A method to assist in the proper management of rule bases in Web information systems

Abir Boujelben
Universite de Sfax Faculte des Sciences Economiques et de Gestion de Sfax, Sfax, Tunisia, and
Ikram Amous
National School of Electronics and Telecommunication of Sfax, Sfax, Tunisia

Received 23 November 2018
Revised 18 April 2019
26 April 2019
Accepted 28 April 2019

Abstract

Purpose – One key issue of maintaining Web information systems is to guarantee the consistency of their knowledge base, in particular, the rules governing them. There are currently few methods that can ensure that rule bases management can scale to the amount of knowledge in these systems environment.

Design/methodology/approach – In this paper, the authors propose a method to detect correct dependencies between rules. This work represents a preliminary step for a proposal to eliminate rule base anomalies. The authors previously developed a method that aimed to ameliorate the extraction of rules dependency relationships using a new technique. In this paper, they extend the proposal with other techniques to increase the number of extracted rules dependency relationships. The authors also add some modules to filter and represent them.

Findings – The authors evaluated their own method against other semantic methods. The results show that this work succeeded in extracting better numbers of correct rules dependency relationships. They also noticed that the rule groups deduced from this method's results are very close to those provided by the rule bases developers.

Originality/value – This work can be applied to knowledge bases that include a fact base and a rule base. In addition, it is independent of the field of application.

Keywords Semantic web, OWL, SWRL, Rule bases, Rules dependency, Web information systems

Paper type Research paper

1. Introduction and preliminaries

Web information system is an information system that uses Web technologies to deliver information and services, to users or other information systems. It consists of some components operating together to accomplish a specific task. Usually, such a system must be up-to-date with its environment. Its knowledge base, which includes a fact base enhanced in some cases by a rule base, continuously acquires new knowledge. The large amount of information on the internet, the large number of users and the complexity of the application may cause raising inevitable inconsistencies in rule bases governing Web information systems. Such problems can undermine the performance of the system and the quality of its service. So, inconsistency elimination has to be accomplished. On the other hand, when experts need to edit the rule base, they must have a clear idea about the existing rules and their execution process. Thus, and due to the increasing sizes of rule bases, the automation of their management tasks (adding, editing and deleting rules, in addition to inconsistency elimination) has become a necessity. This crucial need is still tackled. For example, Ksystra and Stefaneas (2016) address the problem of automating rule bases verification. Salfinger *et al.* (2014), Khattak *et al.* (2016) and Davtalab and Malek (2018) try to manage context-aware systems and their evolutionary context rules.

International Journal of Web Information Systems
Vol. 15 No. 5, 2019
pp. 577-593

1744-0084
DOI 10.1108/IJWIS-11-2018-0081

Besides, there is a variety of available tools that offer help to system experts and to domain experts. For example, there are tools that allow a comprehensive representation of the rules, others try to explain the progress of the inference of some knowledge (Bak *et al.*, 2013) (O'Connor and Das, 2006). But the reasoning process within a huge rule base is still toilsome to follow. Recently, the successful adoption of Semantic Web technologies by Web information systems in many areas of application has led to new challenges for solving the problem of rule bases management. In this research framework, extracting dependencies between the elements of a rule base is considered as a fundamental step forwards the automation of such tasks.

1.1 Semantic Web technologies

Among the main objectives of the World Wide Web Consortium (W3C) is the development of technologies for the provision of structured data to be processed by machines. Thus, several technologies have been proposed. They include RDF (resource description framework), a flexible data model, SPARQL (SPARQL Protocol and RDF Query Language), a query language and RDFS (RDF Schema) and OWL (Ontology Web Language) which are schema and ontology languages for describing concepts and relation- ships (Arenas *et al.*, 2016). Ontology is a technology that introduces not only a shareable and reusable knowledge representation but also allows inferencing new knowledge about the domain.

OWL defines its fact base and SWRL (Semantic Web Rule Language) allows to express its rule base. In OWL, concept classification is achieved using classes and subclasses, whose instances are called individuals, between which semantic relationships are defined. OWL is based on three types of entities: classes (e.g. Person), instances (e.g Martin) and properties. These latter have two types: ob- ject properties defined between two classes (e.g. hasFather (Person, Man)) and data properties expressing the classes attributes (e.g. hasAge (Person, integer)). SWRL expresses the rules in the Horn form. Each rule is then composed of an antecedent part (Body) that infers a consequent part (HEAD). Each of these parts is composed of a positive conjunction of atoms each of which has the form Predicate (argument1, argument2, . . ., argumentN). The predicate references an OWL entity (class, property, etc.) and the arguments are variables, classes, data values, etc (See Table I). Consider Rule-x and Rule-y two examples of SWRL rules. Rule-x expresses the fact that a person over 17 is an Adult. While Rule-y indicates that an Adult may have a driving license:

Rule-x: *Person* (*p*) ∧ *hasAge* (*p, age*) ∧ *swrlb: greaterThan* (*p,* 17) → *Adult* (*p*)
Rule-y: *Adult* (*p*) → *canttetDrivingLicense* (*p, true*)

1.2 Dependency relationships

The dependency relationships between the elements of a rule base afford important knowledge specially when dealing with rule bases management. They reflect the state of the base and this plays an important role in the detection of the inconsistencies that it can

Table I.
SWRL atoms types

Atom type	Example
Class	Person(x)
Object property	hasParent(x, y)
Data property	hasAge(x,18)
Data range property	xsd:date(d)
Built-in	swrlb:lessThan(n,25)
Same individual	sameAs(x, y)
Different individuals	differentFrom(x, y)

incarnate. A dependency relation between two rules indicates that the second rule depends on the first rule. It also shows that the first rule must be executed before the second one. This implies that the application of the second rule requires the use of the facts inferred by the first one. As example, we consider $Rule - x$ and $Rule - y$ previously stated. It is obvious that $Rule - x$ produces facts that $Rule - y$ will use to infer other facts. So, $Rule - y$ depends on $Rule - x$.

In this research work, we propose a method for an automatic detection of dependencies between the elements of a rule base. Our proposal represents a step towards an approach for the automation of eliminating anomalies in a rule base. It paves the way also for the automation of rule bases management task.

The major contributions and novelties of our proposal are summarized as follows.

- First, we amend the existing semantic dependency extraction technique by analyzing more types of atoms.
- Second, we add the search for dependencies between atoms having different types (e.g. we analyze the dependency between a class atom and an object property atom).
- Furthermore, we integrate a previous work (Boujelben and Amous, 2018) as a third technique to extract dependencies between rules.
- We also propose to represent the extracted dependencies with an ontology to benefit from its expressiveness and inference capacities for accomplishing any further tasks and offer the designer a better opportunity to enjoy the abilities of ontologies to get knowledge about the state of the rule base according to his own perspective.

The rest of the paper is organized as follows: Section 2 presents a literature review. Section 3 introduces our proposal and details its modules performance. The results of its evaluation are provided and discussed in Section 4. Section 5 presents a conclusion and our future work.

2. Related work

Several approaches and tools are available for managing rule bases evolution (Khattak *et al.*, 2016). Most recent works addressing such a problematic are based on extracting dependencies between a rule base elements. In this section, we enumerate and discuss the existing works tackling the issue.

To explore the structure of a rule base is to explore the dependency relationship among the facts and the rules contained in the rule base. The idea of extracting dependencies between rules consists in defining which rule depends on which others. Such information allows revealing some knowledge about the rules execution process and the rule base status. Extracting dependency relationships between rules has been investigated since expert systems appearance (Fenton *et al.*, 2001) (Huhns and Acosta, 1988). It is a key task when dealing with rule bases issues: representing rules, generating rules execution order, detecting inconsistencies (Wu *et al.*, 1994), ensuring correct queries answers and explanations (Huhns and Acosta, 1988).

2.1 The state of the art of extracting dependencies between rules

In literature, searching for rules dependencies is achieved in different ways. Some methods use the rule patterns compliance. Other methods analyze the rule usage data. Some others are based on extracting dependencies between the head and the body of each couple of rules in the rule base.

2.1.1 Pattern compliance. Katta *et al.* (2016) propose a system to ameliorate the quick decision making in Software- Defined Networks. The pattern of each rule is determined, and a dependency exists between two rules if their patterns intersect.

2.1.2 Usage data. Xitao *et al.* (2014) introduce a framework to modify the forwarding policies installed in distributed switches of a network. The authors extracted dependency relationships between rules to eliminate unnecessary priority updates. The dependency graph is built incrementally along the compilation process. In Zacharias and Borgi (2006), the authors present an approach to ensure a clear visualization of rule bases. Their proposal refers to given usage data to find and highlight frequently used rules and their dependencies. For managing large rule bases, Dani *et al.* (2014) propose a system based on the dependencies between rules. These dependencies are determined as a function of rules execution frequency data generated from applying the rules over a data set.

2.1.3 Atoms analysis. The methods includes in this category analyze relationships between first rule head atoms and second rule body atoms. They are divided into two main groups: syntactic methods and semantic ones. *Syntactic methods* use equality and equivalence relationships between atoms' predicates. *Semantic methods* enhanced syntactic ones by using hierarchical relationships defined between entities referenced by the atoms' predicates. Dolinina and Shvarts (2015) suggest to reduce the time of making decisions on the base of grouping of the rules and variables, and the dependencies between the rules. In this work, the dependency extraction is based on the fact that some of the head atoms of the first rule exist in the body of the second one. In Baget *et al.* (2014), the authors proposed a tool, based on a rule dependency-based approach, to ensure the termination of a breadth-first forward chaining algorithm in the context of Ontology-Based Query Answering. The dependency extraction uses the equality between atoms predicates and their variables unification. On the other hand, the first version of Axiomé (Hassanpour *et al.*, 2009) was based on an analysis of references to the same classes and object properties without considering the variables analysis. In 2010, the second version of Axiomé (Hassanpour *et al.*, 2010) was proposed to enhance the first version by analyzing the domain and range of object property atoms and considering the hierarchical relationships between classes and properties defined in the ontology. Furthermore, Slider-p (Chevalier *et al.*, 2016) is a reasoner proposed to handle streaming data with a growing background knowledge base. It is based on the analysis of a dependency graph built using a semantic method. To build the graph, the authors considered only class atoms and object property atoms.

2.2 Synthesis

Overall, the summary hints at several gaps in the previous dependency extraction research:

- The dependency analysis based on the rule execution history requires an important number of iterations. This should be followed by an anal- ysis of the consistency and accuracy of the results obtained after iterations.
- The dependency analysis based on the verification of the existence of a dependency between each pair of rules is based on the search for whether the first rule can produce facts that can be exploited by the second rule. This is due to the fact that the expression *R*2 *depends on R*1 implies that the head of R1 produces facts that will be exploited by the body of *R*2. This method is distinguished from the first one by the characteristic that it is made independently from the execution of the set of rules. So it does not disturb the system performance. In addition, it is based on the semantics expressed by the ontology. So, it is based on the logic of the domain. Its

main drawback is the fact that its atoms analysis is based only on class atoms and object property atoms.

- The interfaces used for dependencies representation are difficult to manage and to understand due to the big number of rules. The dimensions of tables and matrices obtained are enormous. This leads to inability to interpret the dependencies obtained. This results in an insufficiency of taking advantage of the extracted knowledge.
- In the existing work interested in rules dependency extraction, we did not find any experimentation that computes and compares the numbers of correct extracted dependencies and false ones.

In the following section, we introduce a new method to handle all the cited gaps by leveraging existent work and proposing a new technique to guarantee better results.

3. RuDES method

In this section, we introduce our method, called RuDES, for extracting dependencies among a rule base. Its general architecture is provided in Figure 1. It takes as input the fact base and its associated rule base, and generates possible dependency relationships between each couple of rules. RuDES is based not only on semantics expressed by the ontology definition but also on those expressed by the rule base. It consists of five different modules:

(1) same type atoms analysis;
(2) different type atoms analysis;
(3) heads analysis;
(4) filtering dependencies; and
(5) storing the extracted dependencies in an ontology we called RuDOnt.

To overcome the problem from different perspectives, the first three modules operate to extract dependencies using different techniques. Each module provides a direction and its associated weight to the Filtering module which will choose the most suitable one. The extracted dependency is represented, using the fifth module, as an instance in the RuDOnt ontology. Thereby, the designer is offered the opportunity to enjoy the services provided by the available ontology visualization and query tools.

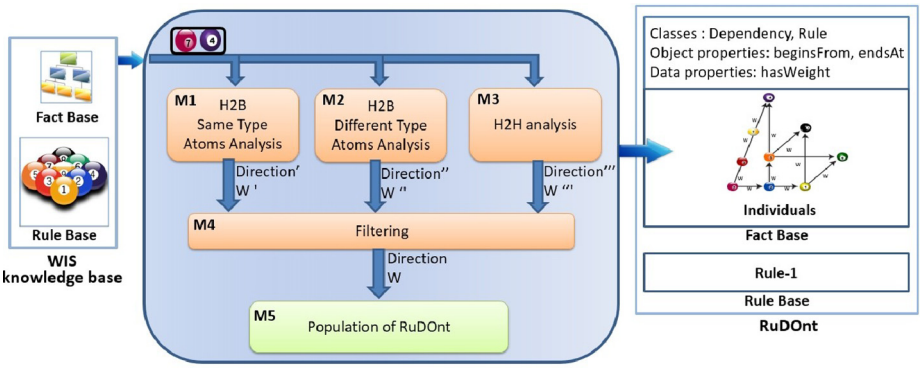

Figure 1.
RuDES architecture

The following subsections detail all modules performance and end with an execution example on some of the following rules relating to the DFO ontology (Section 4.1).

RD-1: Person(p) ∧ Half Morning(hm) ∧ Food(f) → hasDietSnackMorn(p, f)
RD-10: *Person*(p)∧*hasBmiType*(p, *bmitype*)∧*swrlb: equal* (*bmitype*, "*Healthy*")∧ *hasttlycemicIndex*(p, *gi*) ∧ *swrlb: greaterThanOrEqual*(*gi*, 80) ∧ *swrlb*:lessThanOrEqual(gi, 120) ∧ Diabetic(p) ∧ hasDietSnackMorn(p, f) → *Banana*(*f*) ∧ *dietSnackMorn*(*f*, *f* 042)
RD-23: *Person*(p) *hasBmiType*(p, *bmitype*) *swrlb: equal*(*bmitype*, "*Overweight*") *hasttlycemicIndex*(p, *gi*) *swrlb: greaterThanOrEqual*(*gi*, 120) *swrlb: lessThanOrEqual*(*gi*, 150) *Diabetic*(p) *hasLunch*(p, f) *Biscuits*(f) *dietSnackBmi*30*gi*120 (*f*, *f* 001)

3.1 Module 1: Same type atoms analysis

Algorithm 1 Computing weights in case of two atoms having the same type:

Require: (a1, a2) : two atoms having the same type
Require: *type: the atoms*^{j} *type*
Ensure: w^j : *real*
1: *n: integer*
2: *wp*, $w^j \leftarrow 0$
3: **if** *there are relationships between all atoms*^{j} *parts* **then**
4: **if** *type* ∈ {*classAtom*; *objectPropertyAtom*} **then**
5: *wp* ← 1
6: **else if** *type* = *datatPropertyAtom* **then**
7: *wp* ← 0.6
8: **end if**
9: *n* ← *countHierarchicalRelationships*(a1, a2)
10: $w^j \leftarrow wp - 0.1 * n$
11: **end if**

The main idea of this module is based on two facts: (i) if R_j depends on R_i, then the facts inferred by R_i (i.e. expressed by R_i's head) will be used by Rj (i.e. expressed by R_j's body), (ii) the treated atoms must be of the same type. A dependency analysis is performed between each pair of rules R_i, R_j from the rule base. It searches for a dependency from R_i to R_j, then for a dependency from R_j to R_i. Then the weakest dependency is discarded. A dependency research from R_i to R_j is based on analyzing all couples of atoms (a_i, a_j) having the same type, where a_i is from the first rule head and a_j is from the second rule body (Algorithm 1). There is a dependency relationship from R_i to R_j if there is a dependency relationship between one atom couple (a_i, a_j). And this affirmation is confirmed if there is equality or equivalence or hierarchical relationships between all parts of a_i and a_j (Algorithm 1 L3 detailed in Figure 2). The weight attributed to the dependency between the rules is the highest among all weights computed for the couples of atoms.

In case of *two class atoms*, we look for equality or equivalence relationships ranging from the first referenced class to the second. In case of *two object property atoms*, they are interdependent if there is a dependency between the predicates and between the arguments having the same position. We note that a couple of same individual atoms and

couples of different individuals atoms are treated as object properties having the same predicate. In case of *two data property atoms*, we seek for equality, equivalence and hierarchical relationships between the predicates and between the first arguments.

The types of the relationships found between a_i and a_j parts (rel_i in Figure 2 (i [1.2])) allow specifying the weight to be assigned to the dependency between this couple of atoms. The relations of equality and equivalence indicate a strong dependency, whereas the relations of hierarchy weaken it (Algorithm 1 L9 and L10). We note also that in case of data property atoms, weights are lower than other cases. This is due to their relation to data values which vary during the inference process (Algorithm1 L7).

3.2 Examples

3.2.1 Applying module 1 on (RD-1, RD-10). Module 1 analyzes couples of atoms (ai, aj) having the same type. ai is an atom from RD-1 head and aj is an atom from RD-10 body. As it can be seen, there is the only couple (hasDietSnack- Morn(p, f), hasDietSnackMorn(p, f)). There is an equality relationship between the predicates. The same class Person is referenced by both atoms first arguments. And the same class Food is referenced by both atoms second arguments. So, there is a dependency from RD-1 to RD-10 which weight is w1 = 1 (no hierarchical relationships to decrease the weight).

3.2.2 Applying module 1 on (RD-10, RD-1). On the other hand, to analyze the dependency from RD-10 to RD-1, Module 1 analyzes the dependency between the atom Banana(f) and, the atoms Person(p) and Food(f). There is no dependency between the couple (Banana(f), Person(p)). Yet, there is a dependency between (Banana(f), Food(f)), and its weight is initialized to 1. Banana is declared in the ontology definition as sub-class of Food (a hierarchical relationship). So, w2 = 1-0.1 = 0.9.

As w1 > w2, Module 1 states that there is a dependency from RD-1 to RD-10 which weight was w' = 1.

3.3 Module 2: different type atoms analysis

This module relies on (i) analyzing the dependency from the first rule head to the second rule body (Algorithm 2, L7 and L8), and (ii) treating pairs of atoms having different types (Algorithm 2, L9). It analyzes the dependency from the first rule to the second, then it analyzes the opposite direction. To analyze the dependency from one rule to another, whatever are the types of atoms, this modules opts for extracting the classes invoked in each one (Algorithm 2, L10 and L11) and sees if there is a relationship (equality or equivalence or hierarchy) defined in the fact base linking a class invoked in the first atom to a class invoked in the second one (Algorithm 2, L14). As in the previous module, the type of a found relationship allows specifying the weight to be assigned (Algorithm 2, L15). Thus, the equivalence and equality relationships maintain the dependency strength weight, a hierarchy relationship decreases it. After each direction analysis, the module keeps the weight of the strongest dependency found among those of all analyzed couples of atoms (Algorithm 2, L16 and L22).

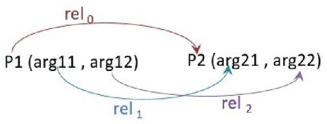

Figure 2.
Dependency analysis between two atoms having the same type

Algorithm 2 Different type atoms analysis

```
Require: Ri, Rj : rule
Ensure: w^jj : real
  1 : aH, aB : atom
  2 : clsH, clsB : set of classes
  3 : cH, cB : class
  4 : x, wp : real
  5 : rel ∈ equivalence, equality, hierarchy
  6 : wp ← 0
  7 : for aH ∈ Head(Ri) do
  8 :   for aB ∈ Body(Rj) do
  9 :     if Type(aH) = Type(aB) then
 10 :       clsH ← classesIn(aH)
 11 :       clsB ← classesIn(aB)
 12 :       for cH ∈ clsH do
 13 :         for cB ∈ clsB do
 14 :           rel ← relationSearch(cH, cB)
 15 :           x ← relationAnalysis(rel)
 16 :           wp ← Max(wp, x)
 17 :         end for
 18 :       end for
 19 :     end if
 20 :   end for
 21 : end for
 22 : w^jj ← wp
```

3.3.1 Example: applying Module 2 on (RD-1, RD-23). Module 2 analyses couples of atoms having different types. It proceeds with extracting the invoked classes in each atom. Then, it looks for relationships connecting them. Thus, the treated atom couples are (*hasDietSnackMorn*(*p, f*) , *Person*(*p*)); (*hasDietSnackMorn*(*p, f*) , *hasBmiType*(*p, bmitype*)); etc. For the first couple, there is an equality relationship between the class referenced by the first argument of the first atom and the class referenced by the second atom predicate (class Person). So, there is a dependency from RD-1 to RD-23 which weight is w1 = 1 (no hierarchical relationships are involved in the analysis). The other couples of atoms are similarly analyzed. The highest weight found is w1 = 1.

3.4 Module 3: heads analysis

This module is based on extracting semantics from the fact base and others from the rule base. On the one hand, in the fact base, object properties express relationships between classes. An object property indicates a specific direction (an upward link, a downward link or a horizontal link) from one class to another (Hirst *et al.*, 1998). Downward links are defined from the more general class to the more elementary one, and upward links are defined from the more elementary class to the more general one (Hirst *et al.*, 1998). On the other hand, for rule bases developed in a top-down process, first inferred facts concern the more general classes than latter ones. Yet, for rule bases developed in a bottom-up process, first inferred facts concern the more elementary classes. Thus, the main idea of this module performance is to follow upward links in case of rule bases developed in a bottom-up process, and to follow downward links in case of rule bases developed in a *top-down process*.

We present in Algorithm 3 the way Module 3 extracts a dependency between two rules. It returns the direction of the extracted dependency (*Rx* to *Ry* or vice versa) and its weight.

This module is based on four main treatments. First, it extracts the classes invoked in both rules heads in two different sets (Algorithm 3 L8, L9). Then it refers to the fact base to extract the object properties defined between classes from one set and those from the other (achieved by the function objectPropertyLinks in Algorithm 3 L10). After that, each object property is analyzed with reference to the fact base and to the rule base. For each property, there is a weight wp_i and a direction $direction_i$. Finally, the dependency between Rx and Ry, if found, gets the highest computed weight (Algorithm 3 L37) as well as its corresponding direction (Algorithm 3 L38).

The object properties analysis is achieved as follows. The weights are computed with reference to the approach A used to develop the rule base (is it as top-down or a bottom-up) (Algorithm 3 L14 to L28). They also depend on the hierarchical relationships defined between the classes invoked in the rules heads (C_m and C_n) and those referenced in the definition of the property in the fact base (C_1 and C_2)(Algorithm 3 L30 to L36). A weight decreases (i) if the approach A and the object property do not express the same direction (Algorithm 3 L21, L22), and (ii) if there are hierarchical relationships between the analyzed classes (C_m, C_n and C_1, C_2)(Algorithm 3 L31 to L33, L34 to L36).

More details about this module performance are provided in a previous work.

Algorithm 3 Extracting the dependency between two rules using Module 3

```
Require: Rx, Ry: rule
Require: A ∈ {top − down; bottom − up}
Ensure: direction ∈ {from Rx to Ry, from Ry to Rx}
Ensure: w^jjj: real
 1: Sx, Sy: set of classes
 2: C_m, C_n, C_1, C_2: classes
 3: OP: set of object properties
 4: size: integer
 5: p_i, p_owl: predicate of an object property
 6: wp_i, wp_max: real
 7: wp ← 0
 8: Sx ← classesIn(Head(Rx))
 9: Sy ← classesIn(Head(Ry))
10: OP ← objectPropertyLinks(Sx, Sy)
11: size ← | OP |
12: for p_i(C_n, C_m) ∈ OP, i ∈ [1, size] do
13:   wp_i ← 1
14:   if p_i(C_n, C_m) expresses the same direction as A then
15:     if C_n ∈ Sx and C_m ∈ Sy then
16:       direction_i ← from Rx to Ry
17:     else
18:       direction_i from Ry to Rx
19:     end if
20:   end if
21:   if p_i(C_n, C_m) expresses the opposite direction to that
      expressed by A then
22:     wp_i ← wp_i − 0.1
23:     if C_n ∈ Sx and C_m ∈ Sy then
24:       direction_i ← from Ry to Rx
25:     else
26:     direction_i ← from Rx to Ry
27:     end if
28:   end if
```

```
29: end for
30: p_owl(C1, C2) ← OWLdefinition(p_i(C_n, C_m))
31: if C_n is a sub − class of C1 then
32:    wp_i ← wp_i − 0.1
33: end if
34: if C_m is a sub − class of C2 then
35:    wp_i ← wp_i − 0.1
36: end if
37: w^jjj ← the maximum value wp_max among all wp_i
38: direction ← the direction_i that corresponds to wp_max
```

3.4.1 Example: Applying module 3 on (RD-1, RD-23). Unlike the previous modules, this one analyzes the dependency between RD-1 and RD-23 heads. First it extracts the invoked classes in both parts in two separate sets (resp. Sx and Sy). Thus, Sx consists of the classes Person and Food, and Sy is made up of the classes Biscuit and Food. In a further step, the module searches for object properties linking classes from Sx and classes from Sy in the ontology definition. In this case, there are 15 object properties defined from class Person to class Food (e.g. hasDietSnackMorn(Person, Food); hasLunch(Person, Food), etc). All these object properties express a downward link from class Person to class Food (implicitly an upward link from class Food to class Person). Knowing that the rule base is developed in a top-down process, Module 3 follows the downward direction. Person is invoked in RD-1 head and Food is invoked in RD-23 head. So, there is a dependency from RD-1 to RD-23. In this analysis, there were no hierarchical relationships between the classes invoked in the rules and the classes referenced in the properties definitions. Besides, the direction expressed by the properties is the same as the one expressed by the process used for developing the rule base. So, the weight of the extracted dependency is w''' = 1.

3.5 Module 4: Filtering

The role of the filtering module consists in keeping the most suitable weight w^j taking into account the module that generated it. Thus, before making its choice, the module multiplies each weight by a coefficient *ci* that we have already fixed. The highest coefficient (c3 = 1) is attributed to Module 3 as it is based on knowledge extracted from the ontology definition and from the rule base. The midst coefficient (c1 = 0.9) is attributed to Module 1 as it is based on knowledge from the ontology definition and equality between the analyzed atoms. The weakest coefficient (c3 = 0.8) is attributed to Module 2, as it relies on the less certain knowledge (equality or equivalence or hierarchical relationships between classes from the first rule's head to those from the second rule's body).

Then, the filtering module keeps the highest weight among those provided by the first three modules. It should be noted that other values can be assigned to these coefficients while maintaining this order and these margins of difference between them. We have experimentally verified our choice and it was validated.

3.5.1 Example: applying the filtering module. All the dependency relationships extracted for the couple (RD-1, RD-10) and (RD-1, RD-23) are summarized in Table II.

Dependency from RD-1 to RD-10: The resulting weight is 0.9. It is the highest among wm1 = 0.9*w'=0.9, wm2 = 0.8*w"=0.8 and wm3 = 1*w'''=0.9.

Dependency from RD-10 to RD-1: The resulting weight is 0.8. It is the highest among wm1 = 0*0.9 = 0, wm2 = 1*w"=0.8 and wm3 = 0*w'''=0.

As a result, the filtering module states that there is a dependency, named d110, between RD-1 and RD-10. It begins from RD-1 to RD-10 and has the weight 0.9.

Dependency	Module 1w'	Module 2 w"	Module 3 w"'	$w^j * ci$	Filtering Max val	w
RD-1 to RD-10	1	1	0.9	1*0.9 1*0.8 0.9*1	*0.9*	0.9
RD-10 to RD-1	0	1	0	0*0.9 1*0.8 0*1	0.8	0
RD-1 to RD-23	0	1	0.9	0*0.9 1*0.8 0.9*1	*0.9*	0.9
RD-23 to RD-1	0	1	0	0*0.9 1*0.8 0*1	0.8	0

Notes: $w^j \in \{w^j, w^{jj}, w^{jjj}\}$ and $i \in [1, 3]$

Table II. First three modules and filtering module application results

The same process is applied on the dependency between RD-1 and RD-23. The outcome dependency, named d123, is defined from RD-1 to RD-23 and has the weight 0.9.

3.6 Module 5: population of RuDOnt

Our work is devoted to facilitate the management of big rule bases. The extraction of the dependencies between those rules allows collecting knowledge about them. Thus, it allows to have a clear idea about the base and facilitates its modification without disrupting its good state. Ontologies give the power to model, reason and manage complex data which is the case of dependency relationships extracted among a huge rule base. We think that the use of ontologies is the most appropriate solution to benefit from their ability to infer new knowledge about the rule base and to reason to answer queries that might be asked by the designer. This can be used through existing ontology manipulation tools as Protégé, ViCoMap (Ristoski and Paulheim, 2015), etc. (Bak *et al.*, 2013) (Yadav *et al.*, 2016). To save the extracted knowledge about dependencies between rules, we created an ontology called RuDOnt which structure is presented in Table III.

It includes two main classes: rule and dependency. A dependency relationship is defined between two rules through two object properties: beginsFrom and endsAt which domain is the class Dependency and range is the class Rule. Each defined dependency has its own weight defined by the numerical data property hasWeight. To specify the transitive property for dependency relationships, we defined the following SWRL rule (*Rule* − 1).

Rule-1: Dependency(d1) ∧ beginsFrom(d1, Rx) ∧ endsAt(d1, Ry) ∧ *Dependency*(*d2*) ∧ *beginsFrom*(*d2, Ry*) ∧ *endsAt*(*d2, Rz*) ∧ *Dependency*(*d3*) → *beginsFrom*(*d3, Rx*) ∧ *endsAt*(*d3, Rz*)

Classes	Class: Rule SubClassOf: owl:thing Class: Dependency SubClassOf: owl:thing
Object properties	ObjectProperty: begins From Domain : Dependency Range: Rule ObjectProperty: endsAt Domain : Dependency Range: Rule
Data properties	DataProperty: hasWeight Domain : Dependency Range: real

Table III. RuDOnt ontology structure

Example: The dependencies extracted by previous modules are then encoded in the RuDOnt ontology as shown in Table V.

4. Evaluation

In this section, we provide an evaluation of our method. We implemented a prototype integrated into Protégé1 editor using the java language. As study cases, we considered two ontologies: WS-SecurityPolicy ontology (WS-SP), from the Web service domain, and Diabetic Food Ontology, from the medical field. To the best of our knowledge, ontologies including a rule base with an already known execution scenarios are scarce.

4.1 Data set

WS-SecurityPolicy ontology (WS-SP) (Brahim *et al.*, 2016) is from the Web Security domain. Its main role is to specify and match web service security policies between a requester and a provider of a Web service. It includes about 80 rules developed in a bottom-up process. The rules are divided in five groups. Each one is assigned a specific task. The tasks have to be performed in the following order (Figure 3a). The first group instantiates the semantic relations that may exist between atomic security properties. The second group elements instantiate the semantic relations that may exist between complex security properties. Then, the third group does the matching between the security assertions. After that, the elements of the fourth group carry out the matching between the security alternatives. Finally, the last group achieves the security matching degree decision.

Diabetic Food Ontology (DFO) was developed with collaboration with domain experts (doctors and dietitians) to specify diabetic everyday meals. To accomplish this task, it includes 24 rules developed in a top-down process. They are divided into four groups (Figure 3b). The first group computes the body mass index (BMI) of the patient. Then, the second group

Table IV. Population of RuDOnt with informations about d110 and d123

Individuals	d110 instanceOf: Dependency d123 instanceOf: Dependency
Object properties	beginsFrom(d110, RD-1) · endsAt(d110, RD-10) beginsFrom(d123, RD-1) · endsAt(d123, RD-23)
Data properties	hasWeight(d110, 0.9) · hasWeight(d123, 0.9)

Table V. Extracted dependencies by RuDES modules (WS-SP case)

	Module 1 (%)	Module 2 (%)	Module 3 (%)
Correct dep.	5.33	1.78	94.22
False dep.	29.41	94.66	2.71
Missing dep.	94.67	98.22	5.78

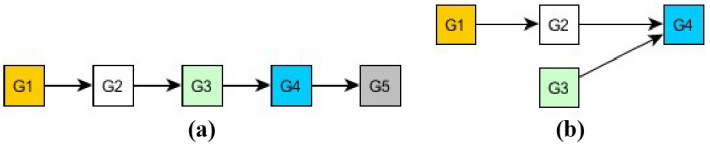

Figure 3. Rule groups provided by ontologies designers

Source: (a) WS-SP; (b) DFO

specifies the type of the BMI index. The third group indicates the period when the specified patient should take the meal. Finally, the fourth group uses it in addition to the type of the BMI index, generated by the second group, to specify the meal that the patient will take.

Axiomé is a free open source Protégé plugin for SWRL rule management (Hassanpour *et al.*, 2010). The rule base visualization is accomplished using a dependency graph developed following a semantic method. This method is based on searching for dependencies by:

- analyzing references to the same OWL classes and object properties; and
- analyzing the object properties domains and ranges.

Axiomé searches for dependencies using equality, equivalence and hierarchical relationships.

Thus, we think that this work includes almost all existing techniques for the atoms analysis method.

4.2 Performance of dependency extraction modules

To evaluate the performance of the different dependency extraction modules (Modules 1, 2 and 3), we applied them separately to the two case studies and we compared the obtained results to those provided by the rule bases developers. We were then able to determine the numbers of correct dependencies, of false dependencies and those not extracted.

Tables V and VI display the extracted dependencies in the two cases of study. They show the percentage of correct and false dependencies among all those extracted by each module of the RuDES method.

Module 1 extracts dependencies by analyzing pairs of atoms, having the same type, one of which is from the Head of the first rule and the second is from the Body of the other. The results showed that this module determined only a small number of false dependencies. On the other hand, it missed a large number of non-extracted dependencies. This fact proves that the analysis of atoms having only the same type impedes the extraction of some dependencies. But it contributes in detecting some dependencies that are not extracted by other modules. Module 2 acts in the same way as the first module, but it studies atoms having different types. This module has actively contributed to the extraction of correct dependencies in the DFO case. But the number of false dependencies is high. In the WS-SP case, it extracted a small number of correct dependencies. It extracted also a large number of false dependencies compared to the other two modules. These results show that this module sometimes participates in increasing the number of correct dependencies[1], but it contributes to extracting an important number of false dependencies. We also noticed that it allowed to determine dependencies not extracted by the other two modules. To take advantage of this module, we eliminated its drawback by the Filtering module.

However, Module 3 allowed the extraction of a large number of correct dependencies. It generated only a small number of false dependencies in both cases. In the case of WS-SP, Module 3 extracted more than 2 of the dependencies. In the DFO case, however, it has pulled out almost

Table VI. Extracted dependencies by RuDES modules (DFO case)

	Module 1 (%)	Module 2 (%)	Module 3 (%)
Correct dep.	46.29	100	55.55
False dep.	0	53.04	0
Missing dep.	53.71	0	44.45

the half. This fact proves the effectiveness of the proposed technique. This is due to the association of knowledge expressed by the fact base and that expressed by the rule base.

These results proved that each of these three modules participates, in some cases, in raising the number of correct dependencies. In other cases, it participates in reducing the number of false ones. This proves that these modules are rather complimentary.

In addition, we compared the sets of correct dependencies provided by each module. We noticed that they are disjoint. This confirms that using all of these techniques is mandatory to improve dependency extraction results.

In the next section, we present the evaluation of all the modules of our method and the interaction between them.

4.3 RuDES performance

To position ourselves considering the existing work, we looked for the most complete existing one, which is Axiomé.

We applied all our method's modules and Axiomé on the two case studies already presented. The results are displayed in Tables VII and VIII These latter show the amounts of correct and false extracted dependencies (among all the extracted ones), and those of the dependencies that have not been found (missing ones). These values were computed in the same way as the one used for computing the results presented in the previous section.

In WS-SP ontology case, Axiomé extracted a small number of dependencies among which almost 30 per cent are false. RuDES practically extracted all the requested dependencies with only 4.44 per cent of false ones. These results are due to the fact that Axiomé does not consider built-in atoms, while 54 per cent of the WS-SP ontology rules are based on this type of atoms.

In the DFO case, as the rule base does not include any built-in atom, Axiomé did not extracted any false dependency. It managed to extract almost half of the requested ones. As for RuDES, it extracted almost all the requested dependencies with a low number of false ones. These better results are due to the fact that RuDES involves the technique used by Axiomé and extends it with other techniques as shown before.

Moreover, in WS-SP rule base, the dependency graph generated by Axiomé included 62 isolated rules (which have no income or outcome edges). But RuDES graph does not include any isolated rule. So, it can be concluded that, in the two study cases, RuDES allowed improving the results of dependencies extraction among the elements of a rule base compared to the existing work.

4.3.1 Dependencies representation. Some existing works have tried to provide a comprehensible representation of the rule base to help the experts in a better edition. Some

Table VII.
Extracted and missed dependencies (WS-SP case)

	Axiomé (%)	RuDES (%)
Correct dep.	5.33	95.55
Missing dep.	94.67	4.45
False dep.	29.41	2.71

Table VIII.
Extracted and missed dependencies (DFO case)

	Axiomé (%)	RuDES (%)
Correct dep.	46.29	94.44
Missing dep.	53.71	5.56
False dep.	0	12.06

have represented each rule separately (Hassanpour *et al.*, 2010), others have used tables (Xitao *et al.*, 2014), and others have used graphs (matrices) (Hassanpour *et al.*, 2010). It is obvious that if the base exceeds a dozen rules, its comprehension and its management become complicated. Thus, we proposed to use an ontology. This latter represents the dependencies between the rules. This allows the experts to take advantage of knowledge about the state of the rule base using appropriate tools as we mentioned in Section 3.5.

4.3.2 Complexity analysis. Here, we present an analysis of the performance of our proposition's algorithm. To do this, we analyzed its temporal complexity and spatial complexity.

4.3.3 Temporal complexity. Our method modules represent the main operations of its algorithm. Each of the first three ones has a worst-case complexity equal to $O(n2)$. The other two have constant complexity. Thus, the worst-case complexity of the general algorithm is equal to $O(n2)$. We note that semantic methods mentioned in Section 2 have the same complexity as our work.

4.3.4 Spatial complexity. In all application cases, our algorithm takes as input the fact base, the rule base, and the type of process used for developing the rule base. So, the input size depends only on the size of the base. On the other hand, when executed, our algorithm uses four intermediate variables, those that allow exchanging results between the modules. Each of these variables has the form of a couple (weight, direction) where the weight is a real and the direction is a String from the list *Rx2Ry, Ry2Rx* . Thus, the spatial complexity of our algorithm is constant.

In conclusion, the temporal and spatial complexities of our method are acceptable, which proves the adaptability of our proposal to large rule bases.

4.4 Evaluation by rules clustering

Rule bases consist of groups of rules. The elements of each group are responsible for a well-defined task during the inference process. The rules of the same group can be applied in any order. Yet, rules from different groups must execute in a well-defined order to infer correct and consistent knowledge.

In the two used ontologies, the designers provided the rules in groups (as described in Subsection 4.1). Figure 3(a) presents the rule groups provided by the WS-SP ontology designers, whereas Figure 3(b) shows the rule groups provided by the DFO ontology designers. In this subsection, we choose to prove the efficiency of our method using the rules clustering to compare the resulting groups to those provided by the ontologies designers. The clustering algorithm is based on extracting the rules groups from the rules dependency graph DepG. The elements of the same group have the same incoming edges and the same outgoing ones.

As shown in Figure 4(b), in the case of the DFO ontology, the extracted dependencies between the groups are all correct. In addition, the dependencies to be extracted are all present. Thus, RuDES succeeded in extracting the dependencies allowing to guarantee a correct and a consistent inferred knowledge. Besides, in the WS-SP ontology case [Figure 4(a)], the dependencies between groups are all present and their performance order is close to the one specified by the designers. On the other hand, in both cases (Figure 4), there is a coincidence between the extracted groups and those provided by the ontologies designers. This proves that RuDES succeeded in determining which rules would be responsible for the same task.

5. Conclusion and future work

Rule bases supporting Web information systems performance are continuously modified to be up to date with the systems' environments. These changes may cause inevitable inconsistencies which have to be eliminated to ensure information consistency and service quality. In this paper, we proposed a method to boost the automation of rule bases

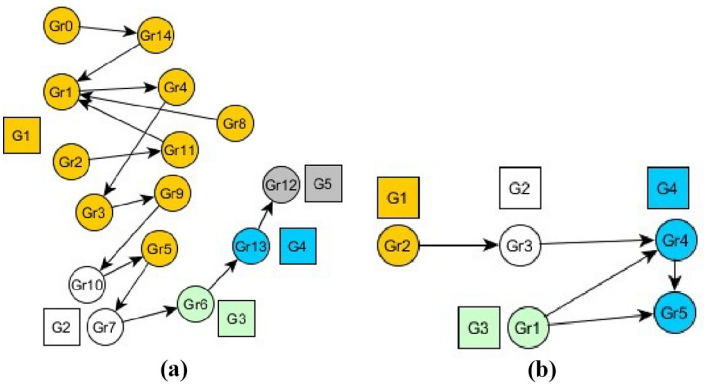

Figure 4.
Rule groups extracted using RuDES results

Source: (a) WS-SP; (b) DFO

management. Our method, called RuDES, includes three modules performing the extraction of dependencies from different perspectives. While the first one is based on amending the existing technique, the second follows knowledge flows between first rule head and second rule body, and the third, introduced in a previous work, uses a correspondence between the direction showed by the process type used to develop the rule base and the directions of knowledge flows expressed in the fact base. The results of our method, compared to existing works, proved that our proposal has improved the existing results by eliminating a high number of false extracted dependencies and minimizing the number of non-extracted correct ones. In our future work, we intend to ameliorate our results by eliminating more false dependencies. Besides we look for eliminating all inconsistencies that might exist in a rule base (redundancies, conflicts, etc.).

Note

1. https://protege.stanford.edu/

References

Arenas, M., Grau, B.C., Kharlamov, E., Marciuska, S. and Zheleznyakov, D. (2016), "Faceted search over rdf-based knowledge graphs", *Journal of Web Semantics*, Vol. 37, pp. 55-74.

Baget, J.-F., Garreau, F., Mugnier, M.-L. and Rocher, S. (2014), "Extending acyclicity notions for existential rules", *Proceedings of the Twenty-First European Conference on Artificial Intelligence*, pp. 39-44.

Bak, J., Nowak, M. and Jedrzejek, C. (2013), "Graph-based editor for Swrl rule bases", *Ruleml*, Vol. 2.

Boujelben, A. and Amous, I. (2018), "A new method for rules dependency extrac- tion", *22nd International Conference on Knowledge-based and Intelligent Information Engineering Systems*.

Brahim, M.B., Chaari, T., Jemaa, M.B. and Jmaiel, M. (2016), "The sem- spm approach: fine integration of ws-securitypolicy semantics to enhance matching security policies in soa", *Service Oriented Computing and Applications*, Vol. 10 No. 3, pp. 337-364.

Chevalier, J., Subercaze, J., Gravier, C. and Laforest, F. (2016), "Incremental and directed rule-based inference on rdfs", *International Conference on Database and Expert Systems Applications*, pp. 287-294.

Dani, M.N., Faruquie, T.A., Karanam, H.P., Subramaniam, L.V. and Venkat- achaliah, G. (2014), Rule set management. Google Patents, US Patent 8,700,542.

Davtalab, M. and Malek, M.R. (2018), "A spatially aware policy conflict resolu- tion for information services", *Journal of Ambient Intelligence and Smart Environments*, Vol. 10 No. 1, pp. 71-81.

Dolinina, O. and Shvarts, A. (2015), "Algorithms for increasing of the effectiveness of the making decisions by intelligent fuzzy systems", *Journal of Electrical Engineering*, Vol. 3, pp. 30-35.

Fenton, W.G., McGinnity, T.M. and Maguire, L.P. (2001), "Fault diagnosis of electronic systems using intelligent techniques: a review", *IEEE Transactions on Systems, Man and Cybernetics, Part C (Applications and Reviews)*, Vol. 31 No. 3, pp. 269-281.

Hassanpour, S., OConnor, M. and Das, A. (2010), "Visualizing logical dependencies in Swrl rule bases", *Semantic Web Rules*, pp. 259-272.

Hassanpour, S., OConnor, M.J. and Das, A.K. (2009), "Exploration of Swrl rule bases through visualization, paraphrasing, and categorization of rules", *International Workshop on Rules and Rule Markup Languages for the Seman- tic Web*, pp. 246-261.

Hirst, G., St-Onge, D., *et al.* (1998), "Lexical chains as representations of context for the detection and correction of malapropisms", *WordNet: An Electronic Lexical Database*, Vol. 305, pp. 305-332.

Huhns, M.N. and Acosta, R.D. (1988), "Argo: a system for design by anal-ogy", *Artificial Intelligence Applications, 1988, Proceedings of the fourth Conference on*, pp. 146-151.

Katta, N., Alipourfard, O., Rexford, J. and Walker, D. (2016), "Cacheflow: dependency-aware rule-caching for software-defined networks", *Proceedings of the Symposium on Sdn Research*, p. 6.

Khattak, A.M., Khan, W.A., Pervez, Z., Iqbal, F. and Lee, S. (2016), "Towards a self adaptive system for social wellness", *Sensors*, Vol. 16 No. 4, p. 531.

Ksystra, K. and Stefaneas, P. (2016), "Formal analysis and verification support for reactive rule-based web agents", *International Journal of Web Information Systems*, Vol. 12 No. 4, pp. 418-447.

O'Connor, M.J. and Das, A. (2006), The swrltab: an extensible environment for working with swrl rules in protégé-owl.

Ristoski, P. and Paulheim, H. (2015), "Visual analysis of statistical data on maps using linked open data", *International Semantic Web Conference*, pp. 138-143.

Salfinger, A., Retschitzegger, W. and Schwinger, W. (2014), "Staying aware in an evolving world specifying and tracking evolving situations", *Cognitive Methods in Situation Awareness and Decision Support (cogsima), 2014 IEEE International Inter-Disciplinary Conference on*, pp. 195-201.

Wu, C.-H., Lee, S.-J. and Chou, H.-S. (1994), "Dependency analysis for knowledge validation in rule-based expert systems", *Artificial Intelligence for Applications, 1994, Proceedings of the Tenth Conference on*, pp. 327-333.

Xitao, W., Chunxiao, D., Xun, Z., *et al.* (2014), Compiling minimum incremental update for modular Sdn languages, *Proceeding of the 3rd ACM Sigcomm Workshop on hot Topics in Software Defined Networking*, *Acm press*, *New York, NY*, pp. 193-198.

Yadav, U., Narula, G.S., Duhan, N. and Jain, V. (2016), "Ontology engineering and development aspects: a survey", *International Journal of Education and Management Engineering (Engineering)*, Vol. 6 No. 3, p. 9.

Zacharias, V. and Borgi, I. (2006), "Exploiting usage data for the visualization of rule bases", *Proceedings of the 3rd International Semantic Web User Interaction Workshop swui*, *Citeseer*.

Corresponding author
Abir Boujelben can be contacted at: abeer.bujelben@gmail.com