Extending SWRL Rules to Integrate Object-Oriented Techniques in Hybrid Ontologies

Souad Bouaicha, Zizette Boufaida
Department of Computer Science
LIRE Laboratory
Mentouri University
Constantine, Algeria
{bouaicha.souad, zboufaida}@gmail.com

ABSTRACT: *Scalability of ontology reasoning is a key factor in the practical adoption of ontology technologies. To improve the quality of reasoning on a simple ontology, we propose to enrich the comportment of the ontology concepts by rules. This enrichment is done by extending a version of SWRL and its execution in Jess.*

The extending of SWRL rules consists of adding new keywords as OWL ontology classes and properties, and post-translating them by using rewriting meta-rules. These rules adapt the technique of Object-Oriented programming and create a new hybrid ontology.

The reasoning on this hybrid ontology makes it possible to improve in terms of results the quality of the previous reasoning. To mention and clarify the importance of this kind of reasoning, we use a case study inherent to interpret a check up in preventive medicine.

Keywords: Hybrid Ontologies, SWRL rules, Rules/Concepts Integration, Reasoning

Received: 12 June 2012, Revised 5 August 2012, Accepted 9 August 2012

© 2012 DLINE. All rights reserved

1. Introduction

While reasoning over OWL ontologies is a provably intractable computational problem, it has been observed that the ontologies encountered in practice rarely involve a combination of constructs that leads to intractability. By relying on sophisticated optimizations, reasoners were developed in which they can handle ontologies with large Tboxes, yet these still do not provide adequate performance on ontologies containing large ABoxes. This has so far prevented the usage of OWL in applications that depend on large data sets, such as metadata management and information integration.

Parallel to the development of reasoning techniques for OWL [10], recent work in semantic Web has concentrated on the addition of rules to OWL which provides an additional layer of expressivity [3].

In some applications, rules are consecrated to a specific task, which can be achieved independently of the ontology. In these

cases, it is possible to use two distinct languages with specific inference engines. But other applications, where rules are used to extend the expressivity of OWL, require the reasoning on rules in conjunction with ontology. Although the combination of OWL DL and the rules are uncertain [1, 2]

The ultimate goal of this work is the development of hybrid ontology which reuses techniques of Orient-Object Programming that could be automatically converted into various rule engine formats, and executed directly. That's why; we propose an approach to enrich the comportment (behavior) of the ontology concepts with rules. This enrichment is done through two manners. 1) Binding attributes and roles of concepts with conditions. This aspect constitutes the major contribution of this work. 2) Creating rules to combine individuals and atoms of concepts. The latter is found, in a large scale, in literature.

This internal and external enrichment of the ontology concepts gives rise to hybrid ontologies, which permits the amelioration of results of the reasoning. To clarify the importance of such reasoning, we apply a case study inherent to interpret a check up in preventive medicine, using standard and current tools of semantic Web.

2. Related Work

In literature, there are two main approaches to integrate OWL language withe the rules: the homogeneous approach and the hybrid one.

2.1 Homogeneous Approach

The homogeneous approach treats the predicates of the rules and the ontologies homogeneously, to create a new logical and simple language. The general idea is that the rules can use unary and binary predicates of the ontology as well as the predicates of the rules. Thus, a new reasoner is necessary, able to handle and manipulate the new homogeneous language emerged [4].

2.2 Hybrid Approach

The hybrid combination follows a distinct architecture of two subsets in which, each one treats a part of the knowledge base. More specifically, it combines the possibilities of a reasoner on description logics and the possibilities of executing rules of an inference engine in order to define rules on the ontology layer. [5].

Thus in the hybrid approach the ontology remains unchanged and rules are built on top of ontologies. This makes possible the integration of existing rule reasoner with ontology reasoner for reasoning in the hybrid language, rather than developing a new reasoned from scratch.

2.2.1 Some Examples of Hybrid Systems

• AL-log: The AL-log language described in [12] is a hybrid integration of Datalog and the Description Logic ALC. The ALC DL is a simple Description Logic admitting only the following class constructors. The DL queries in the bodies of the AL-log rules are restricted to (ground or open) class-instance membership queries; property instance membership queries are not allowed. Moreover, the variables of the DL queries in the body of an AL-log rule must also appear in the non-DL atoms of the body or in the head. Thus the DL queries are typing constraints on the variables.

• CARIN: Is defined as a family of languages with the intention to provide a hybrid integration of Datalog with different description logics. The class of DL logics considered includes any subset of the description logic ALCNR, and the intersection constructor for properties. Thus CARIN extends AL-log by integrating Datalog with more expressive DL and by admitting more general DL queries in the bodies of the hybrid rules. [13]

• Integrating Answer Set Programming with DL: Proposals to integrate function-free rules with answer set semantics and Description Logics are presented by several authors. The paper [14] uses DL-queries of the ALC description logic as constraints in rule bodies. The rules allow disjunctive heads. In particular, the work [15] presents semantics with inûnite open domains which is capable of dealing with an interesting fragment of OWL DL extended with rules. Again, this is a homogeneous system.[14]

• DLEjena [6] is inspired from the hybrid and homogeneous approach for the integration of the rules and ontologies. The architecture of DLEJena has four modules: the first charges the ontology and separates its terminological and assertional parts. The second reasons on the Tbox part, the third reasons on the Abox part and the last one reason with the help of the applied rules to Tbox and Abox parts.

• In [7], the plugin of SWRLJessTab is illustrated on the Family ontology rules to show how certain reasoning could be provided for interoperability between SWRL and OWL.

• In [8], the symbolic knowledge is represented in ontology of the cortical structures. For the representation, standard languages of the semantic Web (OWL, SWRL) are used. In order to enrich ontology, it is extended by Horn clauses. These rules permit to propagate relations and to infer new facts from those existing.

• Alloy [9] is a system for reasoning on ontology written in OWL and SWRL or SWRL-Fol. It is based on the transformation of OWL ontology and SWRL rules in a program called Alloy's program and its analysis starts from a charged model to verify automatically its uniformity.

2.3 Discussion

The study of the previous work shows that there are two different fundamental approaches. The first one is homogeneous where the rules and the individuals of the ontology are compiled without separation between them. This leads to a base of facts compiled in the same inference engine [9]. The hybrid systems are the second approach. They are made of several subsets; each of them treats a different part of the knowledge base and use new formalisms of representation and specific procedures of reasoning [6, 7 and 8]. In such systems, the problem of the complementary in answers is asked. Really, if the inference is executed separately with OWL reasoner and an inference engine, some inferences are evidently missed.

Therefore, the interoperability between the SWRL rules and the ontology requires a close integration.

In this paper, we are interested to complete answers of a hybrid system by proposing an approach that allows enriching the concepts comportment. This enrichment is effected via rules reasoning on the attributes. Concretely, we propose a complete process of creating a rule base to the terminological part of ontology and whose steps are detailed in the following sections.

2.4 Limitations of SWRL

The current SWRL 1.0 specification omits some typical rule language constructs, sacrificing some expressiveness to ensure decidability and/or efficiency. We have previously identified some limitations and workarounds in writing teaching strategies in SWRL [7]; in this work, we directly address these limitations.

2.4.1 Flat list

A SWRL rule body (or head) is a flat list of atoms. No block structure is supported (as there are no constructs that would need them). Note that either of #E or #F would also entail adding block structure.

2.4.2 E. Conjunction only

Disjunction (E1) and negation (E2) are excluded.

2.4.3 F. No quantifiers

Explicit universal and existential quantifiers are excluded. All atoms are implicitly universally quantified.

2.4.4 No user-defined functions

SWRL provides a library of built-in functions for primitive math, string, and date operations. It doesn't allow an external function to be defined or called (which fails 1.1#C).

2.4.5 Assertions only

A SWRL rule head "*makes*" its atoms true; the standard implementation is to add these atoms as new facts to the knowledge base. Existing knowledge cannot be changed. [11]

3. Process of Creation of Intra-Concept Rule Base

Our contribution consists of proposing a process to create these rules (cf. Figure 1).

In the following sections, we detail the various steps of our process presented in figure 1.

3.1 Needs Specification and Conceptualization Steps

These steps are guided by experts. They lead to a structured document which contains concepts and attributes as well as the relations between them. After the revelation of the various experts, this part makes up of a whole of phases making it possible

to lead to an intermediate, semi-formal representation (RIs) of a rule.

3.2 Formalization Step

A complete formal and operational representation, from the conceptual representation of rules is obtained while passing by another semi-formal one which play the role of intermediary. The rules are defined in the following form:

$$R: b_1 \cap b_1 \cap .....b_n \rightarrow a \quad (1)$$

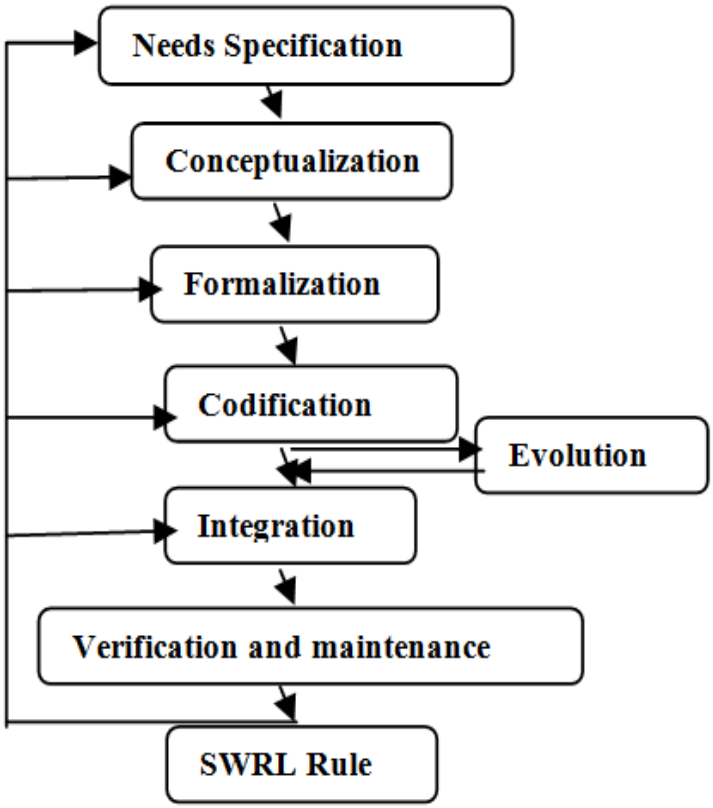

Figure 1. Process of creation of intra-concept rule base

They are composed of two parts:

• The antecedent ($b_1 \ldots b_n$) is the conjunction of atoms. An atom can be an attribute, a role, or built-ins. In this study, we consider the cases where the atoms are attributes or built-ins.

• The result of the conjunction of the atoms exists in the conclusion part. (a)The latter gives a result which corresponds to a chosen rules type.

Let D, OP, DP, builtIn be respectively a datatype URI or an enumerated value range, object property, datatype property and built-In relation. Let *i*, *j* be object terms, and v, v_1, ..., v_n be datatype terms (data literals or data value ranging variables) and let *p* be a built-in name. Then, the set of **atoms** is defined by the following grammar:

$$Atom \rightarrow D(v) \mid OP(i, j) \mid \mid DP(i, v) \mid builtIn(p, v_1, ..., v_n) \quad (2)$$

The writing of a rule initially obliges us first to choose its type. This last is connected to its objective:

• Calculate the value of a property based on other properties: can be used to initialize new instances with default values of properties.

• Check constraints and perform data validation: for example, need to automatically raise inconsistency flags when currently available information does not fit the specified integrity constraints.

The result of this step is the formal representation of the rules. To express the passage of the conceptual model, (described in the preceding step), to a formalized rule, the conjunction between the atoms is represented by an intersection between them and the conditions on values of an attribute (≥/ ≤ /=..) are represented by a built-in.

3.3 Codification Step

This step consists of translating the result of the previous phase in a language of definition of rules. For this reason we choose to extend SWRL by rules. To overcome this, we devise a generic approach of extending SWRL rules by adding new keywords as OWL ontology classes and properties and post-translating them using rewrite meta-rules. The resulting "*extended SWRL*" language is sufficiently expressive to formulate the new type of rules which adapt the techniques of the Object-Oriented Programming, as shown in the following (cf. Figure 2) :

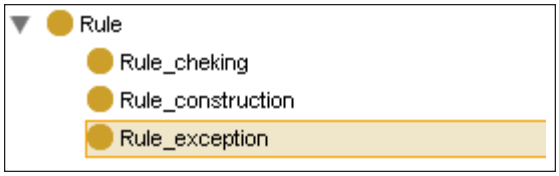

Figure 2. Keywords extension

For example, the RuleConstruction class derives new values from other properties of the same concept. We use also the SQWRL language, because of its semantic functionalities which are richer than those of SWRL.

The result of this step is a RDF document, made up of a SWRL representation which corresponds to the structure of the rule and a RDF part which provides mechanisms giving access to the individuals.

3.3.1 Extending SWRL rules (keywords extension)

We propose a generic approach for adding new keywords as OWL ontology classes and properties. We use the following simple java code (cf. Figure 3) to create the new concepts (keywords).

After the creation of new concepts, we use there in the SWRL rules then convert them to a Jess format using XSLT and SweetRules. Apply meta-rules to rewrite the keywords before evaluating the rules in Jess.

3.3.2 Rule Rewriting in Jess

Once the rules are in Jess format, we apply our own meta-rules to rewrite those rules that contain any of our keywords. Jess (equivalently, Java) has a Lisp-like ability to treat unevaluated rules as data, manipulate them using list and string operations, and evaluate a list or string as a rule. An unevaluated Jess rule has a tree structure. We represent a tree-of-tokens as a Vector-tree of String, using standard Java classes, i.e. the elements of each Vector are either String for a leaf token, or recursively a Vector-tree for a sub list. We define a rewriting meta-rule for each keyword, as a tree rewriting operation, which examines a rule's Vector-tree, modifies any elements, splices any new sub trees into place, and so on. Each meta-rule deletes its own keyword from the rule or replaces it with the corresponding Jess keyword.

Keyword matching is done with respect to SweetRules' output. Our **RuleConstruction** keyword is an OWL class, so SWRL deems it to be a class membership test. SweetRules, being agnostic to our keywords, convert **RuleConstruction** to the Jess rule pattern for a class membership test.

The rule rewriting step uses a simple iteration that applies every meta-rule to every Jess rule string. Translated rule strings that have changed are then re-evaluated in Jess.

3.4 Evolution Step

This step consists of following the evolution of the rule. After each modification, the mechanism of reasoning on ontology enables us to ensure their integration in the ontology, and their execution on the chosen inference engine.

```
package com.yoshtec.owl.testclasses.NewKeywords;

import com.yoshtec.owl.annotations.OwlDataProperty;
import com.yoshtec.owl.annotations.OwlDataType;
import com.yoshtec.owl.annotations.OwlIndividualId;
import com.yoshtec.owl.annotations.OwlObjectProperty;
import com.yoshtec.owl.annotations.oprop.OwlInverseObjectProperty;

public class ruleConstruction  {

    @OwlDataProperty(uri="http://www.yoshtec.com/ontology/test/ruleConstruct#attribut")
    @OwlDataType(uri="http://www.w3.org/2001/XMLSchema#string")
    private String attribut = null;

    @OwlDataProperty(uri="http://www.yoshtec.com/ontology/test/ruleConstruct#role")
    @OwlDataType(uri="http://www.w3.org/2001/XMLSchema#int")
    private Integer role = null;

    @OwlObjectProperty(uri="http://www.yoshtec.com/ontology/test/ruleConstruct#Contained_in")
    @OwlDataType(uri="http://www.yoshtec.com/ontology/test/ruleConstruct#Patient")
    private Matryoshka contained_in = null;


    @OwlIndividualId
    private String name = null;

    public ruleConstructImpl(){

    }

    public ruleConstructImpl(String name){
        this.name = name;
```

Figure 3. Java code corresponding to a Rule Construction

3.5 Integration Step

This step is effected by the arrangement of RDF code corresponding to the intra-concept rules in the hierarchy of ontology.

3.6 Checking Step

Here RACER reasoner is used to exploit the services of inference which it provides. An inference engine of rules is necessary to execute them. The result will be a list of the errors which can comprise the ontology checked by RACER or a consistent ontology. It should be noted, that if the checking of ontology fails, it is necessary to modify certain parts of the rule (we consider that ontology is consistent). So it appears reasonable to go up, if it is necessary until the step of conceptualization to avoid modifications which do not respect the semantics of the domain ontology.

4. Issues for Implementation

OWL is hard-coded against specific design patterns, but anything that goes beyond those patterns cannot be expressed. Furthermore, the choice of supported design patterns is misguided by theoretical assumptions about DL inferencing that are quite often irrelevant for practical purposes.

We have recently introduced the new SWRL keywords extension as a mechanism that uses the new keyword to formalize the meaning and behavior of Semantic Web concepts. They provide RDF vocabulary that can be used to attach rules to class definitions.

4.1 Description of the Application Domain

We chose preventive medicine as applicative field (the ontology of this domain has static and dynamic knowledge). The first type models the physical concepts, such as the isolated biological tests (hemoglobin, cholesterol…) the questionnaires have submitted to the patients, they have diagnosed pathologies. The second type specifies the type of supported reasoning; the rules of diagnosis and therapeutic treatment in the case of emergency. Thus we show how the rules were conceptualized then formalized and translated in RDF language by using Protégé tool and how to integrate them in ontology and finally how to check them through the use of RACER system.

We begin the design with the construction of the rules table, where we specify for certain concepts, the attributes necessary for each rule (cf.Table1)

Rules	Concept	Description	Type	Consequence	Conclusion
R1	Patient	Calculate	Rule Of BMI	Weight, size construction	BMI[1]
R2	Hemoglobin test	To indicate the type of Anemia	SWRL Rule (extra-concept)	Value of the test, age, sex	Result _H
R3	Vgm_test	To indicate the type of Vgm	SWRL Rule (extra-concept)	Value of the test, age, sex	Result_V
R3	Lymphocytes test	To indicate he type of lymphocytes	SWRL Rule (extra-concept)	Value of the test, age, sex	Result _L
R4	Hemoglobin test	To check the values of the attributes	Rule of checking	The attributes which have a default value in the concept Hemoglobin Test	Trigger an error.

Table1. Description of the Rules

After the conceptualization of the rules, we pass now to the step of formalization

R1: Patient (?p), Weight (?x), size (?y), swrlm:eval (?BMI, "?x/ ?y* ?y", ?x, ?y)^ **RuleConstruction** (?BMI) → sqwrl:select (?P,?BMI)

R2: Patient (?p), integer [>= 18 , <= 40] (?age), hasAge (?p, ?age), integer [>=19,<=25] (?BMI) -> hasStat (?p, normal)

R3: Patient (?p), doing_T_H (?p, y),integer [>=14,<=17] (?y), hasSex (?p, Male) -> hasAnemai (?p)

R4: Patient (?p), doing_T_V (?p, ?y),integer [>= 80 , <= 95] (?y), hasSex (?p, Male) -> has_Macrocytosis (?p)

R5: Patient (?p), doing_T_L (?p, ?y),integer [>=4000 , <= 10000] (?y), hasSex (?p, Male) -> has_Lymphocytose (?p)

R6: Patient (?p), hasAnemia (?p, ?x), has_Macrocytosis (?p, y)-> has_ Macrocytic_Anemia (?P, ?y)

So, in the formalization step, Protégé tool is used to codify them in SQWRL or SWRL language depending on the type of the rule. The RDF code source of each rule is integrated with RDF code corresponding to the ontology.

To motivate our work from a practical point of view, we propose the following system architecture which describes the application of the suggested process with a case study.

In this paper we introduce the new keyword from another angle to illustrate what makes it different, and why we believe that it

[1] Index of Body Mass

provides good solutions to many real-world requirements. Our main point is that this extension or SWRL rules is borrowing good practices from object-oriented programming, modeling languages and integrating object-oriented techniques with the flexible architecture of Semantic Web to produce a new way of working with linked data.

As this example, we'll start with a simple ontology. It defines a class patient with the following characteristics:

In this ontology, a Patient is a class that has four properties. The values of Weight, size, age and BMI are specified by the user, and the new SWRL rule is used to compute and calculate the value of the BMI (The Body Mass Index (BMI) is a measurement tool that compares your height to your weight and gives you an indication of whether you are overweight, underweight or at a healthy weight for your height) property by using the following formula: Weight in kilograms is divided by height in meters squared (weight (kg) / [height (m)]2). We have needed to add a new keyword as SWRL rule that has been attached to a class using the *construct* rule for example to do mathematical calculations.

A key contribution of this new kind of rules is to introduce a mechanism that allows users to organize those SQWRL queries in a natural object-oriented way. These rules are not just plain lists of rules like in comparable rule languages (SWRL etc). That you can arrange the rules in the class hierarchy where they belong. This follows the Object-Oriented principles of abstraction and encapsulation. Since the rules (and constraints) are attached to classes, any human or agent who looks at the ontology can quickly understand the meaning of the classes and properties. Furthermore, the rules are "*scoped* " so that tools are better guided when they need to execute the rules and constraints. Whenever someone changes the values of size weight, then the value of BMI will update automatically, as shown with the following figure the *creation of the rule Construction on protégé* (cf. Figure 4).

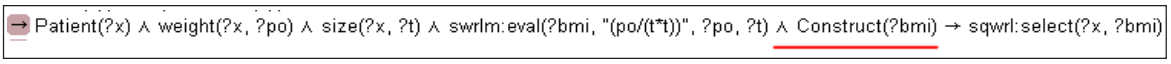

Figure 4. Rule Construction

We assume that a conversion from OWL + SWRL to Jess is available (including a complete implementation of all SWRL built-in functions, or conversion to equivalent Jess language mechanisms). SweetRules [9] provide this conversion, but they involve a fairly complex installation of many third-party software components. The current Protégé + SWRLTab tool set does not provide a conversion to Jess, although this is currently being developed, and should be available soon.

In this section, we consider some issues in implementing a rule. As noted previously, additional infrastructure must be implemented at the Jess level to enforce SWRL's semantics, and to provide certain extensions to SWRL.

4.2 System Architecture

The architecture of the proposed system in this work (Figure5) is based on the hybrid reasoning (the first reasoning is on OWL ontology, and the second on rules). The process supported by this architecture includes three phases: the first one consists of creating the ontology enriched by SWRL rules using Protégé editor. It is followed by the request treatment phase. The last phase concerns the semantic reasoning, in which, we integrate the two modes of reasoning.

4.3 Description of the System Architecturet

This architecture is made up of

• An editor of ontology and SWRL rules; to create hybrid ontology in OWL DL. These concepts are not simple; they are enriched by the two types of rules which we have quoted previously.

• A phase to treat the requests: in this paper, we don't interest in this phase.

• The use of RACER In the system: we choose this kind of reasoner because it is compatible with nRQL requests. An inference engine is also used to execute SWRL rules and the new type of rules. The results provided by RACER are regarded as facts base of the inference engine. From the initial facts (facts base), the inference engine must select SWRL rules to be triggered. The results of the execution have been sent to RACER. These last results may satisfy the request.

To validate the preceding phases, we have used the ontology which formalizes information on the domain of preventive medicine. The user brings the following request:

«Who is the patient reached by Macrocytic anemia and who has a standard weight? »

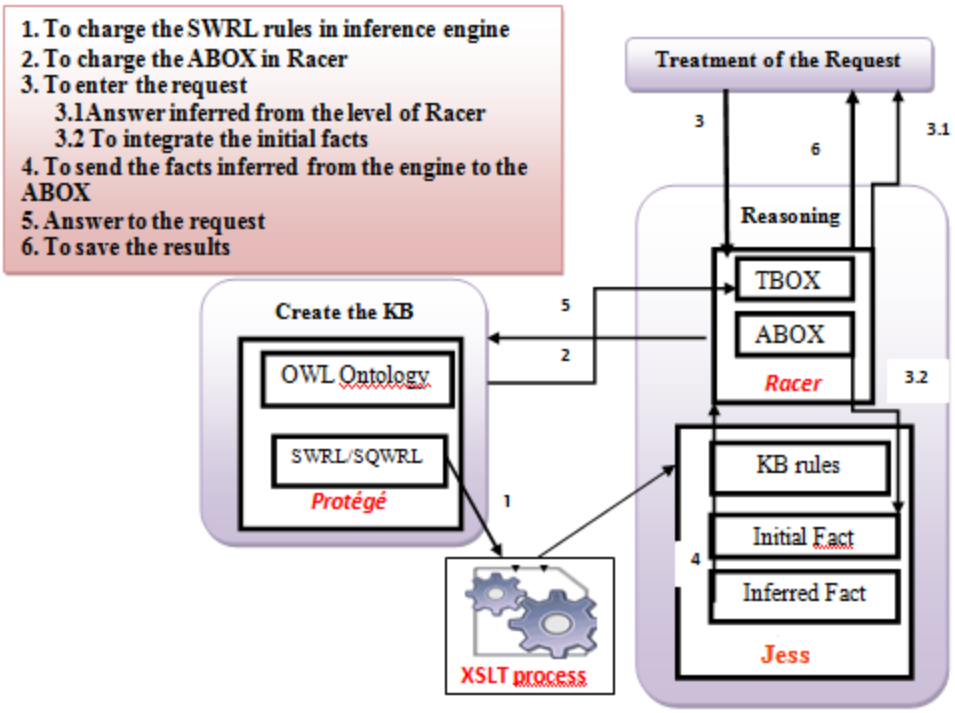

Figure 5. System Architecture

The translation of the request towards the language nRQL makes it possible to obtain the following form:

(retrieve (? patient? Macrocytic_anemia) (? Patient? Anemia_macrocytose reached_macrocytose (? patient? BMI normal)))

The Racer system uses reasoning by classification based on subsumption to classify the request in taxonomy, it will reason on the KB (Tbox+Abox), SWRL rules and rules (of checking and construction). The rules have been triggered in the following order:

Rule 1 (R1) to calculate the BMI. R2, R3 and R4 which are represented in SWRL to indicate the type of Anemia.

We notice that the answer is not complete. We are obliged to infer the new facts which are generated starting from SWRL rules, for example the following rule (cf. Figure 6).

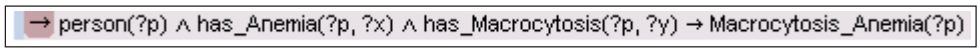

Figure 6. The rule selected by the inference engine

After the execution of this request « who is the patient suffering of Macrocytic anemia », the result is posted to the user.

5. Summary and Future Work

The reasoning on ontologies uses simple inference engine. According to insufficiencies of the supported languages, ontologies are extended by SWRL rules. We assess that SWRL isn't expressive enough to write rule sets for inherent to interpret a check up in preventive medicine. The problem is absence of reasoners capable to interoperate ontologies and rules. The SWRL rules combine the individuals of ontology; don't change the concepts and the knowledge base. So the reasoning stays incomplete. We solve this by adding the missing language constructs as new keywords in OWL/SWRL and a translation step to apply our own rewrite meta-rules.

Our approach which integrates rules to concept is done in two manners: 1) Binding the properties of concepts by SWRL rules for the verification and construction of new properties (intra-concept integration). These rules adapt the Object-Orient techniques.

The heritage of this type of concept makes them rules supported by all its instances. 2) Adding relations according to the variables values by SWRL rules (extra-concept integration).

In addition, the architecture of a system allowing the reasoning on this kind of ontology has been proposed. This system answers the nRQL requests. Many components of this architecture have been implemented. The future work will consist on building an interface (tool) which permits to create the rules and to integrate their RDF code into ontology.

References

[1] Horrocks, Patel-Schneider, P. F., Bechhofer, S., Tsarkov, D. (2005). OWL rules: A proposal and prototype implementation, *Journal of Web Semantics*.

[2] Golbreich, C., Bierlaire, O., Dameron, O., Gibaud, B. (July 2005). What reasoning support for Ontology and Rules? the brain anatomy case study. Workshop Protégé With Rules, Madrid, Spain.

[3] Bouarroudj, S., Boufaida, Z., (April 2010). Raisonnement sur une ontologie enrichie par des règles SWRL pour la recherche sémantique d'images annotées. COSI.201018-20,Ouargla, Algéria.

[4] Bruijn, J., Eiter, T., Tompits, H. (2008). Embedding approaches to combining rules and ontologies into autoepistemic logic. 11th International Conference on Principles of Knowledge Representation and Reasoning (KR2008), Sydney, Australia, September 16-19.

[5] Drabent, W., Henriksson, J., Maluszynski, J. (2007). HD-rules: A Hybrid System Interfacing Prolog with DL-reasoners. *In*: Proceedings of Applications of Logic Programming to the Web, *Semantic Web and Semantic Web Services*. 287, 76-90. CEUR-WS..

[6] A Practical Forward-Chaining OWL 2 RL Reasoner for OWL 2 DL Ontologies Combining Jena and Pellet (technical report) Georgios Meditskos and Nick Bassiliades (January 2009). This work was partially supported by a PENED program, jointly funded by EU and the General Secretariat of Research and Technology.

[7] Golbreich, C. (2004). Combining Rule and Ontology Reasoners for the Semantic Web, Invited talk, Rules and Rule Markup Languages for the Semantic Web, Hiroshima, Japan, Antoniou, G., Boley, H. Editors, LNCS 3323, Springer.

[8] Golbreich, C., Bierlaire, O., Dameron, O., Gibaud, B. (2005). Use case: Ontology with rules for identifying brain anatomical structures. W3C Workshop on Rule Languages for Interoperability.

[9] Hai, H., Wanga, J., Song., D., Jing, S., Jun, S. (2007). Reasoning support for Semantic Web ontology family languages using Alloy. *Multiagent and Grid Systems*. 2, 455-471.

[10] Boris, Motik, JESS. (2008). Scalable Reasoning over Ontologies with Large Data Sets *ERCIM News* 2008 (72).

[11] Wang, E., Kim, Y. S. (2007). Using SWRL for ITS through Keyword Extensions and Rewrite Meta-Rules. Workshop on Ontologies and Semantic Web Services for Intelligent Distributed Educational Systems (SWEL), 13th Int'l. Conf. Artificial Intelligence in Education (AIED), 101– 105, Marina Del Rey.

[12] Donini, F., Lenzerini, M., Nardi, D., Schaerf. AL-log, A. (1998). Integrating datalog and description logics, *Journal of Intelligent Information Systems*,10 (3) 227–252.

[13] Levy, A., Christine, M. (1998). Rousset. Combining Horn rules and description logics in CARIN. *Artificial Intelligence*, 104 (1–2) 165–209.

[14] Rosati, R. (1999). Towards expressive KR systems integrating Datalog and Description Logics. *In*: Proc. 1999 Int. Workshop on Description Logics (DL-1999), p. 160–164.

[15] Heymans, S., Van Nieuwenborgh, D., Vermeir, D. (2004). Semantic web reasoning with conceptual logic programs. *In*: Grigoris Antoniou and Harold Boley, editors, RuleML, *Lecture Notes in Computer Science*, 3323, 113–127. Springer.