associated with an event expression.

An event expression is a Boolean combination of the event keys of the underlying basic events. Therefore, this scenario is just like a trigger for the probabilistic Datalog in order to prevent the systems and this can be occurred because of the disjoint or duplicated events.

From the previous pages it is known that the probabilistic weight gives the probability of the predicate to be true. But just for being an example of disjoint event, to apply the probabilistic Datalog rules for the hyperlink structure (recursive definition in pDatalog). Applying the probability theory for the hyperlink structure can easily cause trouble, because if the multiple rules for the some head and the events are independent, the probabilities are multiplied which are related to the given predicates. But another question rises up; therefore it deals with a hyperlink structure, is there any possibility of having the probability for the given two links twice. The answer is known that would be “Yes”. That is why some simple approaches are not enough or not capable of distinguishing.

Problems rise up for the probabilistic independence of the bodies of the rules not only for the disjoint events.

As it is mentioned in the abstract part;

Interference of probability theory comes to a conclusion which is inconsistency or certain restriction which is occurred at the former version that is based on extensional semantics.

In order to gain the efficient result the latter steps in pDatalog requires intensional semantics. By applying the intensional semantics onto probabilistic Datalog, given Boolean expression can help to identify the events that occur more than once or disjoint events. The background of this operation the basic events are just dealt with, unique event keys, derived facts for IDB predicates and the Boolean combination of the events keys for the related EDB facts.

This scenario is just the same scenario that it was already mentioned in this semantic part of the probabilistic Datalog but this time the scenario is just implemented with intensional database predicates and the extensional database predicates.

2.3 Four Valued Probabilistic Datalog

In this last section of Probabilistic Datalog which is the four-valued probabilistic Datalog [See 30, 35] the possible world assumption is changed from the closed-world assumption [See: 63 page 46, 65] into an open-world assumption [See: 63 page 46, 70] . As it is understandable from the title four different values are available for the facts which are true, false, unknown and inconsistent.

If the evidence can prove that the information is both true and false for the given fact, than the fact has the truth value “inconsistent”. If no information is supplied, the fact has the truth value” unknown”.

Example: For facts, probabilities for true, false and inconsistent are specified.

- 0.6/0.3/0.1 male (mark).

  Mark is a male with probability 0.6, not a male with probability 0.3, and the knowledge is inconsistent with probability 0.1.

- !female (john).

  John is not a female and also from this fact it is easy to derive that men are not women because here, the negated head stands for the probabilities “0/1/0” which respectively true, false and inconsistent.

- 0.9 male (anthony).

  Anthony is a male with probability the “0.9” and not a male with the probability “0.1”, it is easy to derive the other format from the given form which is the short form for “0.9/0.1/0”.

The deterministic datalog program has a chance to contain the four truth values.

2.4 Conversion of Four- valued into Two-valued pDATALOG

Also another scenario that can be occurred, is having a set of possible worlds for the corresponding deterministic Datalog (where this will be true or false) with the probabilistic distribution.

Program which is adapted for four-valued Probabilistic logic can be easily implemented into two-valued Probabilistic Datalog program. In two-valued

Probabilistic Datalog program, two distinct predicates can be obtained which is either a positive knowledge or a negative knowledge.

Two distinct predicates: Positive (a) and negative knowledge (¬ a).

For each fact in two-valued probabilistic Datalog three different values can be obtained which are true, false and inconsistent.

3 OWL LITE

3.1 Acknowledgement

OWL LITE is a type of web ontology. Before OWL LITE, a little bit information is given about what the ontology is.

3.1.1 Ontology

An ontology is a representation of terms and relationships of these terms [See: 14, 46] .Ontology is an artifact language which is created by engineers. It is formed by the specific vocabulary which is used to describe the truth or any event and also the set of assumptions which are related to the meaning of these vocabularies. An ontology is a bridge between the terms and what the terms stand for in the given scenario. It just connects the terms within the given relationships and constitutes a meaningful language for a certain domain.

Therefore, ontology is used to describe a certain domain by the specification. These specifications are composed of the formal-machine models and the understandings of a certain domain.

The structure of an ontology can be described by the main components of itself. The main components in the ontologies are names and constraints. An ontology consists of facts and axioms. The description of classes and the relationships between each other, the description of attributes and the relations between elements are included in the ontology. That is why OWL web ontology language can easily support the vocabularies, terms and the understandings of the content of the given information by the help of the properties. Object and datatype properties can be an example.

3.2 Abstract

This section of the project work describes the first impression and the basic concepts of the OWL LITE web ontology language [See: 8, 9, 52, and 75]. OWL LITE like the other web ontology languages is designed for the applications which are processed the content of the given information. In another way OWL is needed when the application needs to process the content of the given information .There are some reasons for choosing OWL [See: 53] instead of RDF, RDF-Schema and XML .Because OWL has more opportunities to represent the understandings and the semantics [See: 47] than RDF, RDF-Schema and XML. RDF-Schema is converted into a full knowledge representation language for the web by OWL. In the later subsections of this section, this part is explained in details by descriptions, semantics and synopsis of OWL LITE.

The OWL web ontology language has an efficient machine interpretability of web content by applying a formal semantics. These formal semantics create the sublanguages of OWL which are named OWL LITE, OWL DL and OWL FULL.

Now OWL LITE is considered as a starting point of the project work with regard to the probabilistic Datalog (pDatalog). Another reason for choosing OWL is being a revision of DAML-OIL web ontology which is related to RACER system. After collecting the necessary and the efficient information, the vague picture of the research topics with clear ideas can be seen.

OWL LITE is a sublanguage of OWL DL that supports the subset of OWL language constructs. OWL LITE is mostly the choice of the implementers or the users who want to start with language feature with its basics. Therefore the users who need to have simple constraint features and the classification hierarchies choose the OWL

LITE .Because OWL LITE has a simple and quicker migration path for the taxonomies.

Also the most of the used features in OWL and DAML-OIL can easily be found in OWL LITE concept within the descriptions, semantics and synopsis.

There are relations that are already proved between the sublanguages of OWL. These relations are related with the expressions and conclusions that are gained by applications. But there is an important point about these relations that it has to be mentioned; their inverses are not valid and can not be applied to another scenario.

The set of relations are:

- Every legal **OWL LITE** ontology is a legal OWL DL ontology.

  If the specification of a legal OWL LITE ontology is managed then it can be easily assumed that this ontology which is done under the descriptions, semantics and synopsis of OWL LITE can be thought like OWL DL. So the way can be found to implement any subjects of OWL DL into this ontology under the rules and restrictions if it is possible.

- Every valid **OWL LITE** conclusion is a valid OWL DL conclusion.

  If the specification of a valid OWL LITE conclusion is managed then it can be easily pronounced as OWL DL conclusion. One step further from LITE to DL and DL to FULL can be derived. By implementing these conclusions the users or the implementers can add different concepts into the OWL languages like probabilistic Datalog (pDatalog). This will be explained under the name of a section in the project work as a basement of our target topic of implementing probabilistic extension (pDatalog) for rule languages (nRQL).

- Every valid OWL DL conclusion is a valid OWL Full conclusion.
- Every legal OWL DL ontology is a legal OWL Full ontology.

3.3 OWL LITE Synopsis

In this section, an overview of the language features for OWL LITE is given. After just naming them in this section, the following section will explain them shortly.

The following lists are the vocabularies of OWL LITE which are added for describing classes, relations between classes, cardinalities, equalities, properties, typing of properties, characteristics of properties [See: 55, 74] .

If the terms are already present in RDF–Schema and RDF, “Rdf” and “Rdfs” as prefixes are used in OWL LITE Synopsis.

3.3.1 Annotation Properties

The related language features of annotation properties are listed as the followings:

- AnnotationProperty
- Rdfs:comment
- Rdfs:isDefinedBy
- Rdfs:label
- Rdfs:seeAlso
- OntologyProperty

3.3.2 Class Intersection

The related language feature of class intersection is listed as the following:

- *IntersectionOf*

3.3.3 Datatypes

The related language feature of datatype is listed as the following:

- *Xsd:datatypes*

3.3.4 Equality and Inequality

The related language features of Equality and Inequality are listed as the followings:

- AllDifferent
- DifferentFrom
- DistinctMembers
- EquivalentClass
- EquivalentProperty
- SameAs

3.3.5 Header Information

The related language features of header information are listed as the followings:

- *Imports*
- *Ontology*

3.3.6 Property Characteristics

The related language features of Property Characteristics are listed as the followings:

- DatatypeProperty
- FunctionalProperty
- InverseFunctionalProperty
- InverseOf
- ObjectProperty

- SymmetricProperty
- TransitiveProperty

3.3.7 Property Restrictions

The related language features of Property Restrictions are listed as the followings:

- AllValuesFrom
- OnProperty
- Restriction
- SomeValuesFrom

3.3.8 RDF-Schema

The related language features of RDF-Schema are listed as the followings:

- Class (Thing, Nothing)
- Individual
- Rdfs:domain
- Rdf:Property
- Rdfs:range
- Rdfs:subClassOf
- Rdfs:subPropertyOf

3.3.9 Restricted Cardinality

The related language features of restricted cardinality are listed as the followings:

- *Cardinality* (only 0 or 1)
- *MaxCardinality* (only 0 or 1)
- *MinCardinality* (only 0 or 1)

3.3.10 Versioning

The related language features of versioning are listed as the followings:

- *BackwardCompatibleWith*
- *DeprecatedClass*
- *DeprecatedProperty*
- *IncompatibleWith*
- *PriorVersion*
- *VersionInfo*

3.4 Language Description of OWL LITE

This section consists of the description of OWL LITE language features which are named and classified in the synopsis section. Some of the OWL language features with some restrictions and limitations are used by OWL LITE. These are situated as the followings.

3.4.1 Annotation Properties

Annotations are allowed on classes, individuals and also ontology headers by OWL LITE with some certain usage restrictions.

3.4.2 Class Intersection

An intersection constructor takes place in OWL LITE but as it is mentioned for the annotations properties again the usage of this class intersection is limited.

In OWL LITE just the intersection of “named” restrictions and classes can take places. This explanation is also named as a term “IntersectionOf”.

3.4.3 Datatypes

In OWL LITE, the sets of data values are denoted by the OWL datatypes. OWL Datatypes are XML Schema datatypes like string (xsd: string) or float (xsd: float) and the other types.

3.4.4 Equality and Inequality

The following steps consist of the equality and inequality features of the OWL LITE.

- AllDifferent

This feature is used if the distinct objects with the enforcement of names assumptions of the given distinct objects are occurred. Implementing this feature into OWL LITE, getting individuals with mutual distinction can be seen if it is necessary for the application or the scenario of the content of the information which is given.

- DifferentFrom

This feature is used if an individual is wanted to be stated different from the other individuals.

- DistinctMembers

This feature is used if the all members of the given list are wanted to be stated distinct.

- EquivalentClass

This feature is used if two different classes are wanted to be stated equivalent. All equivalent classes have the same instances and also in some cases this is used to create synonymous classes.

- EquivalentProperty

All properties that are already mentioned for EquivalentClass are also applicable for this feature but instead of classes all the statements will be applied to the properties.

This feature is used if two different properties are wanted to be stated equivalent. Also in some cases this feature can be used to create synonymous properties.

- SameAs

This feature is used if two individuals are wanted to be stated same. Different names can refer to the same individual by using these features

3.4.5 Header Information

The notions of ontology inclusions and information are supported by OWL LITE by the help of this feature.

3.4.6 Property Characteristics

- DatatypeProperty

This feature is used if an instance of a class is wanted to be related to an instance of the datatype.

- FunctionalProperty

This feature is used for the property that has at most one value for each individual. But that does not mean that it must have at least one value for its individual, no value for an individual can be accepted. Having a unique value can be stated by this property.

- InverseFunctionalProperty

If a property is inverse functional then it means that the inverse of the given property is also functional. By this way properties can be stated as inverse functional.

- InverseOf

This feature is used to state one property as an inverse on another property.

- ObjectProperty

This feature is used to relate an instance of a class to another instance of a class.

- SymmetricProperty

This feature is used to state the property as symmetric.

- TransitiveProperty

This feature is used to state the property as transitive.

3.4.7 Property Restrictions

- AllValuesFrom

This feature is used with respect to a class if a restriction is wanted to be putted on to a property. This draws a local range restriction for the related class on which the property is applied. In other word a second individual can be implied as an instance of the restriction class where the instance of the first class is related with the property.

- OnProperty

This feature is used to put an indication to the restricted property on which element it is applied by.

- Restriction

This feature is just used in order to figure out the usage of the properties with respect to the instances of the classes.

- SomeValuesFrom

This feature is used to apply a partial restriction on to the property again with respect to the classes. It also stands for the clue of having a restriction on at least one value which is certain for the given property.

3.4.8 RDF-Schema

- Class (Thing, Nothing)

A set of individuals are built up the each classes which are defined on the behalf of the ontology. The set of all individuals are represented as owl: Thing and the empty set is represented as owl: Nothing.

- Individual

Informal description individuals are the bridges that are used to relate one individual to another. They are instances of properties and the classes which are ready to be related to each other. In OWL LITE individuals can have same identities or different identities.

- Rdfs:domain

This is used to show the property limitation of each individual.

- Rdf:Property

This feature is used to state the relationships which can be observed from individuals to individuals or data values.

- Rdfs:range

This feature is used to create the range of limitations of the individuals where they can be occurred as their own values.

- Rdfs:subClassOf

This feature is used to show class hierarchies that one class is a subclass of another class.

- Rdfs:subPropertyOf

This feature is used to show the property hierarchies that one property is a subproperty of another property or more then one property.

3.4.9 Restricted Cardinality

OWL LITE has the cardinality restrictions. The cardinality which is related to OWL LITE can only have a chance to get a value of “0” or “1”.The arbitrary values can not be taken places in the cardinalities of OWL LITE

- *Cardinality*

This feature is used to show the availability of the properties on a class under the satisfied conditions of having both maxcardinality and mincardinality of the values of “1” or “0”.

- *MaxCardinality*

This is feature is also called a functional or unique property. As a structure it shows the instance of the class which has at most one related individual by the applied property, where it has a maxcardinality with the value of “1”.

- *MinCardinality*

This feature is used if the property is needed for all the instances of the class.

It means that any instance of the given class can easily be connected to at least one individual if the property has minCardinality with value of “1” which is respected by the related class.

3.4.10 Versioning

OWL extends this vocabulary which is already mentioned by RDF to describe the versioning information.

3.5 Results of OWL LITE

This section consists of the conclusion of having OWL LITE, design goals and some important hints about OWL LITE.

One of the additional aspects of OWL LITE which differs itself from RDF is being scalable to web needs. In another word it can be easily distributed through many systems and can be extensible. By applying logical expressions, local and optional properties, OWL LITE extends RDF-Schema in to a proper well-defined language which is suitable for web needs.

Some of the design goals of OWL LITE can be written as the followings:

- Ease of use
- Interoperability
- Adaptability to the existing standards
- Internationalism
- Complexity
- Shareable

As a conclusion OWL LITE has DL-based semantics and reasoning which are done via DL-engines. But it has also a restricted cardinality which can not be adapted except the value of “1” or ”0”. There is no explicit negation in OWL LITE but negation can be encoded by using disjointness and the disjunction can be encoded with negation and conjunction.

4 Mapping OWL LITE onto pDatalog

In this section of the project work the implementation of the probabilistic extension to OWL LITE is presented. Although these are still ongoing research areas in which the users and the implementers are trying to extend, map or combine the probabilistic extension (probabilistic Datalog) onto OWL web ontology languages. Some probabilities of the successful implementations are proved by the related people in these research paperworks [See: 13, 34]. The one which its subsections are already mentioned in this project, is the mapping probabilistic Datalog onto OWL LITE. This already proved implementation [See: 34] can be a light to seeking the probabilities of implementing probabilistic extension (pDATALOG) for the rule language (nRQL query language of Racer system).

Also after this section, nRQL query language which is used by Racer system will be explained with its semantics, synopsis, structures, descriptions and restrictions as in the same way and mentality where it has been done for the mapping pDatalog on OWL LITE by the researchers. So as a first step of all related research areas, where the goal of the project is just to combine or extend one of the web ontologies with another one is just passed by brief researches of each topic. Before mapping or extending one onto another one all the restrictions and the expressiveness of each language should have to be figured out in details in order to create the efficient combination where it is sufficient under the possible circumstances, predicates or rules which are given.

The main solution of mapping the probabilistic datalog is done through the additional OWL classes. Implementing these additional classes into the OWL, the problem of marking the probabilities of each OWL is solved.

The classes, properties, individuals and literals are observed in details for a successful implementation. Then the basic ideas which are composed of unary and binary datalog predicates for classes and the properties plus the constants are matched for the mapping procedure. Besides these terms the properties are studied in details and applied for the given scenario like inverse, transitive and symmetric. After all these documentations, the restrictions and the barriers which are situated in front of OWL LITE are pointed out like the cardinality and existential.

But these difficulties can be solved by the four-valued probabilistic Datalog which is based on the open-world assumptions. The way of implementing the solution of this problem can be the set of rules that are defined to use for solving the problem of OWL LITE restrictions onto probabilistic Datalog rules. The pDatalog rules can be used for the cases where OWL LITE is not sufficient. As a result of this compensation the lack of OWL LITE languages for some cases can be covered where the rules would be solution in order to handle the cases. The whole procedures that are mentioned in this paragraph are named as the mapping probabilistic Datalog onto OWL LITE.

But that does not mean that the mapping pDatalog onto OWL LITE has a chance to handle every case. Therefore many primitives can not be mapped onto Datalog.

However the inconsistencies like existential quantifiers and the cardinality restrictions can be detected by the help of the four-valued probabilistic datalog.

The procedure starts with the alternation of classes and properties into unary and binary predicates and it is followed by mapping the primitives as much as it would be possible under the given conditions and the detections of the inconsistencies.

But at the end of these whole procedure cycles the OWL with the extension of probability and the rule is created.

5 nRQL

5.1 Abstract

In this section the brief information about nRQL is given like the objects, variables, features, notations and the other important topics. Because of this reason, this section is started with a quick overview of nRQL under the title of abstract.

As a first step it can be told that nRQL is the expressive query language of RacerPro's [See: 63 page 1]. Retrieving information from the queries is done through the ABoxes and TBoxes. nRQL allows using query variables which ABox individuals hold and has the mission of satisfying the given queries. On the other hand the vocabulary which is used in the queries is supported by the TBoxes. From this point it can be easily pointed out that conjunctive queries are supported by nRQL via the usage of role terms and concepts.

The main features of nRQL [See: 63 page 55] can be listed as follows:

- Special support is given for querying OWL knowledge bases,
- Support is given for the hybrid queries,
- Complex TBox queries are allowed,
- Support is given for the concrete domains,
- Role chains and complex predicate expressions are available,
- Complex queries are available which are built from simple query atoms under well-defined syntaxes and semantics,
- Negation as failure (NAF) [See: 63 page 46] semantics are available.

5.2 The nRQL Language

5.2.1 Query Atoms

Query atoms are the basic expressions of the nRQL language [See: 63 page 57]. Sometimes they are also called simply atoms. These query atoms can be either unary or binary atoms. One object is referenced by the unary atoms and on the other hand the two objects are referenced by the binary atoms. A variable or an ABox individual can be an object.

In nRQL language, the total number of available atoms is three. Each available atom can be negated as failure and ordinary negation.

5.2.1.1 Concept Query Atoms

Concept query atoms are the only query atoms which are unary in nRQL query languages.

In concept query atoms there are “query head” and “query body” as terminologies. The query body is satisfied if the variable is bound to the givens. After this phase of the procedure is satisfied the Racer systems return a list of binding list for the given query. The query head has the responsibility for specifying the format of the returned binding list.

For the transactions of these queries the variables are employed by the active domain semantics where the variables are held by the ABox individuals in the related ABoxes. From this point the possibility of using ABox individuals in the queries can be seen. But choosing the names for the given ABox individuals has to be done very carefully because some specific certain names are reserved in nRQL. Also in

concept query atoms the arbitrary concept expressions can be used but not only atomic names.

5.2.1.2 Constraint Query Atoms

Constraint query atoms are one of the query atoms which are binary in nRQL language. The goal of these query atoms is pointing out the addresses of the concrete domain parts of the knowledge bases. There are some conditions that the constraint query atoms can be used in nRQL. ABoxes, which have TBoxes that do not contain a signature, can be used in the constraint query atoms. The other possibility is to have a signature without containing an individuals-section.

In constraint query language two other features are situated which are called role and feature chain. Nevertheless the feature chains are just like a special case of role chains, so they can be named as role chains only.

Role chains take place in the constraint query atoms in which conditions two single attribute names are not wanted to be used. Because of having a chain only the concrete domain attribute is passed by as the last argument in each chain of the role chains.

The feature chains which are a special case of role chains are functional roles therefore at any place nRQL query can apply on them where a role is used.

As a last remark of this part it is mentioned that in constraint query atoms, not only the simple predicate names should have to used, also the complex predicate expressions can be used by nRQL language.

5.2.1.3 Role Query Atoms

The role query atoms are the last query atoms in nRQL language. Also these query atoms are binary atoms because two objects are referenced. In these atoms role declarations are done in TBoxes.

The role terms can be used in role query atoms but not only the role name. This scenario is just the same for the concept query atoms with the arbitrary concept expressions. In DL's always the negated concepts are preferred instead of the negated roles. However negated roles can be used for ABox querying with nRQL. Because nRQL can compare the given ABox individuals with the role relationship and finds out the result for the related ABox query. Nevertheless negated roles and negated roles in ABoxes are not offered by RacerPro. There is also one difference between the role query atoms and the other two query atoms. The role query atom can be inverted but the others can not be.

5.2.2 Query Head Projection Operators

For retrieving the values of the concrete domain filler attributes of ABox individuals, these projection operators are used. These operators which are called head projection operators [See: 63 page 68] have place in the head query which are a list of objects. The background of needing these operators is coming from the demand for retrieving the values of concrete domain objects. Also variables can not be bound to the concrete domain objects or the values just only to the ABox individuals. But besides these arguments a concrete domain object can be added to the ABox where it can be the filler of the concrete domain attribute. This instance is called a told value

or a concrete domain. As a scenario, the list of concrete domain objects is yield by applying these operators and the list of concrete domain values is yield by retrieving the told values of the concrete domain objects. Moreover, in OWL told values of datatype properties are used for modelling environment. Sometimes fillers, told-fillers, datatype-fillers are used instead of told values. RacerPro can handle the use of datatype properties in OWL with the concept query atoms. The system uses the concept expression to support the extended concept expression syntax in nRQL. Also other property from OWL is annotations-property which is used to annotate resources and also to represent meta information. Again the retrieval of the values of annotation properties can be done by nRQL in the same way as it is already declared for concrete domain objects.

If the attribute projection operator is wanted to be discussed in more details we should have to mention that the concrete domain objects from the ABox which are retrieved by the attribute projection operator have to be from the ABox that are known to be fillers of the concrete domain attribute. So they have to be matched each other before retrieving the concrete domain objects.

5.2.3 Complex Queries

The following constructors and queries [See: 63 page 76-90] are offered by nRQL to be used to combine the query atoms.

- **And**

  nRQL offers the “and” constructor to define the compound or complex queries. Unique name assumption is used for the variables by the RacerPro and according to UNA each variable has to be held by one object in ABox. But

again via RacerPro, the switching off this mechanism for one or all variables can be done whenever it is wanted.

- **Union**

  nRQL offers the “union” constructor to combine the results of queries into one result set. The mechanism for the union constructor: The operator is responsible for computing the union of the answer sets according to the related arguments. But nevertheless the union set can only be efficient and clean if the computed sets have the same arities, in other way the union will compute the wrong the sets for the given arguments. At this point the responsibility of nRQL acts under the proof of having the related sets for the union constructor by considering the name of the variables which are taken as an argument in the queries.

- **NEG**

  nRQL offers the “neg” constructor to implement a negation as failure semantics in the queries. This semantic is mostly used in many applications to measure the completeness of the modelling in an ABox. But this negation constructor is different from the DL’s negation.

  Negation operator can be used with the role query atoms. As it is told for the union constructor again the arities of the answer sets are also important. If the Neg to the binary or the unary atoms are wanted to be applied respectively it will return the binary or unary set as an answer. Also this operator can be applied to the binary constraint query atoms after mentioning the role query atoms. But it is obvious that ABox individual with a negated query atom behaves like a variable on which the unique name assumptions can not be applied.

- **Inv**

  nRQL offers the “inv” constructor to reverse the role query atoms.

- **The Projection Operator for Query Bodies**

  First of all, this operator is different from the operator that is already mentioned with the head projections queries. In these projections nRQL computes the answer sets for the given query expression in which each chain the first arguments give the bindings computed. The resultant set is figure out after computation of the first arguments. By the help of this resultant set before retrieving the binding list, duplication or reordering can be applied as a plus.

5.2.4 Defined Queries

A simple mechanism is used for the specifications of defined queries [See: 63 page 90-96]. In this simply mechanism two parameters take place “formal” and “actual” parameters. The given lists of the parameters are called the formal parameters and these lists can be composed of only variables and individuals. On the other hand in an expression the parameters are called the actual parameters. Actual and formal parameters have to match each other in a proper definition.

Another query in nRQL is the syntactically ambiguous queries. As it is easy to understand from its name, these queries are ambiguous from a binary role query atom. Let’s think about a scenario for this type of queries. A defined query with one arity as well as a concept with the same name as the defined query is given. nRQL will pop up a warning after it assumes that the user refers to the concept name. Also when the same situation is occurred with two arity as well as the role with same name as the define query. Still nRQL will pop up a warning for this ambiguity but again after it assumes that the user refers to the role. But a question can be raised

up. What will happen if the user really wants to refer to the defined query that has the same name as well as the concept and the role? Then the user must use the substitute operator to solve this ambiguity.

Also by the way, already defined queries can be easily used to define new queries.Related to the negation still some problems for the defined queries are left but again this problem can be solved by the projection. Before creating the complement of the query body, a projection has been carried out. Therefore it would be a chance to get the unary complement instead of the binary complement.

nRQL API's can manipulate and access the defined queries but still up to date there was no opportunity to create TBox queries. Instead of that in ABox queries, definitions can be made. These definitions are local and also the defined queries are kept in the related TBoxes. As the TBoxes are changing the definitions are also changing as usual. Also there is a way to put the definition into a different TBox rather than the related one by using the optional keyword.

5.2.5 ABox Augmentation with Simple Rules

In this section before starting with the given title, rough information about ABox is given. After this part, it is confident to deal with the term which is one of the milestones of nRQL like ABox where as the other one which is TBox will be explained in the following parts.

5.2.5.1ABox

ABox is the collection of assertional axioms. In another way it can be easily said that ABox contains the extensional knowledge about the assertions. These assertions are concept and the role assertions. For instance in the concept assertions the concept

expressions are allowed. But in the role assertions where the role is a role expression is not allowed. As a quick overview for the ABox reasoning the realization (data-flow techniques), instance checks, non-subsumption and the graph transformation can be added. The reasoning task of ABox which is the instance checking is done to verify whether the given individual belongs to the specified concept or not. It is just like a checking mechanism for the individuals. By the help of this instance checking the KB consistency is found where it shows that at least one individual has to be addressed to the one of the concepts. Besides the consistency, realization and the retrieval can be figured out by the help of instance checking. But the reasoning can be turned into a little bit complicated phase for the computation because of the individuals.

Reasoning services have to take the whole responsibilities of the KB for the ABox and TBox as well. As the time is passing the lack of a powerful knowledge representation language for the different components that specify the knowledge, is cured by the hybrid reasoning. But it is not an easy way for the knowledge representation while the knowledge components are dealt with the strict interactions.

5.2.5.2 ABox Augmentation

Simple ABox augmentation rules [See: 63 page 96-98] are offered by nRQL to add new ABox assertions to the already existent ABox. If the rules are fired successfully that means that the rules are applicable and after the triggers a set of rule is added to the related ABox. If the RacerPro is set into the mode, the adding procedure is done automatically. Also several nRQL API’s are used to implement different rule applications for the application programs or the users in order to help the adding procedures like under which criteria and how the consequences are added to the ABoxes.

nRQL is not responsible for checking whether the rule is applicable or not. The control of the firing rules is in the hands of the application programs and the users. All the nRQL rules have the antecedents and the consequences. The consequence has the syntax of the ABox assertional axioms where as the antecedent is just like a query body. Generally speaking the variables of ABox axioms from the antecedence (rule body) is called the consequence.

5.2.6 Pseudo-Nominals

In some cases having special individuals in order to refer just for an ordinary expressions can be a plus for a program. RacerPro has the Pseudo-Nominals [See: 63 page 98] as the answer for the given sentence. The Pseudo-Nominals which are the specific individuals can be used to refer within RacerPro expressions, can be handled by the ABox retrieval operations only.

5.2.7 Complex TBox Queries

As the same situation like the ABoxes, in this section rough information about TBoxes is given. After having this opportunity, complex TBox queries [See: 63 page 99-104] can be discussed in details.

5.2.7.1TBox

Namely it is the short form of the terminological box. TBox is the collection of the concept axiom where as the ABox is the collection of assertional axioms. It shows how the concepts and roles are related to each other. One of the most important elements in DL knowledge base is the operation which is used to build the terminology. TBoxes give the shape of the forms and the meanings of the declaration which are closely related to the operations. These basic declarations in a TBox are

known as the concept definitions. The concept definition is always renewed itself according to the previous defined concepts. Also in TBoxes these concept definitions are used to build up the needed and the efficient conditions for the individuals. Due to TBoxes only just one definition is allowed for the concept name which is acyclic. The necessity of using the concept name is based on the ontological decision and also another effect can be the idea of having a terminology for the important aspects in TBoxes. Also these concept names can be ordered by one of the reasoning services of TBoxes which is the subsumption. These names can be started from the specific ones and ended with the very simple and uncomplicated ones.

In a sense acyclic TBoxes can be expanded or unfolded. These two restrictions are mostly common for DL knowledge bases. That is why the argument which is concept definition has some more importance in DL. Just only the atomic concepts are replaced into the complex expressions during the expansions of the defined concepts. But these expansions allow the complexity therefore the solution depends on the brief study on subsumption where it is one of the reasoning services for TBoxes. The other TBox reasoning can be the classification of clustering and the order of nodes also besides that the transformation of general axioms can be added to these groups.

After this quick overview of TBox as the definition, concepts, terminologies and the reasoning services, the topic under the name of complex TBox queries in nRQL can be explained in more details.

5.2.7.2Complex Queries

Relationship patterns of superclass and subclass can be searched in the taxonomies of TBoxes in nRQL. These taxonomies of TBox are called directed acyclic graphs.

These directed acyclic graphs are composed of the equivalent concepts and the directsubsumer of relationship which are respectively represented by the nodes and the edges.

The related scenario for the “taxonomy of TBox” is acted by the role membership axioms (with what the edges are represented) and concept membership axioms. From the equivalence class the name of the node can be chosen. nRQL can query this taxonomy ABox and also the full of nRQL can query the taxonomy TBox with some limitations of the available concept expressions.

5.2.8 The Substrate Representation Layer

The association of a RacerPro ABox with an additional representation layer is done through layer which is called substrate representation layer [See: 63 page 104-117]. Mostly the support for the representation of semi-structured data is given by the substrate layer.

The following three types of substrates are offered by nRQL.

5.2.8.1 The Data Substrate

These types of substrates are pretty much similar to the ABoxes. The nodes and edge-labelled directed graphs are seen in these types. Optional description which is called “label” can be applied to the nodes, but the edges has no opportunity not to have a label. It is a must for the edges to have a label in this substrate model. These data substrate labels can be a list of data literals or just a simple data literal. In these data substrate labels, strings and numbers are allowed as well as the characters. LISP supports the data literals for the data substrate data labels. There is a common

idea that a boolean formula in CNF is pretty much the same as the data substrate label. But also besides the CNF the nRQL can allow to use the data predicates.

Up to that part the information about the data substrate layer is given to add the label but the way of creation is not included till this part. The first step is creating a data substrate node with an appropriate name. A valid name can be created by the symbols, numbers or characters. Then the next step will be the labelling the current node with the description. In the meantime ABox individual creation is needed with the addition of the concept assertion to the ABox which associates with the related data substrate.

After the creation is completed it is always assumed that the object from the created substrate realm is needed whenever the individuals or the variables are used by the users or the application programs.

The reason of the assumption is coming from parallel bounding of ABoxes and the substrate variables. That is why when a binding is made for the variables, the program automatically establishes the corresponding bindings for the related bindings.

Also the data substrate variable satisfies the given data node query expression which acts in the same way as the role and the concept expression for ABox query atoms. The structure is again built up from the nodes and the edges label.

5.2.8.2 The Mirror Data Substrate

The mirror data substrate is used when the automatically creation of the data substrate objects for ABox individuals is necessary. The creation is done for the given ABox with respectively objects and the descriptions of the substrates. This facility is used

mostly by the OWL knowledge bases. As a result RacerPro can handle the automatic creation of the appropriate ABox from the given OWL file. If the RacerPro is set in to this mode then it means that for every element in OWL knowledge bases, RacerPro is responsible for creating the related data substrate objects.

The node of the data substrate is created for each concrete domain object, value and also for the ABox individuals in the mirror data substrates. On the other hand edges are created for the constrained axioms in the mirror data substrates.

After these creations the problem is to distinguish the type of the data substrate objects and these problems are solved by applying the special markers to the appropriate data labels for the given objects. These special markers are applied for the each Abox individuals, concrete domain values and the concrete domain objects. From the notation of these special markers it is so obvious to understand for which case it is applied.

5.2.8.3 The RCC Substrate

The region connection calculus is used to describe and to give the reasoning between the spatial objects. This substrate model is designed for the applications in which cases they represent the object with the spatial characters. So by this RCC substrate which is a special case of data substrate, nRQL can have a chance to support the demands.

RCC networks yield the RCC substrate to create and to query. The difference from other substrates can be explained by having edge labels as constraints. These edge labels are labelled by RCC relationships but as an opposite fact nodes are not constrained.

These relations can be specified as flat lists or single symbols that represent the disjunctions. The coarser knowledge between two spatial objects can be expressed by the disjunction of these relationships. RCC network can react with ABox individuals in order to describe and to reason the spatial descriptions which are associated with the related spatial objects. But as a remark RCC layer is invisible from the Abox's view.

5.3 Racer System with nRQL

As the last step for nRQL, rough information about Racer System which uses the nRQL as a query language is given. Racer System is the description knowledge reasoning system which uses the SHIQ Logic. This logic has some parts from the OWL DL. Racer systems also support the multiple TBoxes and ABoxes with the newly added feature "concrete domains". Concepts and the roles are the main arguments that Racer systems use for the entities with formulated constraints.
Different knowledge bases formats are supported like OWL interface, RDF and XML. Besides these KB's several clients are available for the facilities of Racer System like RICE and the protégé. Predefined queries and the functionalities are provided to query the schema by the Racer. Racer is also used to prove facilities for the given instances. As a last remark Racer does not only query the instances but also it allows to reason.

6 Probabilistic Extension For nRQL

In this part of the project, the overview of the availability of implementing the probabilistic extension for nRQL is given. But before that some of the topics that are already mentioned in this project, would like to be pointed out again where they can be useful to see the vague scene a little bit plain to the eyes. It is mentioned [See: 63 page 1] that nRQL can offer the reasoning services for multiple ABoxes and TBoxes. Besides that nRQL implements the description as SHIQ which is the basic logic of ALC and supports OWL-DL almost completely. Maybe now it is not clear why these three features of nRQL are mentioned again. But later on the pieces are going to launch a total scene for the availability of implementing probabilistic extension in the future works.

In nRQL there are some problems where one of them is the individuals in the class expressions (so-called nominals), by which can be processed in RacerPro using so-called ABoxes [See: 63 page 3]. But they are going to be the instances of a concept with the same name [See: 63 page 50].

Second problem is the inability of RacerPro to process the user-defined datatypes given in external XML schema specifications. Although all required datatypes of OWL DL are supported.

So one of the strongest solutions of implementing the probabilistic extension can go through the way of ABoxes with probabilistic extension which is more efficient and is easy to compute by the help of some paperworks [See: 15, 30].

The general idea of the paperwork [See: 15] is about ABoxes, but this time ABoxes have other assertions which make them different from the others.

Also in the paperwork [See: 15] PALC as you guess “p” stands for the word “probabilistic”, is mentioned. The ALC is just extended (Description logic which is implemented by nRQL) with the probabilistic assertions. So this study gives another opportunity to have the goal. In another way, the probabilistic assertions are stated about the extension of concepts and roles. In [See: 15], details about the implementation and terms onto ABoxes by figuring corollaries, examples and lemmas are declared by the researchers. After the overview of this paperwork, it is easy to have the confidence to say the possibility of having the probabilistic ABox reasoning with the reduction of the solution space of the reasoning problem. Through the way of this paperwork PALC is declared as the DL-framework with probabilistic ABoxes. If nRQL can handle probabilistic ABoxes instead of ABoxes, all the instances of the given domain which are represented by ABoxes that have the probabilities can be used for the query tuple. Again some restrictions are occurred; one of them is the feature of nRQL in which probabilistic ABoxes can be handled. If the semantics and the syntax of the nRQL [See: Chapter 5 in this volume, 39, 56 page 6-9] can handle the newly created ABox, so there will not be any problem to use the probabilistic ABoxes with nRQL. Again the overview of the availability of implementing the probabilistic extension through another way by using the probabilistic ABox can be seen, but still the way of implementing, procedures, corollaries or the lemmas are not situated for nRQL. It is just the research that matches the published, ongoing projects and also the knowledge about the related topics.

Racer can treat an ABox as a database and rewrites queries in order to support TBOX information for the information retrieval [See: 39].

If ABox can be exchanged (database) with the probabilistic one, the database with the weighting of the instances that are situated in the related ABox can be created. The idea is just coming from the relations, if the probability to implement the weighting of the object by using PALC through the ABox can be satisfied, maybe nRQL can use the newly created ABox with the semantic of probability for the TBox queries. Because the language nRQL augments and extends Racer's functional API for querying a knowledge base [See: 39 page 1]. Maybe using PALC instead ALC may lead to have undecidable inference problems like the one that we have with ALC [See: 38 page 6].

In the transaction of the queries, the variables are employed by the active semantics where the variables are held by the ABox individuals in the related ABoxes [See: 63 page 58]. So the semantic of nRQL should have to handle the new semantic of ABox with the probabilistic extension. ABox augmentation rules are offered by nRQL to add new ABox assertions to the already existent ABox. So by the rules, the probabilistic assertion onto existent ABoxes can be applied. If the rules of applying the probabilistic assertion are fired that means that it is applicable and then these sets of rules are going to be added to the ABox [See: 63 page 96]. But nRQL isn't responsible for checking the rules; these are under the responsibilities of the users or the programmers. If a successful implementation of probabilistic ABox can be implemented onto nRQL, the completeness of the modelling in an ABox is checked by NEG [See: 63 page 58].

On the other side of this thought the observation of nRQL is extremely need. What can be the way of implementing the ABox in such a way that handles these kinds of implementation situations?

The answer can be given like that;

The probabilistic extension for nRQL needs some restrictions, because it is not possible to use the probabilistic weights directly for Abox. In nRQL the correctness of all ABOXES is “1”. The first problem is to mention the probabilistic weightings with ABOXES. Also pDATALOG allows for recursive rules, it provides powerful inference.

To apply the probabilistic extension, rules have to be declared by nRQL, but creating rules by nRQL can not yield any problems because of being non-recursive. So the first restriction is the recursive rules. Because if the query depends on the recursive rules, it is not possible to solve the query from top to the bottom, it yields the user to have inconsistencies and incorrect results. But one way of computing the recursive rules is starting from bottom to the top; it still causes problems because for each given links or nodes, the derived facts and their related nodes have to be searched where it is not possible for a big decision tree.

For the implementation the most convenient environment is declared and it just works under the given restrictions.

So for combining the probabilistic extension (pDATALOG) with nRQL, two basic literals one for pDATALOG and one for ABox are needed. The binding between these two literals can be done with the similar idea that is mentioned in [See: 30].

In pDATALOG with ground facts also a probabilistic weighting can be given [See 30, page 2].

0.7 indterm (d1, ir). 0.8 indterm (d1, db)

It shows that document “d1” is about IR with probability 0.7 and about DB with probability 0.8. Retrieving documents is done by means of rules and for our goal of project; the rules are used as the bindings between the two literals. So the probability which comes from ABox with “1” will combine with the probabilistic weight which comes from pDATALOG literal.

q1(X):- indterm(X, ir) & indterm (X, db).

D1 fulfills predicate q1 with a certain probability for the given example.

The index terms are assumed independent so that the computation of the probabilistic AND-combination and also the OR-combination is produced by the rules.

q2(X):- indterm(X, ir).

q2(X):- indterm(X, db).

In hypertext structures where we have directed links between single documents or nodes, pDATALOG *rules* can be used for performing retrievals.

Assume that these links have probabilistic weights;

0.5 link (d2, d1). 0.4 link (d3, d2).

The idea of the given example: If we have a link from D1 to D2, and D2 is about a certain topic, then there is a certain probability that D1 is about the same topic. This probability is specified by the weight of the link predicate. Now we can formulate the rules.

about (D, T):- indterm (D, T).

about (D, T):- link (D, D1) & about (D1, T).

Up to this point the examples and the idea of the reasoning are taken from the paperwork [See: 30].

This is the similar scenario for probabilistic extension for nRQL. The two literals (ABox, pDATALOG), which can be assumed to be hung up in the space, can be combined with a binding (like the link predicate) under some restrictions that will yield the correct result after the computation. But this probabilistic extension can not be implemented for the recursive rules which can not overcome with the inconsistencies. Willing to apply the probabilistic theory for the given scenario leads the problems which are occurred because of the repetition of the probability of the given links.

So for the project, the minimization is preferred in order to implement the probabilistic extension for nRQL, because without any minimization it is not possible to implement pDATALOG onto nRQL via ABoxes. This is called as “minimum model”. By creating new concepts or models, the prototype of implementation can be done with some restrictions and assumptions. In paperwork [See: 30] the researchers are called the minimum model computed the perfect Herbrand model in order to reach their target. The seeds of the idea for the availability of implementation of the probabilistic extension for nRQL come from the mentioned model. For more details [See: 30, page 5].

As a short summary of the implementation, the direct-linked nodes are taken under consideration because of avoiding recursion. The binding rules are generated by nRQL which is in safe mode with non-recursive. The probabilistic weight is computed by the two literals which are ABoxes and pDATALOG. So in a sense that one branch of the whole tree is taken and tried to implement all the necessary concepts with some restrictions onto it. This part can just be applicable for the given scenarios and can not be implemented for the whole nRQL without any restrictions or given model.

The rest of this chapter is just about an idea about probabilistic extension via OWL-DL for rule languages. This part is just being a light for the coming projects in which the probabilistic extension for rule language (Can be nRQL) would like to be implemented via OWL-DL. This research project can be done for PhD thesis because of the necessity of a long time and a brief study for the related topics.

Another research that can light the idea is [See: 61 page 18]. This paperwork announces the ongoing project about the topic that can be very important for the future paperworks. In this paperwork , it is situated that currently the researchers are combining the probabilistic datalog (pDatalog) with OWL-DL so that complex ontologies can be described , Then the new pOWL-DL can be combined (the way that I named), with their ontology matching method with so-called schema matching methods. Maybe the ways of procedures, corollaries and lemmas can be similar with the ones that are done by the paperwork [See: 34].

The implementation of the probabilistic datalog rules onto OWL-DL is another problem. Sometimes it is not possible to implement the probabilistic datalog rules directly onto the web ontologies. Because of being an ongoing paperwork, the way of solving problem is not established yet. But a similar problem with the probabilistic datalog rules and DAML-OIL is solved in the paper [See: 33 page 27-49]. Direct implementation is not possible so the researchers use the XSLT stylesheet to implement the rules onto DAML-OIL. Then the DAML-OIL for the documents in XML format is converted into documents in standard schema. Always there can be a way of overcoming the difficulties if enough research and the knowledge about the concepts and the sub-concepts which can be solution of the real problem in any way, are known.

So this section of Chapter 6 after explaining the implementation of pDATALOG for nRQL via ABoxes has just the overview of the possibility of implementing probabilistic extension onto rule languages via OWL-DL. This part doesn't show the semantics or the syntaxes of implementing pDatalog onto rule languages. It concerns the several projects, paperworks and the results that can be the solution of the future paperworks in which the OWL-DL is wanted to be used as a bridge for combining the probabilistic extension for rule languages

7 Conclusion

This part concerns the last thought and the result of the project. From the starting point of the project the first aim is to find out the related documentation and paperwork for the efficiency of the project. The most important point of the written project is the mentality of the project; this project does not concern the whole idea of explaining the way of implementing probabilistic datalog onto nRQL. The aim of the project is to declare the related concepts in details, to match the similarities by the help of the ongoing or the finished paperworks in order to create the part of scenario which can be adapted for the conclusion of the project work. Also on the way through the study the most difficult and the most time taking part is to find the documentations, because under these topics, it is not easy to find documentations which show exactly what we are looking for. So as a solution to figure out the data that we are looking for, we found out the similar paperworks. After a deeper understanding of the given paperworks, the similarities and the differences are studied in details. Then by using the knowledge about the concepts, the availability of the thoughts are tried to be launched through the conditions and the restrictions in which they are assumed to be satisfied for all the way through the study. So this project is a good researches paper for the given concepts, for each concept the important syntaxes and semantics are declared in a very good and understandable way. For the description of syntaxes and semantics, the reader is assumed to be familiar with topics.

Also all the references and the documents which are used to give lights to this project paperwork are situated. Under the given conditions the availability of the implementing our goal has been tried to explain in this project work.

The result of the project shows that in the near feature the probabilistic datalog can be golden key for the other related concepts in description logic. Also the applications are trying to adapt themselves to this part of the science which the reasoning can be done by the probabilistic datalog. All the way through the project while searching for resources, too many EU framework projects, in which the scope of the projects are implementing the probabilistic datalog onto some other language in somewhat ways are founded. This is another sign that the way implementing probabilistic datalog onto some cases is not an easy job and there are lots of semantics, syntaxes and the rules to be taken under notation. So under all these hard concepts, my thoughts would like to be explained in a simple and a clear way. Hopefully the creation of a good research paper for the given project is presented.

As the last sentences of this part, I would like to figure out that this field of the science is not well defined and not studied for a long time. The history is too short to have unlimited resources and paperworks. Nevertheless future will be the solution of the ongoing projects and new thoughts would have their answers.

8 References

1. A. Paskin Mark. Maximum Entropy Probabilistic Logic. University of California, Berkeley.
2. Baader F., Calvanese D., McGuinness D., Nardi D. & Patel-Schneider P., editors (2003). The Description Logic Handbook. Cambridge, England: Cambridge University Press.
3. Boley Harold, Dean Mike, Grosof Benjamin, Kifer Michael, Tabet Said & Wagner Gerd. (2005, April 27-28). RuleML Overview and Position Statement. Position Paper [96], W3C Workshop.
4. Boley Harold, Dean Mike, Grosof Benjamin, Sintek Michael, Spencer Bruce, Tabet Said & Wagner Gerd. (2005, April 11). W3C Member Submission. Available from http://www.w3.org/Submission/2005/SUBM-FOL-RuleML-20050411.
5. Crestani F., Lalmas M., Van Rijsbergen C. J. & Campbell I. (1998). A Survey of Probabilistic Models in Information Retrieval. ACM Computing Surveys 30(4): 528–552. Available from http://www.acm.org/pubs/citations/journals/surveys/1998-30-4/p528-crestani.
6. Crestani Fabio & Rölleke Thomas. Issues on the Implementation of General Imaging on Top of Probabilistic Datalog. Padova, Italy: University of Padova, Dortmund , Germany: University of Dortmund.
7. Cullot Nadine, Spaccapietra Stefano, Vangenot Christelle & Parent Christine. Ontologies: A contribution to the DL/DB debate. University of Burgundy, France. Swiss Federal Institute of Technology, Switzerland. University of Lausanne, Switzerland.

8. Dean Mike & Schreiber Guus, editors. (2004, February 10).*OWL Web Ontology Language Reference*.W3C Recommendation.
9. De Bruijn Jos. (2005, May 2).Description Logics for the Semantic Web (Lecture 6).Innsbruck, Austria: University of Innsbruck.
10. De Bruijn Jos, Mart´ın-Recuerda Francisco, Ehrig Marc, Polleres Axel & Predoiu Livia, editors. (2005, February 8). Ontology Mediation Management V1. Deliverable D4.4.1 (WP4), SEKT EU-IST-2003-506826 (SEKT: Semantically Enabled Knowledge Technologies). Digital Enterprise Research Institute, University of Innsbruck.
11. De Bruijn Jos, Foxvog Douglas, Lausen Holger , Oren Eyal, Roman Dumitruand & Fensel Dieter.(2004). The WSML Family of Representation Languages. Deliverable D16v0.2, WSML, http://www.wsmo.org/wsml. Available from http://www.wsmo.org/2004/d16/v0.2/.
12. Dekhtyar Alexander. Probabilistic Information Retrieval Part I: Survey. University of Maryland.
13. Ding Zhongli & Peng Yun. (2004). A Probabilistic Extension to Ontology Language OWL. Department of Computer Science and Electrical Engineering, University of Maryland Baltimore County.
14. Dr Bryant. Introduction to OWL. Available from http:// www.comp.rgu.ac.uk/staff/chb/teach.html.
15. Dürig Michael & Studer Thomas. Probabilistic ABOX Reasoning. University of Bern, Switzerland.
16. F. Patel-Schneider Peter. (2001, January 11). A Model-Theoretic Semantics for DAML+OIL.

17. F. Patel-Schneider Peter. (2004, November 2).A Proposal for a SWRL Extension to First-Order Logic.
18. F. Patel-Schneider Peter. (2005, April 11). A Proposal for a SWRL Extension towards First-Order Logic.W3C Member Submission. Available from http://www.w3.org/Submission/2005/SUBM-SWRL-FOL-20050411.
19. Franconi Enrico. Description Logics: Foundations of First Order Logic. Department of Computer Science, University of Manchester. Available from http://www.cs.man.ac.uk/~franconi.
20. Franconi Enrico. Description Logics: Using First Order Logic. Department of Computer Science, University of Manchester. Available from http://www.cs.man.ac.uk/~franconi.
21. Franconi Enrico. Description Logics: Introductory Lecture. Department of Computer Science, University of Manchester. Available from http://www.cs.man.ac.uk/~franconi.
22. Franconi Enrico. Description Logics: Foundations of Propositional Logic. Department of Computer Science, University of Manchester. Available from http://www.cs.man.ac.uk/~franconi.
23. Franconi Enrico. Description Logics: Propositional Description Logic. Department of Computer Science, University of Manchester. Available from http://www.cs.man.ac.uk/~franconi.
24. Franconi Enrico. Description Logics: Structural Description Logics. Department of Computer Science, University of Manchester. Available from http://www.cs.man.ac.uk/~franconi.

25. Franconi Enrico. Description Logics: Logics and Ontologies. Department of Computer Science, University of Manchester. Available from http://www.cs.man.ac.uk/~franconi.
26. Fensel Dieter. (2004, June 14).A rule language for the semantic web.
27. Fikes Richard. & McGuinness D. L. . An axiomatic semantics for RDF, RDF schema, and DAML+OIL.Technical report. KSL-01-01. Available from http://www.ksl.stanford.edu/people/dlm/daml-semantics/abstract-axiomati%c-semantics.html.
28. Fikes Richard & McGuinness D. L., editors.(2001, March). An Axiomatic Semantics for RDF, RDF-S, and DAML+OIL.W3C Note. Available from http://www.w3.org/TR/2001/NOTE-daml+oil-axioms-20011218.
29. Fuhr Norbert. (1999, April 29). Probabilistic Datalog: Implementing Logical Information Retrieval for Advanced Applications. Dortmund, Germany: University of Dortmund.
30. Fuhr Norbert. Probabilistic Datalog: A Logic for Powerful Retrieval Methods. Dortmund, Germany: University of Dortmund.
31. Fuhr Norbert.XIRQL: Eine Anfragesprache für Information Retrieval in XML-Dokumenten.Dortmund,Germany:University of Dortmund.
32. Fuhr Norbert & Nottelmann Henrik. Combining DAML+OIL, XSLT and probabilistic logics for uncertain schema mappings in MIND. Duisburg, Germany: University of Duisburg-Essen.
33. Fuhr Norbert & Nottelmann Henrik.(2003, August 19). Combining DAML+OIL, XSLT and probabilistic logics for uncertain schema mappings in MIND (ECDL). Trondheim, Norway

34. Fuhr Norbert & Nottelmann Henrik. IR Models based on predicate logic. Duisburg, Germany: University of Duisburg-Essen.
35. Fuhr Norbert & Nottelmann Henrik. pDAML+OIL: A probabilistic extension to DAML+OIL based on probabilistic Datalog. Duisburg, Germany: University of Duisburg-Essen.
36. Fuhr Norbert & Nottelmann Henrik (2003, December 16). Probabilistic logics for defining and using P2P service descriptions. MMGPS Workshop. London, UK.
37. Haarslev Volker. (2005). An Introduction to Description Logics. Department of Computer Science and Software Engineering Concordia University, Canada.
38. Haarslev Volker & Möller Ralf. (2001). Description of the RACER system and its applications. Hamburg-Harburg, Germany: Hamburg University of Technology (TUHH).
39. Haarslev Volker[1], Möller Ralf[2] & Wessel Michael[2]. (2004). Querying the semantic web with Racer + nRQL. [1] Montreal, Canada: Concordia University, [2] Hamburg-Harburg, Germany: Hamburg University of Technology (TUHH).
40. Harth Andreas & Decker Stefan. (2004, November 23).OWL Lite- Reasoning with Rules. Working Draft. WSML. Available from http://www.wsmo.org/2004/d20/d20.2/v0.1/20041123.
41. Heflin Jeff. Knowledge Representation Issues for the Semantic Web. Lehigh University.
42. Hitzler Pascal, Angele Jurgen, Motik Boris & Studer Rudi. Bridging the Paradigm Gap with Rules for OWL.

43. Horrocks Ian, Grosof B. N., Decker S. & Volz R. (2003). Description logic programs: combining logic programs with description logic. In Proc. of the twelfth international conference on World Wide Web.
44. Horrocks Ian, Grosof B. N., Patel-Schneider Peter F., Boley Harold, Tabet Said, & Dean Mike. (2003). SWRL: A semantic web rule language combining
45. OWL and RuleML. Daml draft v0.5, DAML. Available from http://www.daml.org/2003/11/swrl/.
46. Horrocks Ian , Patel-Schneider Peter F. & Bechhofer Sean (2003, October 20).Tutorial on OWL. ISWC, Sanibel Island, Florida, USA.
47. Horrocks Ian, Patel-Schneider Peter F., & Hayes Patrick (2004, February 10). OWL web ontology language semantics and abstract syntax. Recommendation, W3C.
48. Horrocks Ian, Patel-Schneider Peter F., Parsia Bijan & Hendler James. Semantic Web Architecture: Stack or Two Towers.
49. Horrocks Ian & Z. Pan Jeff . (2004). OWL-E: Extending OWL with expressive datatype expressions. IMG Technical Report IMG/2004/KR-SW-01/v1.0, Victoria University of Manchester. Available from http://dl-web.man.ac.uk/Doc/IMGTR-OWLE.pdf.
50. J. Rapaport William. Holism, Conceptual-Role Semantics, and Syntactic Semantics. New York, USA: State University of New York.
51. Kersting Kristian. Representational Power of Probabilistic-Logical Models: From Upgrading to Downgrading. Freiburg, Germany: Albert-Ludwigs University of Freiburg.
52. L. Costello Roger & B. Jacobs David. (2003). A Quick Introduction to OWL Web Ontology Language. The MITRE Corporation.

53. Liebig Thorsten & Noppens Olaf. Combining Browsing and Editing with Reasoning and Explaining for OWL Lite Ontologies. Ulm, Germany: University of ULM.
54. M. Jaeger. (1994). Probabilistic reasoning in terminological logics. In Proc. of Knowledge Representation-94, pages 305-316.
55. McGuinness D. L. & Van Harmelen Frank. (2002, July 29).Feature Synopsis for OWL Lite and OWL.W3C Working Draft. Available from http://www.w3.org/TR/2002/WD-owl-features-20020729.
56. Moeller Ralf & Wessel Michael. (2005). A high performance semantic web query answering engine. Hamburg-Harburg, Germany: Hamburg University of Technology (TUHH).
57. Motik Boris, Sattler Ulrike, and Studer Rudi.(2004).Adding DL-safe rules to OWL DL.
58. N. Nilsson. (1986). Probabilistic logic. AI, 28:71-87.
59. Nardi Daniele & J. Brachman Ronald. An Introduction to Description Logics.
60. Nottelmann Henrik. MIND: An architecture for multimedia information retrieval in federated digital libraries. Dortmund, Germany: University of Dortmund.
61. Nottelmann[1] Henrik & Straccia[2] Umberto.(2004, November 9). A probabilistic, logic-based framework for automated ontology matching.[1] Duisburg, Germany: Institute of Informatics and Interactive Systems, University of Duisburg-Essen, [2]ISTI-CNR, Pisa, Italy.
62. R. Giugno, T. Lukasiewicz. (2002). P-SHOQ (D): A probabilistic extension of SHOQ (D) for probabilistic ontologies in the semantic web. Technical report.
63. Racer Systems GmbH & Co. KG. (2005, April 14). RacerPro User's Guide Version 1.8. Available from http://www.racer-systems.com.

64. Racer Systems GmbH & Co. KG. (2005, May 30).RacerPro Reference Manual Version 1.8.
65. Seipel Dietmar. An Efficient Computation of the Extended Generalized Closed World Assumption by Support-for-Negation Sets. Tübingen, Germany: University of Tübingen.
66. Text Retrieval and Mining
67. U. Straccia . Uncertainty and Description Logic Programs: A Proposal for Expressing Rules and Uncertainty on Top of Ontologies. Italy.
68. U. Straccia. (2004). Uncertainty in description logics. In Proc. of IPMU 04, pages 251-258.
69. Von Fintel Kai & Heim Irene. Lecture Notes on Intensional Semantics. Massachusetts Institute of Technology.
70. Vrandecic Denny, Haase Peter, Hitzler Pascal, Sure York & Studer Rudi. DLP – An introduction. Karlsruhe, Germany: University of Karlsruhe.
71. W3C. (1999, February). Resource description framework (RDF) model and syntax specification.Technical report, World Wide Web Consortium. Available from http://www.w3.org/TR/1999/RC-rdf-syntax-19990222/.
72. W3C. (2001, March).DAML+OIL. Technical report, World Wide Web Consortium. Available from http://www.w3.org/TR/daml+oil-reference.
73. W3C. (2002, April). RDF vocabulary description language 1.0: RDF Schema. Technical report, World Wide Web Consortium. Available from http://www.w3.org/TR/2004/RC-rdf-schema-20040210/.
74. W3C. (2004). OWL. Technical report, World Wide Web Consortium. Available from http://www.w3.org/TR/2004/REC-owl-features-20040210/.
75. Web Ontology Language (OWL). (2003, June 18). Neuchâtel University.

76. X. Li, P. Morie & D. Roth. (2005). Semantic integration in text: From ambiguous names to identifiable entities. *AI Magazine. Special Issue on Semantic Integration*.
77. Z. Pan Jeff. Requirements for a Semantic Web Rule Language. Manchester, UK: University of Manchester.