ISSN 2007-9737

RuDES: a Semantic Method for Rules Dependency Extraction

Abir Boujelben, Ikram Amous

MIRACL laboratory, University of Sfax,
Tunisia

abeer.bujelben@gmail.com, ikram.amous@enetcom.usf.tn

Abstract. Nowadays, a variety of domains take advantage of rule based systems. In practice, such systems are continuously evolving by acquiring new knowledge and including more rules. The automation of the rule bases management has therefore become a must. Such a task becomes more fluent when based on the analysis of dependencies between the rules. This research work, introduces three main novelties while tackling the issue of rules dependencies detection. First, it amends the existing work. Second, it presents a new technique based on following the knowledge flow defined by the ontology. Third, it propounds a new way to represent the rules dependencies. To check the correctness and performance of our work, a prototype is developed and applied on three different ontologies from different fields.

Keywords. Semantics, rules dependency, ontology, rules, OWL, SWRL.

1 Introduction

Over the past two decades, society has been witnessing an increasing rely on rule based systems (RBS). These take advantage from available data in their environments. The rapid growth in domain knowledge is the main reason for the evolution of knowledge and rule bases [17, 19]. The experts need to continuously update the rule base for the added knowledge (see Figure 1). The periodic, and in some cases manual, updates are complex and may cause raising inevitable inconsistencies in rule bases. To ensure the adaptability of RBS knowledge bases to developments, they have been represented by ontologies.

The introduction of ontologies in RBS thus aims to reduce conceptual and terminological confusion, and to tend towards sharing. This fact allows to improve communication, interoperability and re-usability. Despite their capabilities, ontologies must be extended by rule bases.

Thanks to the Semantic Web, ontology has found a standard formalism based on taxonomy. It is a classification of concepts in a hierarchical form allowing them to be classified according to their meanings or significance [13]. This formalism has been implemented by a variety of ontology representation languages, among which we cite RDF (Resource Description Framework), SPARQL (SPARQL Protocol and RDF Query Language) and OWL (Ontology Web Language). In OWL, the concepts classification is achieved through the use of classes and subclasses. Their instances are called instances (or individuals). Semantic relationships allow to link the instances. Ontologies may be improved by sets of business rules stored in rule bases.

A rule allows describing relationships which cannot be described using the description logic used in ontology representation languages. Rule bases allow sharing and reuse existing rules. They allow also improving ontology capacities of expressiveness and inference. Thus, to provide constraints going beyond the notions available from RDFS and OWL languages, rule languages are also standardized. Among which we cite SWRL (Semantic Web Rule Language) [16]. World Wide Web consortium (W3C) has proposed SWRL as a solution for rule-based systems in the semantic web domain.

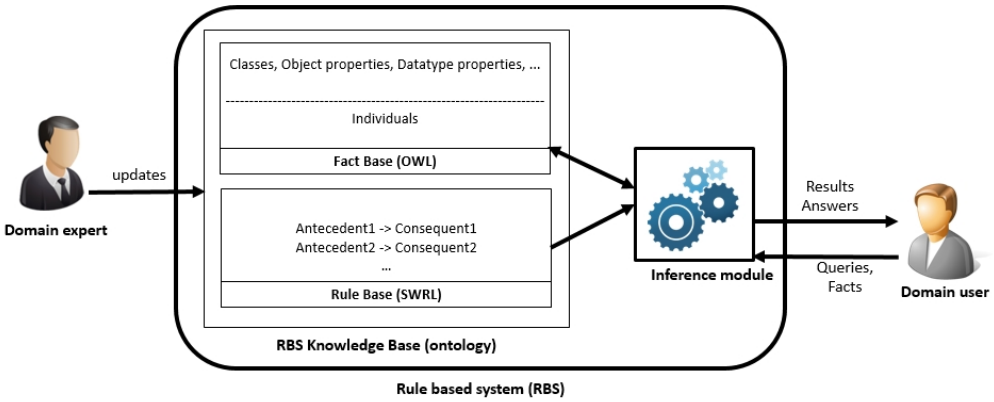

Fig. 1. Architecture of an RBS

Table 1. SWRL atoms types [16]

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)

It allows to apply production rules within the Semantic Web.

SWRL rules are used in association to the facts expressed in OWL in recent research in several fields such as cloud computing [6, 22], the medical field [2, 21, 24] and smart cities [12]. They express, in a formal way, a fact between two classes. The example 1 presents *R-1* an example of rules which indicates that a person over 18 is considered as an adult.

Example 1 *Example of SWRL rules*
R-1: $Person(x), hasAge(x, a), greaterThan(a, 17) \rightarrow Adult(x)$

SWRL rules contain an antecedent part (body), and a consequent part (head).

The antecedent and the consequent consist of positive conjunctions of atoms. Each atom is an expression of the form: P(arg1, arg2,...,argn) where P is a predicate symbol (class, property...) and arg1, arg2,...,argn are arguments (individuals, data values, variables, classes, ...). All types of atoms are displayed in Table 1.

In this paper, we provide a semantic method to extract the dependencies among the elements of a rule base. It represents our first step for the automation of their management. We rely on existing Semantic Web technologies in order to take advantage of the semantics of the system domain. The rest of the paper is organized as follows: Section 2 presents the related work. Section 3 exposes our method and details its performance. In Section 4 we evaluate and discuss the evaluation of our proposal. In Section 5 we present a conclusion and our future work.

2 Related Work

The extraction of dependency relationships between rules has been studied since the emergence of expert systems. It is an essential task when dealing with issues related to rule bases: representation of rules, generation of the order of execution of rules, detection of inconsistencies, verification of explanations and responses to requests.

Table 2. Summary table of work on dependency relationships extraction

	Method	application domain	Interface	Goals			Evaluation
				Dec. Amel.	Comprh. Repr.	Grouping	
Based on a reference entity	Katta et al., 2016	SDN	Table	+	-	-	Prototype
	Bouker et al., 2012	intelligent systems	Graph	+	-	-	-
Based on data analysis	Xitao et al., 2014	SDN	Table	-	-	-	Prototype
	Dani et al., 2014	rule based systems	not mentioned	-	-	+	scenario +example
	Krötzsch et al. 2013	intelligent systems	Graph	-	+	-	-
	Zacharias et al. 2006	Semantic Web	Graph	-	+	-	Prototype
Syntactic	Dolinina et al. 2015	Intelligent Fuzzy Systems	Graph	+	-	-	-
	Dani et al. 2014	rule based systems	not mentioned	-	-	+	scenario +exemple
	Baget et al. 2014	data management	Graph	+	-	-	-
	Hassanpour et al. 2009	Semantic Web	Graph	-	+	+	tool
Semantic	Ezekiel et al., 2018	workflow management	Graph	+	+	-	Prototype
	Chavalier et al., 2016	rule based systems	Graph	+	+	+	tool
	Hassanpour et al. 2010	semantic web	Graph	-	+	+	tool

Dec. Amel. : Decision Amelioration
Comprh. Repr. : Comprehensive Representation
SDN : Software Defined Networks

In the literature, the search for dependencies between the rules is carried out in different ways. Some methods refer to a criterion. Others analyze usage data from the rule base. Other methods are based on the extraction of dependencies between the atoms of the antecedent and of the consequence of each pair of rules.

2.1 Methods based on a Reference Entity

These methods allow the extraction of dependencies between rules based on a reference entity (a pattern, a rule, etc.). [18] propose a system to improve rapid decision-making in software defined networks. The proposed method defines different patterns for the basic rules. The analysis of the dependencies between the rules is thus ensured by analyzing these patterns. The method is based on the organization of the rules in a hierarchical form. Thus, the authors defined dependencies in the form of direct dependencies and complex dependencies. These notions made it easier to update the dependency graph when deleting a rule. [5] propose an approach to maintain rule bases.

They represented them with a dependency graph according to calculated dominance criteria. To do this, they propose to define a dominant rule for the base, and then calculate the degree of dominance of the latter for each of the other elements of the base. This allows to possible to define the succession relationships between the elements of the base. [28] present a tool called S2REd to assist in the development of a rule base. Their main aim is to facilitate the creation, editing and understanding of rules by representing each rule in a graph. The proposed tool offers the end user semantic assistance by modelling the knowledge represented by the rule processed and that expressed by the other elements of the rule base. This is based on loading (or defining) and matching meta-model rules.

2.2 Methods based on the Analysis of Usage Data

These methods are based on analyzing the results of the rule base application on one or more fact bases.

[25] introduce a Framework to modify the transfer policies installed in the distributed switches of a network. The authors extracted the dependency relationships between the rules to eliminate unnecessary updates. Dependency graph generation is achieved incrementally along the compilation process.

In [20], the authors present an approach to ensure a clear visualization of the rule bases. Their proposal refers to usage data provided to find frequently used rules and highlight their dependencies. [9] propose a system based on dependencies between rules in order to manage large bases of rules. These dependencies are determined as a function of the execution frequencies of the rules calculated by the application of the rules on a dataset. [27] propose an approach for the visualization of rule bases using their usage data which take the form of proof trees. This approach mainly aims to help the user while evaluating the quality of the rule base and while modifying it. The collection of usage data is done through requests introduced either during the tests, either during the system specification phase or during its use.

2.3 Methods based on Consequent-Antecedent Analysis

These methods are the most used. They are based on the fact that if a rule *Ry* depends on a rule *Rx*, then *Rx* must provide data which will be used by *Ry*. These methods therefore analyze the relationships between the consequent atoms of the first rule and the antecedent atoms of the second. They are divided into two groups: *syntactic methods* and *semantic methods*. Syntactic methods seek syntactic dependency relationships. They are then based on relations of equality and equivalence between the atoms predicates. Semantic methods seek semantic dependency relationships. Indeed, they have improved syntactic methods by adding the use of hierarchical relationships defined between entities referenced by the predicates of atoms.

Slider-p [8] is a reasoner proposed to manage the data of a dynamic knowledge base. It is based on a rule dependency graph.

[10] suggest reducing the decision-making time based on grouping rules and variables, and on the dependencies extracted between the rules. In this work, the extraction of dependencies is based on the fact that some consequent atoms of the first rule are invoked in the antecedent of the second. In [1], the authors have based their proposal on dependencies between the elements of the rule base. They proposed a tool ensuring the response to the queries posed to ontologies. Dependency extraction is based on the search for equality relations between the atoms predicates and on their variables unification. This work uses the notion of k-dependency to check if the dependency between two rules can last.

[15] offer a rule base representation tool called Axiomé. Axiomé is based on referencing the same classes and the same object properties without taking into account the analysis of the variables. In 2010, the second version of Axiomé was proposed in [14] to enrich the first version by analyzing the domain and the range of object property type atoms. Hierarchical relationships between classes and between properties defined in the ontology are also considered.

[11] propose a bottom-up approach to govern workflows [1] in business processes. This work presents a model for workflows components built based on the dependency graphs between the consequents and the antecedents of the business rules. [23] propose an approach which exploits dependency graphs to represent solutions to constraint systems. They are also used in this same work to represent the execution of protocols by detecting the causal dependencies between the rules.

Our study of the art allowed us to draw Table 2. We classified these methods according to the aim of each work. The first type of methods are based on the reference to an entity, the second type of methods are based on the analysis of usage data. The first type of method did not provide an understandable representation of the base while the second type of method approach did not improve the speed of decision-making.

[1] A sequence of operations taking the form of an activity model for organizing resources into processes that transform materials, provide services or process information.

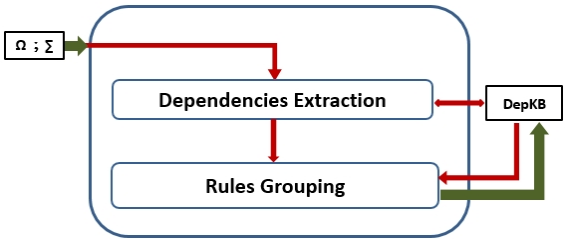

Fig. 2. RuDES architecture

The third type of approach for extracting dependency relationships are syntactic methods. These methods analyze the dependency relationship between the rule base elements by analyzing the dependency relationship between their atoms. The last type of methods are semantic ones. They actually represent an extension of syntactic ones. Syntactic and semantic methods failed to provide a sufficient number of correct dependencies.

Overall, we noticed that the representations of the base provided by existing work are not clear. In addition, the results of dependency relationships extraction are not sufficient to be able to properly manage a large rule base. Confronted with these problems, we propose a method for improving the results of extracting dependency relationships. Our method provides an understandable representation adapted to the needs of the expert.In the following section, we present a method to deal with the cited gaps.

3 RuDES Method

The main goal of our method, called RuDES for **Ru**les **D**ependency **E**xtracion base on **S**emantics, is to extract the dependency relationships between a rule base elements. Its general architecture is provided in Figure 2. It takes as input the ontology Ω and its associated rule base Σ. Our method acts in two main steps : dependencies extraction and groups construction. It returns the dependency relationships that may exist between the elements of Σ and save them in the knowledge base DepKB (for **Dep**endencies **K**nowledge **B**ase). Then, based on these relationships, it groups the rules. The constructed groups are saved back in DepKB.

During dependencies extraction (see Algorithm 1), at every iteration, RuDES processes a couple of rules (Rx, Ry) from the base Σ. This pair goes through the first two sub-modules for the analysis of the dependency relationship that may exist between rule Rx and rule Ry. Each of these sub-modules uses a different technique from the other to provide a confidence degree DC_i, indicating the certainty degree of the extracted relationship, and a direction $direction_i$ indicating the direction of the relation ($D_{Rx \rightarrow Ry}$; $D_{Ry \rightarrow Rx}$). We note that, during measuring confidence degrees, we follow the principle of keeping the strongest cause among all the causes which participated in the existence of the dependency between the rules.

The couples *($direction_i$, DC_i)* generated by each of the two modules represent the input of the sub-module *Dependency Filtering* which will retain the couple *(direction, DC)* having the most appropriate values for this dependency relationship. Finally, the *Rules Grouping* module classify the rules into groups based on the dependency relationships going to and from each rule.

Algorithm 1 Dependency relationships Extraction

Require: Σ : rule base
Require: depKB : dependency relationships knowledge base
Ensure: depKB : populated dependency relationships knowledge base
1: **Var**
2: DC1, DC2, DC : real
3: direction $\in \{ D_{Rx \rightarrow Ry}; D_{Ry \rightarrow Rx} \}$
4: dep(Rx,Ry) : dependency relationship between Rx and Ry
5: **Begin**
6: **for** {Rx, Ry} $\in \Sigma$ **do**
7: DC1, DC2 $\leftarrow$ 0
8: direction $\leftarrow$ null
{**Analyzing dependency relationship between Rx and Ry**}
9: DC1 $\leftarrow$ analyse dependency from Rx to Ry
10: DC2 $\leftarrow$ analyse dependency from Ry to Rx
{**Dependencies Filtering**}
11: (direction, DC) $\leftarrow$ Filtering(DC1,DC2)
12: dep(Rx,Ry) $\leftarrow$ (direction, DC)
{**Dependency Saving**}
13: save dep(Rx,Ry) in DepKB
14: **end for**
15: **End.**

3.1 Analyzing Dependency Relationships

RuDES begins its processing with analyzing the dependency relationship between two rules Rx and Ry using two different techniques. Each of them is implemented in a sub-module. The first is called CADA for ***C**onsequent-**A**ntecedent analysis based on **D**ifferent-type **A**toms*. As its name suggests, this sub-module analyzes the dependency from Rx to Ry by analyzing the dependency between the consequent of the first rule and the antecedent of the other. This stems from the fact that if Ry depends on Rx, certainly Ry will make use, in its antecedent, of knowledge inferred by consequence from Rx. In addition, this sub-module is also based on the analysis of atoms having different types.

The second sub-module is called CCA for **C***onsquent*-**C***onsequent* **A***nalysis*. It is based on a principle different from the previous sub-module. It mainly uses the relationships between the facts inferred by the first rule and the facts inferred by the second. This sub-module, unlike the other sub-module, is not based solely on facts expressed by the ontology Ω. It also uses knowledge based on the rule base Σ itself [4].

3.2 Dependencies Filtering

After analyzing the dependency relationship between two rules Rx and Ry, each of the sub-modules CADA and CCA provide a dependency which may be different from the other. The Dependencies Filtering sub-module role consists in keeping the most appropriate dependency. This choice is made using the confidence degrees DC_i (i $\in$ {CADA, CCA}) of each dependency and the sub-module that generated it (Algorithm 2).

The highest coefficient ($c_{\_CCA}$ = 1) is assigned to the CCA sub-module because it is based on knowledge extracted from the definition of the ontology Ω and those extracted from the rule base

Algorithm 2 Dependencies Filtering

Require: dep$_{CADA}$=(direction$_{CADA}$,DC$_{CADA}$) : dependency generated by CADA
Require: dep$_{CCA}$=(direction$_{CCA}$,DC$_{CCA}$) : dependency generated by CCA
Ensure: dep=(direction,DC) : saved dependency
1: **VAR**
2: x1, x2: real
3: direction : dependency direction
4: **Begin**
5: c$_{_CCA}$ ← 1
6: c$_{_CADA}$ ← 0.8
7: x1 ← DC$_{CADA}$ * c$_{_CADA}$
8: x2 ← DC$_{CCA}$ * c$_{_CCA}$
9: **if** Max(x1,x2)=x1 **then**
10: direction ← direction$_{CADA}$
11: DC ← DC$_{CADA}$
12: **else**
13: direction ← direction$_{CCA}$
14: DC ← DC$_{CCA}$
15: **end if**
16: **End.**

Σ. The other two sub-module is based only on knowledge extracted from Ω. Then its coefficient c$_{_CADA}$ is lower than c$_{_CCA}$ (c$_{_CADA}$=0.8 and c$_{_CCA}$=1). These coefficients are chosen after carring out some experiments that proved them.

3.3 Rules Grouping

Most rule bases decompose into groups of rules, each of which is responsible for accomplishing a task. As example we cite the rule base associated to the ontology *Vehicul Ontology* [3] which includes a group of rules responsible for determining a vehicle position in relation to others. It also includes another group responsible for proposing actions appropriate to a danger situation, etc. On the other hand, there are rule bases where each task is accomplished by one rule.

We cite the example of the base associated to the ontology *Autism Ontology* [26]. The elements of its rule base define how each phenotype should be derived from a set of clinical outcomes.

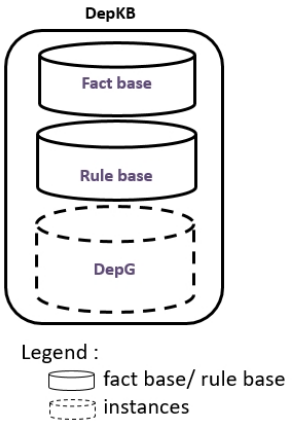

Fig. 3. DepKB architecture

Each rule is based on knowledge defined in the ontology to infer new knowledge without using any other rule.

In our proposal, the *Rules Grouping* module build the groups in order to facilitate and optimize inference processes and rule base verification. Rules Grouping module is based on dependency relationships already extracted. The rules belong to the same group take, as input, facts that relate to the same classes. They also produce facts concerning the same classes. Thus, the rules of the same group have the same incoming edges and the same outgoing edges. The groups are saved in DepKB.

3.4 Saving Rule Dependencies

The DepKB knowledge base is represented by an ontology. As shown in Figure 3, it is composed of a fact base, a rule base and an instance base. The fact base is detailed in Table 3. It has three classes, three object properties and one data property. A dependency relationship is represented using the classes *Dependency* and *Rule*, connected by the object properties *beginsFrom* and *endsAt*. The confidence degree of a dependency relationship is expressed using the data property *hasWeight* (see Figure 4).

Each instance of the *Dependency* class has only one instance of the properties *beginsFrom*, *endsAt* and *hasWeight*.

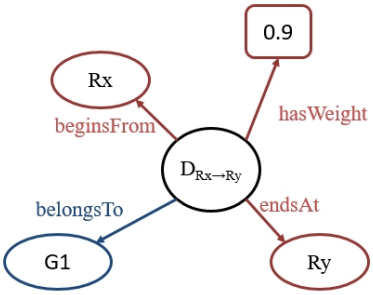

Fig. 4. $D_{Rx \rightarrow Ry}$ representation in DepKB

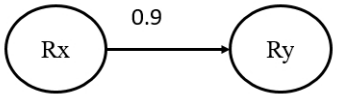

Fig. 5. Simplified representation of $D_{Rx \rightarrow Ry}$

Dependency(dep1) , beginsFrom(dep1,Rx) , endsAt(dep1,Ry) ,

Dependency(dep2) , beginsFrom(dep2,Ry) , endsAt(dep1,Rz) ,

Dependency(dep3) → beginsFrom(dep3,Rx) , endsAt(dep3,Rz)

Fig. 6. RTrans: rule expressing the transitivity property of dependency relationships

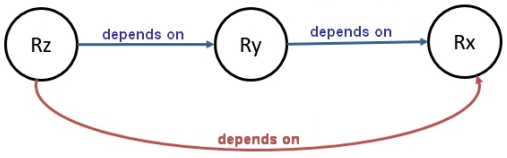

Fig. 7. Transitivity property of dependency relationships

Rule groups are represented using the class *Group* linked to the class *Rule* by the object property *belongsTo* (see Figure 4). Each instance of the *Rule* class has only one instance of the *belongsTo* property. In Figure 4 we show show to represent the dependency relationship $D_{Rx \rightarrow Ry}$ in the ontology DepKB. This representation can be simplified for the experts as presented in Figure 5.

The DepKB ontology also includes the rule *R-Trans*, represented in Figure 6. It expresses transitivity property of dependency relationships. This rule can be read as follows: if there is a rule *Ry* which depends on a rule *Rx*, and if there is a rule *Rz* which depends on the rule *Ry*, then the rule *Rz* depends on the rule *Rx* (see Figure 7). DepKB also includes the DepG part which contains all the instances of the *Dependency* class representing the saved dependency relationships.

4 Evaluation

In this section, we present the experiments results we carried out on RuDES method. We start by evaluating each of the dependency extraction techniques, then we present the evaluation of the entire method.

4.1 Ontologies used for Evaluation

We applied our approach on rule bases associated to three ontologies from various fields: Web service security field, medical field and road safety field.

4.1.1 WebService-SecurityPolicy Ontology

WebService-SecurityPolicy (WS-SP) [7] is an ontology from Web service security field. Its primary role is to specify and match web service security policies between a requester and a web service provider. This base includes 79 rules. The developers of this rule base divided it into five groups, each of which has a specific task (see Figure 8a). These tasks must be performed in a specific order. The first group instantiates the semantic relationships that may exist between the requester's atomic security properties and those of the service provider.

Table 3. DepKB fact base

Classes	· Class: Rule Subclass of: owl: Thing · Class: Dependency Subclass of: owl: Thing · Class: Group Subclass of: owl:Thing
Object properties	· Property:beginsFrom (Dependency, Rule) Restricion: functional · Property:endsAt(Dependency, Rule) Restriction: functional ·Property: belongsTo (Rule, Group) Restriction: functional
Data property	· Property: hasWeight (Dependency, real) Restriction: functional

Table 4. WS-SP ontology overview

	WebService-SecurityPolicy
Domain	Network Service Security
# rules	79
# groups	5

Table 5. Overview of the DFO ontology

	Diabetic Food Ontology
Domain	Medical
# rules	24
# groups	4

Table 6. Overview on the VO ontology

	Vehicul Ontology
Domain	Road safety
# rules	77
# groups	3

The elements of the second group instantiate the semantic relationships that can exist between the complex security properties of the requester and those of the service provider. Then the third group matches their security assertions. The fourth group makes the connections between their security alternatives. Finally, the last group decides the degree of security correspondence between their assertions (Perfect match, close match, possible math and no match).

4.1.2 Diabectic Food Ontology

Diabectic Food Ontology (DFO) is an ontology developed in collaboration with an expert using the ontology ontFood of the BioPortal [2] of the National Center for Biomedical Ontology of the United States. The purpose of the DFO ontology is to specify the ingredients of daily meals for diabetic patients. The rule base associated to this ontology includes 24 elements. This rule base has four groups (see Figure 8b). The first calculates the patient's body mass index (BMI). Then the second group specifies this index type. Then, the third group indicates the period during which the specified patient must take his meal. Finally, the fourth group uses the period and the BMI type to specify the ingredients of the meal that the patient will take.

4.1.3 Vehicul Ontology

Vehicle Ontology (VO) [3] is an ontology intended to be integrated into vehicles travelling on the road. It allows to suggest to a vehicle's driver different actions for each persuaded emergency. This task is achieved based on data acquired from sensors. The rule base associated to the VO ontology includes 77 rules divided into 3 groups (see Figure 8c). The first and second groups allow to deduce respectively the position and the distance of the vehicle with respect to the one it precedes and that it follows on the road. The third group uses facts from the ontology and knowledge inferred from the previous groups to deduce the actions appropriate to the vehicle situation (change speed or change line).

[2] https://bioportal.bioontology.org/resource_index

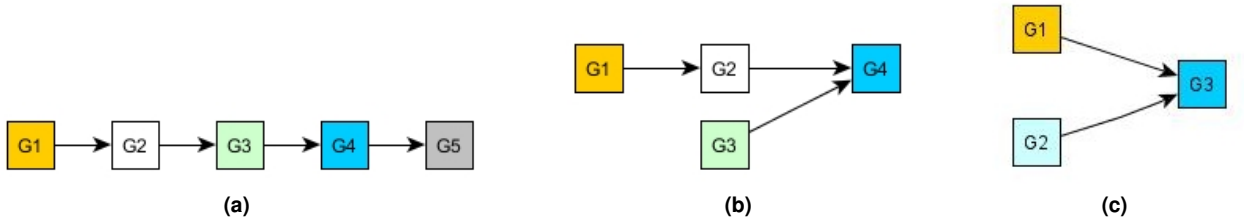

Fig. 8. Rule groups provided by ontologies designers (a:WS-SP, b:DFO, c:VO)

4.2 CADA and CCA Sub-Modules Performance

To evaluate the performance of both dependency extraction sub-modules (CADA and CCA), we applied them separately on the study cases already mentioned. Then we compared the obtained results to those provided by the rule bases developers. Thus, we were able to define the number of correct dependencies, the number of false dependencies and the number of non-extracted (missed) dependencies.

The CADA sub-module extracts dependencies by analyzing pairs of different type atoms. One atom comes from the consequent of the first rule and the other from the second's antecedent. Table 7, Table 8 and Table 9 present the dependency relationship rates found compared to those missed. They also show the rates of false dependencies extracted compared to all of those found.

These tables show that CADA actively contributed to the extraction of the correct dependencies in DFO and VO cases. In return, the number of false dependencies is high. In the WS-SP case, CADA has extracted a small number of correct dependencies. It also extracted a large number of false dependencies compared to the other module. These results show that this sub-module sometimes contributes to the increase in the number of correct dependencies, but it helps to extract a large number of false dependencies. We also noticed that it contributed to detect the dependencies missed by CCA. To take advantage of this sub-module, we eliminated its drawback by the Dependencies Filtering sub-module.

Besides, the CCA sub-module allowed the extraction of a large number of correct dependencies. It generated only a small number of false dependencies in all of the case studies. In the case of the WS-SP database, CCA extracted more than $\frac{2}{3}$ of correct dependencies.

Table 7. Dependencies extracted by CADA & CCA (**Base WS-SP**)

	CADA	**CCA**
correct dep.	1.78 %	94.22 %
missed dep.	98.22 %	5.78 %
false dep.	94.66 %	2.71 %

Table 8. Dependencies extracted by CADA & CCA (**Base DFO**)

	CADA	**CCA**
correct dep.	100 %	55.55 %
missed dep.	0%	44.45%
false dep.	53.04 %	0%

In the cases of the DFO and VO databases, it extracted almost half. This result proves its efficiency. This fact is due to its use of knowledge expressed by the fact base and those expressed by the rule base.

All these results prove that both CADA and CCA participate, in some cases, in increasing the number of correct dependencies. In other cases, they help reduce the number of false ones. This proves that these modules are complementary.

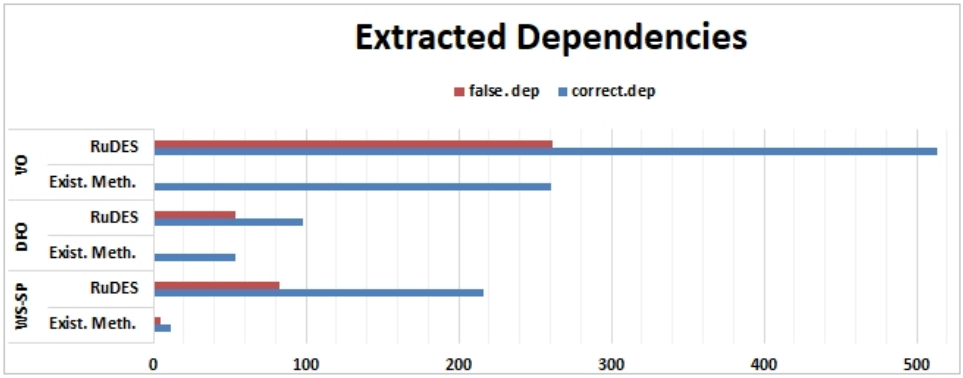

Fig. 9. All dependencies extracted by Existing Methods and RuDES

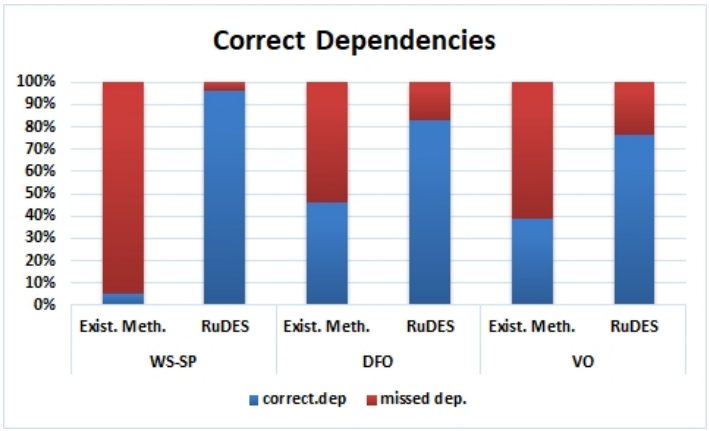

Fig. 10. Correct dependencies extracted by Existing Methods and RuDES

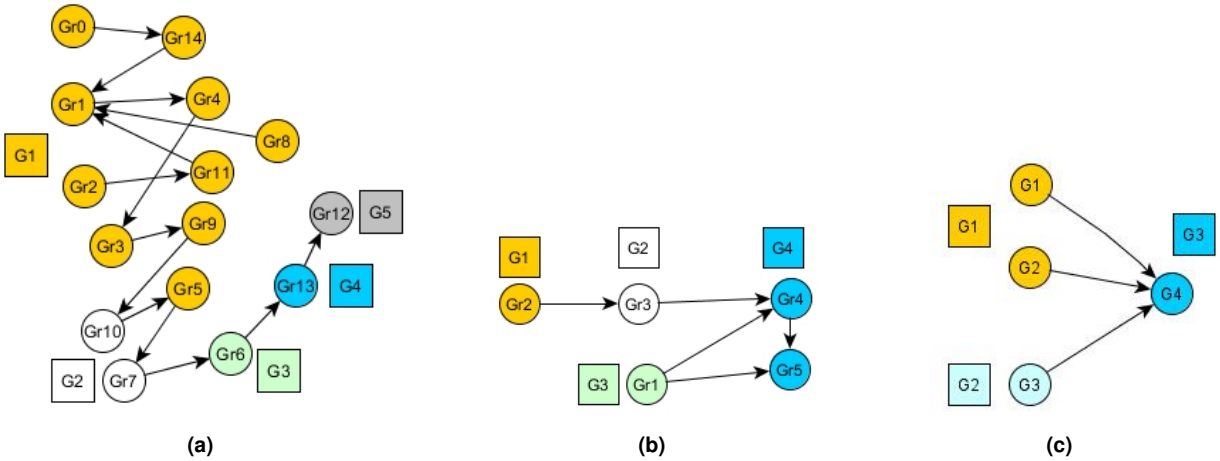

Fig. 11. RuDES Rule groups (a:WS-SP, b:DFO, c:VO)

Table 9. Dependencies extracted by CADA & CCA (**Base VO**)

	CADA	**CCA**
correct dep.	92.8 %	63.24 %
missed dep.	7.2 %	36.76 %
false dep.	33.2 %	0 %

4.3 RuDES Performance

To position ourselves vis-a-vis existing work interested in dependencies extraction, we compared our method to Axiomé. Axiomé is the most complete existing semantic method. It includes all the techniques of semantic ones. It is available as a plugin in Protégé[3], an ontology development environment.

We applied RuDES and Axiomé to the study cases already presented. The results are shown in Figure 9 and Figure 10. These values are computed in the same way as that used to calculate the results presented in the previous section.

Figure 9 presents the numbers of correct dependencies versus the numbers of false extracted ones. Figure 10 shows, for each case, the number of correct dependencies extracted versus the number of missed ones. The figures show that in the case of WS-SP base, the existing methods have extracted a small number of dependencies among which nearly 30% are false. Whereas RuDES has extracted practically all the requested dependencies with almost perfect precision (small number of false dependencies).

These results are due to a large number of WS-SP base rules using atoms whose types are not considered by existing methods. In the case of the DFO and VO bases, the rule bases do not include these atoms. Thus, existing methods did not extract false dependencies in DFO case and a small number in the VO case. They managed to extract almost half of the number of dependencies requested.

[3]https://protege.stanford.edu/

RuDES extracted almost all the requested dependencies with a small number of false ones. This results improvement is owing to the facts that RuDES (i) extends the existing methods with analyzing a larger number of atoms and (ii) uses a new technique as shown before.

4.4 Rules Grouping Module Performance

The developers of each of the studied ontologies provided the rules in groups (as described in subsection 4.1). Figure 8a presents the rule groups provided by the WS-SP ontology designers, whereas Figure 8b shows the rule groups provided by the DFO ontology designers. 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 this subsection, we choose to prove the efficiency of our method by comparing 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 11b, 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 11a), 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 all three cases (Figure 11) 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

Integrated into self-adapting and dynamic systems, the rule bases associated to ontologies must be updated with each change relating to the knowledge they represent. This causes the appearance of anomalies which have a noticeable effect on these systems states. Dependency relationships represent a means of managing rule bases. In this manuscript, we have proposed a method called RuDES. Our method combines two different techniques for the extraction of dependency relationships between the elements of a rule base. In this work, we represent these relationships using an ontology we called DepKB. This will allow the experts to consult the state of the rule base by adapting the representation to their needs using available tools for ontology management. The experiments we performed have proved the efficiency of our proposal compared to existing work. In our future work, we aim to ameliorate our method's results using other techniques for dependency relationships extraction.

References

1. **Baget, J.-F., Garreau, F., Mugnier, M.-L., & Rocher, S. (2014).** Extending acyclicity notions for existential rules. *Proceedings of the Twenty-first European Conference on Artificial Intelligence*, IOS Press, pp. 39–44.
2. **Benjemmaa, A., Ltifi, H., & Ayed, M. B. (2019).** Design of remote heart monitoring system for cardiac patients. *International Conference on Advanced Information Networking and Applications*, Springer, pp. 963–976.
3. **Bermejo, A., Villadangos, J., Astrain, J. J., & Cordoba, A. (2013).** Ontology based road traffic management. In *Intelligent Distributed Computing VI*. Springer, pp. 103–108.
4. **Boujelben, A. & Amous, I. (2018).** A new method for rules dependency extraction. *22nd International Conference on Knowledge-Based and Intelligent Information Engineering Systems*, Elsevier.
5. **Bouker, S., Saidi, R., Yahia, S. B., & Nguifo, E. M. (2012).** Ranking and selecting association rules based on dominance relationship. *Tools with Artificial Intelligence (ICTAI), 2012 IEEE 24th International Conference on*, volume 1, IEEE, pp. 658–665.
6. **Brabra, H., Mtibaa, A., Petrillo, F., Merle, P., Sliman, L., Moha, N., Gaaloul, W., Guéhéneuc, Y.-G., Benatallah, B., & Gargouri, F. (2019).** On semantic detection of cloud api (anti) patterns. *Information and Software Technology*, Vol. 107, pp. 65–82.
7. **Brahim, M. B., Chaari, T., Jemaa, M. B., & Jmaiel, M. (2016).** The semspm 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.
8. **Chevalier, J., Subercaze, J., Gravier, C., & Laforest, F. (2016).** Incremental and directed rule-based inference on rdfs. *International Conference on Database and Expert Systems Applications*, Springer, pp. 287–294.
9. **Dani, M. N., Faruquie, T. A., Karanam, H. P., Subramaniam, L. V., & Venkatachaliah, G. (2014).** Rule set management. US Patent 8,700,542.
10. **Dolinina, O. & 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.
11. **Ezekiel, K., Vassilev, V., Ouazzane, K., & Patel, Y. (2018).** Adaptive business rules framework for workflow management. *Business Process Management Journal*.
12. **Fki, Z., Ammar, B., & Ben Ayed, M. (2018).** Risk prediction in smart home care. *2018 IEEE 32nd International Conference on Advanced Information Networking and Applications (AINA)*, IEEE, pp. 866–873.
13. **Gruber, T. (1993).** What is an ontology. *WWW Site http://www-ksl. stanford. edu/kst/whatis-an-ontology. html (accessed on 07-09-2004)*.
14. **Hassanpour, S., OConnor, M., & Das, A. (2010).** Visualizing logical dependencies in swrl rule bases. *Semantic Web Rules*, pp. 259–272.
15. **Hassanpour, S., OConnor, M. J., & 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 Semantic Web*, Springer, pp. 246–261.
16. **Horrocks, I., Patel-Schneider, P. F., Boley, H., Tabet, S., Grosof, B., Dean, M., et al. (2004).** Swrl:

A semantic web rule language combining owl and ruleml. *W3C Member submission*, Vol. 21, pp. 79.

17. **Javed, M., Abgaz, Y. M., & Pahl, C. (2013).** Ontology change management and identification of change patterns. *Journal on Data Semantics*, Vol. 2, No. 2-3, pp. 119–143.
18. **Katta, N., Alipourfard, O., Rexford, J., & Walker, D. (2016).** Cacheflow: Dependency-aware rule-caching for software-defined networks. *Proceedings of the Symposium on SDN Research*, ACM, pp. 6.
19. **Khouri, S. & Bellatreche, L. (2017).** Design life-cycle-driven approach for data warehouse systems configurability. *Journal on Data Semantics*, pp. 1–29.
20. **Krötzsch, M. & Rudolph, S. (2013).** On the relationship of joint acyclicity and super-weak acyclicity. Technical report, Tech. rep. 3037, Institute AIFB, Karlsruhe Institute of Technology.
21. **Laadhar, A., Ghozzi, F., Megdiche, I., Ravat, F., Teste, O., & Gargouri, F. (2019).** Partitioning and local matching learning of large biomedical ontologies. *Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing*, ACM, pp. 2285–2292.
22. **Labidi, T., Mtibaa, A., Gaaloul, W., Tata, S., & Gargouri, F. (2017).** Cloud sla modeling and monitoring. *2017 IEEE International Conference on Services Computing (SCC)*, IEEE, pp. 338–345.
23. **Milner, K. (2018).** *Detecting the misuse of secrets: foundations, protocols, and verification*. Ph.D. thesis, University of Oxford.
24. **Sbissi, S., Mahfoudh, M., & Gattoufi, S. (2019).** Mapping clinical practice guidelines to swrl rules. *World Conference on Information Systems and Technologies*, Springer, pp. 283–292.
25. **Xitao, W., Chunxiao, D., Xun, Z., et al. (2014).** Compiling minimum incremental update for modular sdn languages. *Proc of the 3rd ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking. New York: ACM Press*, pp. 193–198.
26. **Young, L., Tu, S. W., Tennakoon, L., Vismer, D., Astakhov, V., Gupta, A., Grethe, J. S., Martone, M. E., Das, A. K., & McAuliffe, M. J. (2009).** Ontology driven data integration for autism research. *2009 22nd IEEE International Symposium on Computer-Based Medical Systems*, IEEE, pp. 1–7.
27. **Zacharias, V. & Borgi, I. (2006).** Exploiting usage data for the visualization of rule bases. *Proceedings of the 3rd International Semantic Web User Interaction Workshop SWUI. Citeseer*.
28. **Zetta, T., Kontopoulos, E., & Bassiliades, N. (2011).** S 2 red: A semantic web rule editor. Technical report, International Hellenic University (Operation – Development).

Article received on 12/09/2018; accepted on 06/01/2020.
Corresponding author is Abir Boujelben.