Study on Pear Diseases Query System Based on Ontology and SWRL

Qian Sun and Yong Liang

School of Information Science and Engineering,
Shandong Agricultural University, Taian, 271018, China
{applesq,yongl}@sdau.edu.cn

Abstract. This paper studied the construction of Pear Diseases Domain Ontology (PDDO), and the realization of query system based on PDDO and SWRL. First, an approach to build PDDO based on SWRL was proposed, which consists of confirming core concepts, adding the properties of concepts and the relationships between concepts, adding the instances of concepts, representing domain ontology, adding SWRL rules and reasoning. Then the query system model and implementation algorithm were given. The query system, which integrates SWRL with Jess reasoning engine based on the SWRL rules, realized disease query, instance query, and diagnosis query by Protégé-OWL API. The query system realized excavating implicit relationships and renewing PDDO relative to previous system. Through the query system based on reasoning pear diseases knowledge can be obtained from PDDO according to user needs, furthermore, new and inferred knowledge are written to the PDDO owl file.

Keywords: Protégé, SWRL, Protégé-Owl API, pear; ontology, query, JESS.

1 Introduction

With the development of owl ontology language, more and more knowledge systems based on domain ontology are developed. The development of knowledge system includes knowledge representation, storage, reasoning, query and so on, in which query and reasoning are the key technologies, through them knowledge can be obtained from ontology [1]. In order to realize pear diseases knowledge query based on reasoning, the query system based on PDDO and SWRL is studied in this paper.

2 Technologies and Tools on Ontology

2.1 Protégé

Protégé is a open source ontology editor, which allows user to model ontology. The Protégé platform provides two main ways of modeling ontology via the protégé-frames and protégé-owl editors. Protégé ontology can be exported into a variety of formats including OWL [2] ,RDF(S) [3], and XML Schema. Further more, Protégé

D. Li and Y. Chen (Eds.): CCTA 2013, Part II, IFIP AICT 420, pp. 24–33, 2014.

can be extended by way of a plug-in architecture and a Java-based Application Programming Interface (API) for building knowledge-based tools [4].

2.2 Protégé-OWL API

The Protégé-OWL API is an open source Java library for the Web Ontology Language and RDF(S). The API provides classes and methods to load and save OWL files, to query and manipulate OWL data models, and to reason. Furthermore, the API is optimized for the implementation of graphical user interfaces [5].

2.3 SWRL

SWRL is a Semantic Web Rule Language based on a combination of the OWL DL and OWL Lite sublanguages of the OWL Web Ontology Language with the Unary/Binary Datalog RuleML sublanguages of the Rule Markup Language. SWRL includes a high-level abstract syntax for Horn-like rules in both the OWL DL and OWL Lite sublanguages of OWL. A model-theoretic semantics is given to provide the formal meaning for OWL ontology including rules written in this abstract syntax [6].

2.4 JESS

Jess is a Java-based rule engine. Jess system consists of a rule base, fact base, and an execution engine. It has been used in Protégé-based tools, e.g. SWRLJessTab[7], SweetJess, JessTab.

3 Construction of Pear Diseases Domain Ontology Based on SWRL

At present, ontology construction methodologies have not been standardized, there are numerous frequently quoted approaches. In this paper, an approach for building domain ontology based on SWRL is proposed. The process of it consists of the following phases:

3.1 Confirming Core Concepts

In this case, core concepts of pear diseases domain are collected according to features needed by diseases diagnosis and pear diseases. Firstly, “Pear-tree” is selected as the first concept, and then the concepts relating to “Pear-tree” are selected. Table1 gives the names and explanations of these concepts.

Table 1. The general concepts of PDDO

Name		Explanation
Disease		Refers to the categories of the pear diseases
Growing-period		Reflects the time that pear elapses from seeding to mature
Part		Refers to the diseased parts of pear, instances: root, fruit
Pathogen	Feature	Reflects the features of pathogens. Instance: black small point
	P-kind	Refers to the kinds of the pathogens that cause pears fall ill
Pear-tree		Refers to pears
Symptom	Color	Reflects the variety of colors that the diseased parts of pears change to
	Shape	Reflects the variety of shapes of the spots.
	Dynamic-symptom	Reflects the symptoms of the diseased pears , instance: rotting.

3.2 Adding the Properties of Concepts and the Relationships between Concepts

In this case, every core concept has different properties to be added, for example: "describe" is added as the property of "disease" to describe the features of diseases. In addition, the relationships between concepts are added. Table2 gives the relationships between "Pear-tree" and other concepts.

Table 2. The relationships between "Pear-tree" and other concepts

Name	Explanation
At-part	Reflects the relationship between "Pear-tree" and "part"
Has-ds	Reflects the relationship between "Pear-tree" and "Dynamic-symptom"
Has-c	Reflects the relationship between "Pear-tree" and "Color"
Has-disease	Reflects the relationship between "Pear-tree" and "Disease"
Has-shape	Reflects the relationship between "Pear-tree" and "Shape"
Has-feature	Reflects the relationship between "Pear-tree" and "Feature"
Has-pathogen	Reflects the relationship between "Pear-tree" and "P-kind"
At-period	Reflects the relationship between "Pear-tree" and "Growing-period"

3.3 Adding the Instances of Concepts

It is necessary for building domain ontology to supply the instances of concepts. For example: in this case, "change color" , "die", "dry-rot", " falling-off", " putrescence" , "rotting", "spotting", "wilting " are added as instances of "Dynamic-symptom". "branch", " fruit", "fruit-stem", "leaf", and "root" are added as instances of "Part". Names of pear diseases are added as instances of "Disease", for example:. "Steptomyces-scabis", "Pear-Brown-blight", "Pear-Rust", "Pear-Valsa-Canke",

"Pear-anthracnose", "Pear-blackPedicle-disease", Pear-black-shank", " Pear-black-spot", "Pear-blight" and so on.

3.4 Representing Pear Diseases Domain Ontology

In this case, protégé 3.4.8 is selected as the developing tool, so the PDDO can be represented by OWL. Further illustrate below:

Firstly, according to the confirmed core concepts, corresponding classes of PDDO are created by using protégé. Classes structure of PDDO can be shown by OWLVizTab in protégé (see Fig.1). Secondly, data properties and object properties of classes are added, relationships between concepts can be represented by way of adding object properties. For example: Fig.2 represents the relationships between "Pear-tree" and other classes by JambalayaTab. Finally, instances of classes are added. After that, an owl file (tree.owl) is created by protégé.

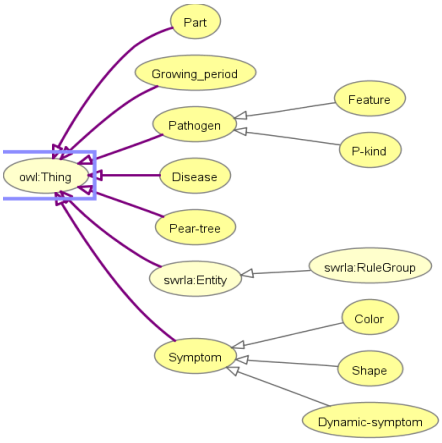

Fig. 1. Classes structure of PDDO

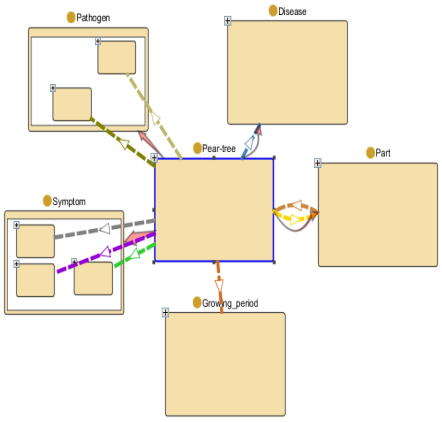

Fig. 2. Relationships between "Pear-tree" and other classes

3.5 Adding SWRL Rules and Reasoning

3.5.1. SWRL Editor

The SWRL Editor is an extension to Protégé-OWL, which supports the interactive editing of SWRL rules. The editor can be used to create, edit, read and write SWRL rules .It is accessible through the SWRLTab within Protégé-OWL [8] (see Fig.3).

3.5.2 Building SWRL Rules Library

A rule axiom consists of an antecedent (body) and a consequent (head), each of which consists of a set of atoms. Atoms can be of the form C(x), P(x,y), sameAs(x,y) or differentFrom(x,y), where C is an OWL description, P is an OWL property, and x,y are either variables, OWL individuals or OWL data values. where both antecedent

and consequent are conjunctions of atoms written a1 ∧ ... ∧ an. Variables are indicated using the standard convention of prefixing them with a question mark (e.g., ?x) [9].

For example: the symptoms of “rust of pear” are as follow: it mainly destroys leaves, makes leaves show circle yellow scabs, further more some yellow acicular points on scabs. The description of this symptom by SWRL rule is as follow:

Pear-tree(?x) ∧ At-part(?x, leaf) ∧ Has-ds(?x, spotting) ∧ Has-c(?x, yellow) ∧ Has-shape(?x, circle) ∧ Has-feature(?x, yellow-acicular-small-point) → Has-disease(?x, Pear-Rust)

Using this syntax, some rules relating to pear diseases diagnosis are created by the SWRL Editor, so that SWRL rules library based on PDDO is built. (see Fig.3).

3.5.3 Reasoning Based on SWRL Rules

The SWRL Editor itself does not perform any inference. However, a bridge mechanism is provided to allow interoperation with rule engines [10]. At present, the Jess rule engine is supported, and is accessible through the SWRLJessTab that is a plug-in to the SWRLTab in Protege-OWL. It supports the execution of SWRL rules and provides a graphical interface to interact with the SWRLJessBridge. After editing rules, “OWL+SWRL->JESS” button can be pressed, which can transfer all SWRL rules and Pear diseases OWL knowledge to the Jess rule engine. When "Run Jess" button is pressed, Jess will run its inference engine and possibly generate new knowledge[11]. At that point, this inferred knowledge can be passed back to the owl file (tree.owl) by pressing the "Jess->OWL" button (see Fig.3). In this case, reasoning based on rules realizes building new relationships between instances of “Pear-tree” and “Disease”. For example: a new relationship between “pear3” the instance of “Pear-tree” and “Pear-Rust” the instance of “Disease” is inferred by the above SWRL rule ,that is “pear3” Has-disease “ Pear-Rust” (see Fig.4).

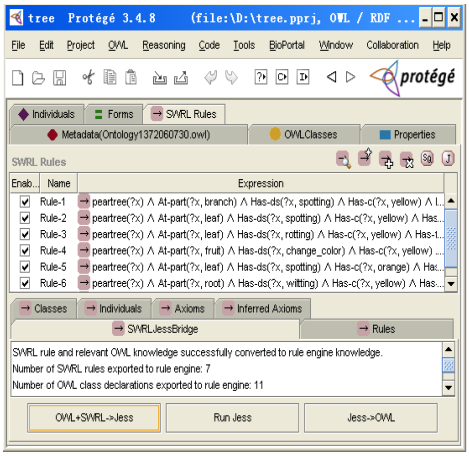

Fig. 3. SWRL Editor

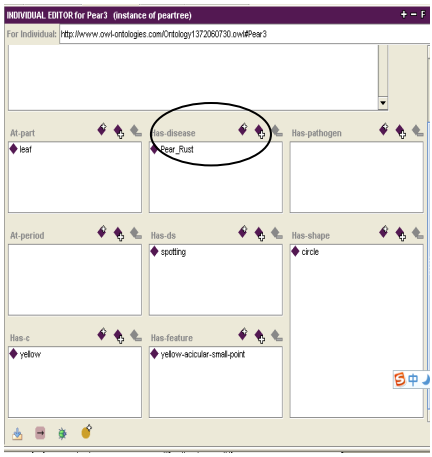

Fig. 4. New relationship inferred

4 Design and Realization of Query System Based on PDDO and SWRL

4.1 Architecture of Query Model

In order to design query system, the model of it is built at first .The model consists of six modules, which are shown in the Fig.5. Further illustrate below: By using interactive query interface, users can customize query conditions, which are sent into query processor. The functions of the query processor are executing corresponding algorithms and calling Protégé-OWL API methods according to the given query conditions. Parsing ontology file, it is a way to access and draw information from ontology file. Jess rule engine can transfer all SWRL rules and Pear diseases OWL knowledge to the engine, run its inference engine, and pass inferred knowledge back to the owl file .Finally, the query results can be outputted in the visual interface.

4.2 Parsing the Pomology Domain Ontology

In this study Protégé-OWL API is used to parse the PDDO. An OWLModel is created through the ProtegeOWL.createJenaOWLModel(), which can load an OWL files, then the resources can be created, queried, deleted through the methods of OWLModel .In this case , owlModel1 is created to load “tree.owl” file. The methods of Protégé-OWL API that are used in this study are listed below: the certain OWLClass can be obtained by calling of getOWLNamedClass(), the certain RDFIndividual can be obtained by calling of getRDFIndividual(), the certain RDFProperty can be obtained by calling of getRDFProperty(), calling getUserDefinedOWLObjectProperties() can get all object properties collection.

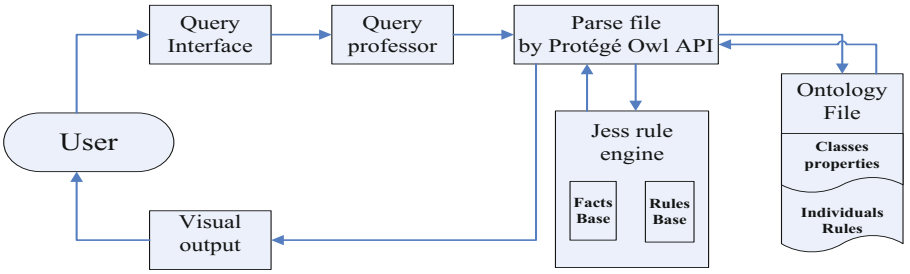

Fig. 5. Flow-process diagram of query model

4.3 Realization of Query System Based on PDDO and SWRL

This query system realizes three kinds of query, which are disease query, instance query, and diagnosis query. Further illustrate below:

1) Disease Query

The instances of "Disease" class are listed on the left side of visual query interface, and the TextArea is on the right side. While user selects one from the list, the introduction of the certain pear disease is displayed in the TextArea (see Fig.6). Details of the method are as follows:

Firstly, the instances list of "Disease" class are realized by using JList. GetOWLNamedClass() is called by the defined owlModel1 to obtain the OWLClass "Disease", and then getInstances() is called by it to obtain all instances of OWLClass "Disease", finally, the instances are saved into a collection, iterated, and added into the listModel of JList. The algorithm is as follows:

```
Collection ins=Disease.getInstances();
DefaultListModel listModel = new DefaultListModel();
for(Iterator i4=ins.iterator();i4.hasNext();)
      {     OWLIndividual in=(OWLIndividual ) i4.next();
            String t=in.getLocalName();
            listModel.addElement(t); }
```

Secondly, addMouseListener () is added to answer that the user clicks one from the list. Calling GetRDFIndividual() that OWLModel provides to change the instance selected by user into the RDFIndividual, and then getPropertyValue （) is called by the certain RDFIndividual to obtain the value of the "describe" data property.

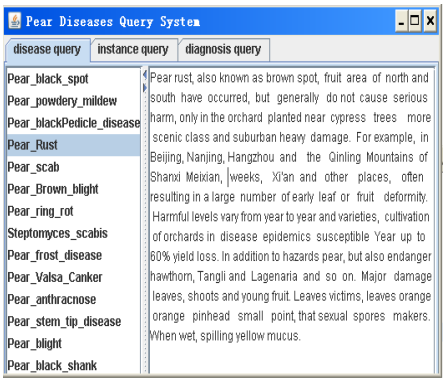

Fig. 6. Disease query interface

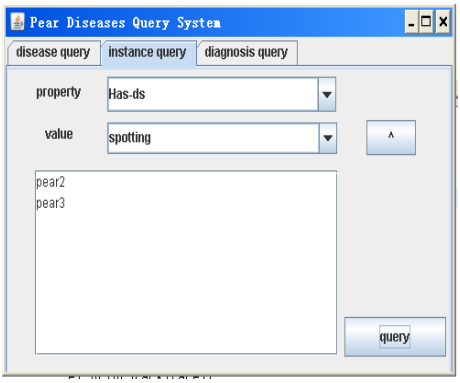

Fig.7. Instance query interface

2) Instance Query

Interactive instance query supports query on the instances of classes based on SWRL rule. User can select a property from properties of classes through the "property" JComboBox, and then select a value of the certain property through "value" JComboBox. If the "∧" button is pressed , another property and it's value can be selected. While user presses the "query" button, the query results are displayed in the TextArea. For example : if the instances of "Pear-tree" class that show yellow circle spots on the leaves would be queried , user can select "At-part", "leaf" , "Has-ds",

“spotting” , “Has-c”, “yellow”, “Has-s”, “circle” and press “query” button at last (see Fig.7).Details of the method are as follows:

Firstly, obtaining properties and corresponding values to generate a string used by SQWRL queries. For example: with regard to the above query, the string generated is as follows : Pear-tree(?x) ∧At-part(?x, leaf) ∧ Has-ds(?x, spotting) ∧ Has-c(?x, yellow) ∧ Has-shape(?x, circle).

Secondly, creating an instance of SQWRL query engine for owlModel1.

Finally, running SQWRL queries. Calling the `runSQWRLQuery()that` SQWRL query engine provides to run SQWRL queries.

3) Diagnosis Query

Interactive diagnosis query supports adding instances of “Pear-tree” class into the PDDO and diagnosis based on reasoning. User can input the name of a instance in the “name” JTextField, select properties from properties of “Pear-tree” class through the “property” JComboBox, and then select values of properties through “value” JComboBox, after that, pressing “add” button can realize adding a new instance, it’s properties, and values, furthermore, pressing “diagnosis” button can realize diagnosing the disease that the certain instance gets by reasoning based on SWRL(see Fig.8). Details of the method are as follows:

Firstly, adding a new instance, properties and values. GetOWLNamedClass() is called by the defined owlModel1 to obtain the OWLClass “Pear-tree”, and then createOWLIndividual() is called by this OWLClass to create a new instance in owlModel1, in the end setPropertyValue (OWLProperty, OWLIndividual) is called by the certain instance to add it’s properties and values.

Secondly, reasoning based on SWRL. Getting all SWRL rules (SWRL rules library) using the SWRLFactory, and then creating an instance of SWRLRuleEngine for owlModel1 by SWRLRuleEngineFactory.create(), finally using the infer( ) that SWRLRuleEngine interface provides to load rules and knowledge from owlModel1 into a rule engine, run the rule engine, and write any inferred knowledge back to owlModel1.

Thirdly, inferred knowledge are written to the owl file. Fig.9 shows the modified file, the blue block represents the instance, it’s properties and values that are added in Fig.8, “<Has-disease rdf:resource="#Pear_Rust"/>” of blue block is a new relationship obtained by reason. The algorithm is as follows:

```
FileOutputStream outFile1=new FileOutputStream(uri);//
uri file path
Writer out=new OutputStreamWriter(outFile1,"UTF-8");
OWLModelWriter omw=new
OWLModelWriter(owlModel1,owlModel1.getTripleStoreModel().
getActiveTripleStore(),out);omw.write();out.close();
FileInputStream file1 = null;file1 = new
FileInputStream(uri1);
```

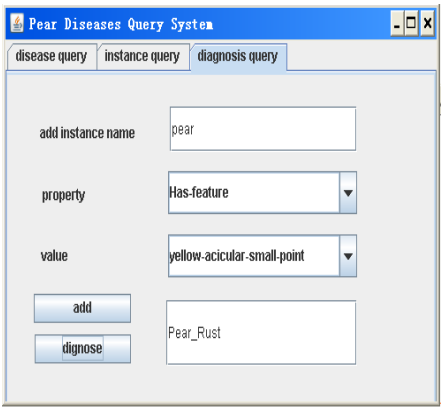

Fig. 8. Disease query interface

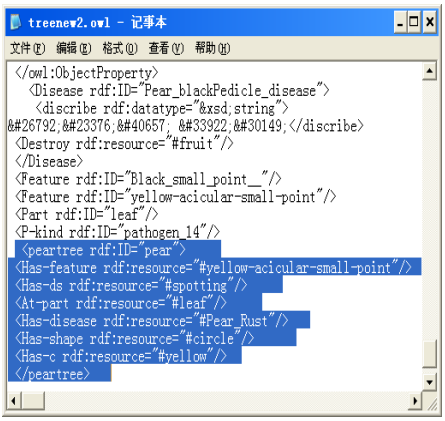

Fig. 9. Modified owl file

5 Conclusion

In this paper, the modeling of PDDO based on SWRL is studied, design and realization techniques of the query system based on PDDO and SWRL are proposed. This query system based on reasoning realizes disease query, instance query, and diagnosis query by using Protégé-Owl API, furthermore, it supports interactive query. Through it pear diseases knowledge can be obtained from PDDO according to user needs, new and inferred knowledge are written to the PDDO owl file. Query results show that the query system based on PDDO and SWRL is practical for users to query information from PDDO.

Acknowledgment. I would like to express my gratitude to all those who have helped me during the writing of this thesis. I acknowledge the help of Professor Liang Yong. I do appreciate his professional instructions. Last but not the least, my gratitude also extends to my family who have been assisting, supporting and caring for me all of my life.

References

1. Li Hua, Q.: Ontology Storage and Querying Technology: Master's Thesis, Beijing University of Posts and Telecommunications (2007)
2. Harmelen, F., Hendler, J., Horrocks, I., et al.: OWL Web Ontology Language Reference. World Wide Web Consortium (February 10, 2004), http://www.w3.org/tr/owl-ref
3. http://www.w3.org/RDF/
4. http://protege.stanford.edu/
5. http://protege.stanford.edu/plugins/owl/api/
6. http://www.w3.org/Submission/SWRL/#1

7. Golbreich, C., Imai, A.: Combining SWRL rules and OWL ontologies with Protégé OWL Plugin, Jess, and Racer. In: 7th International Protégé Conference, Bethesda, MD (2004)
8. http://protege.cim3.net/cgi-bin/wiki.pl?SWRLEditorFAQ
9. http://www.daml.org/2003/11/swrl/abstract.html#2.1
10. : Knowledge Representation and Semantic Reasoning of Mandarin Fish Disease Diagnosis Based on Ontology and SWRL. Journal of Library and Information Sciences in Agriculture 21(06) (June 2009)
11. Excavating ImplicitRelation Based on SWRL. Information Analysis and Research (2011)