CONSTRAINT SATISFACTION PROBLEM SOLVING BASED ON OWL REASONING

Haoyi Xiong
College of Electrical and Electronic Engineering, Huazhong University of Science and Technology, 430074, Wuhan, P.R China

Ying Jiang
School of Information Management, Wuhan University, Wuhan 430072, P.R. China

ABSTRACT

With the rapid development and wide application of the Constraint Satisfaction Problem (CSP) theory, the constraints are manifold diversities. In order to support a general solving mechanism, we borrow the OWL reasoning technology in semantic web, and turn the traditional CSP solving into CSP solving based on OWL reasoning (Ocr). According to Ocr, we present the Domain and Constrains of CSP by using OWL classes, and describe variables' states with OWL individuals. We apply OWL knowledge conflicts detection, which can detect the conflicts between OWL individual and OWL class, to evaluate whether the variables' state is satisfied all constrains. This paper first briefly introduces the basic description of Ocr, and Enumerating-Reasoning algorithm for solution space searching. We provide a demo of solving a practical four color problem with Ocr. Finally, in this paper, we pay an outlook to the future works about the more effective and intelligent Ocr.

KEYWORDS

CSP, RDF/OWL, OWL Knowledge Conflicts Detection.

1. INTRODUCTION

With the rapid development and wide application of the CSP theory [2], such as scheduling, planning, resource allocation, the constraints are manifold diversities. It is hard to build a CSP modeling in a general way. In order to solve that, we provide Ocr as a general methodology and Enumerating-Reasoning algorithms for solution searching.

A CSP is defined as <**X**, **D**, **C**>, which includes variables, domain and constraints [2]. In this article, those elements of CSP are considered as three kinds of knowledge [1]. We apply OWL individuals [10] to describe the variables' state and OWL class [10] to describe constraints and domain. Then we determine whether variables are valid in domain and satisfied all constraints by using OWL knowledge conflicts detection [4]. For example we can map a simple CSP—"select integers from 3 to 8 fill the blanket of □ ≠ □" into an OWL based CSP description, as OWL classes of domain: (owl:class num, owl:onProperty, (rdf:resource hasInteger, owl:allValuseFrom, (owl:class range, owl:one of, {3, 4, 5, 6, 7, 8}))), and OWL classes of constraints: (owl:class a, owl:isSubClassOf, owl:class num), (owl:class b, owl:interSectionOf, {owl:class num, (owl:class, owl:ComplemtOf, owl:class a)}). We enumerate each possible variables in domain such as {(en1, rdf:type, rdf:resource a), (en2, rdf:type, rdf:resource b), (en1, hasInteger, 1), (en2, hasIntger, 2)}, and finally we can get the set of solutions. In this way we solve the CSP description on OWL and the basic solution searching in solution space.

Actually, we apply Jena [15] and Protégé [14] combination with RacerPro [16] to implement the platform for Ocr. Jena is a Java framework for building Semantic Web applications. It provides a programmatic environment for RDF, RDFS and OWL, SPARQL and includes a rule-based inference engine. And Protégé is a free, open source ontology editor and knowledge-base framework. RacerPro is an OWL Reasoner and inference server for the Semantic Web. In our research, Jena API is used to program the reasoner for OWL

Knowledge Conflicts Detection; we apply Jena as the Reasoning machine for OWL Knowledge Conflicts Detection, and Protégé as the editor of OWL.

The remainder of this paper is organized as follows. Sect. 2 presents a brief description of Ocr; Sect.3 illustrates Constraint-Evaluation with Enumerating-Reasoning algorithm; to present a demo of Ocr, we solve a practical four color problem with Ocr in Sect. 4. Finally we make a conclusion on Ocr and outlook towards future work in Sect. 5.

2. A BRIEF DESCRIPTION OF OCR

This section presents a brief description of Ocr. We mainly discuss the definition of OWL based Variables' State, OWL based Constraints, OWL based Variables' Domain, and the relationship between Ocr and CSP. In OWL, the conceptions of knowledge base and ontology base are interchangeable. So the knowledge base we will mention in following could be considered as ontology base as well.

Definition 1 (Conflicts Detection Function). *We define **R** to be Conflicts Detection Function, and **R** is defined as: Let P to be a set of OWL individuals and B to be a set of OWL classes' set. Then **R**: P×B→ {true, false}. In this way, we use **R** to determine whether OWL individuals are satisfied their OWL classes by using OWL knowledge conflicts detection. Actually, we choose Jena as a reasoning machine to construct **R**.*

Definition 2 (OWL based Variables' State). *We define **Vr** to be OWL based Variables' State, which indeed is a piece of knowledge for describing the states of variables in CSP. In practice, we use OWL individuals to present **Vr**.*

Definition 3 (OWL based Variables' Domain). *We define **Dr** as OWL based Variables' Domain which is a set of OWL classes to present the domain of CSP. And the properties of **Vr** must be the instance of OWL classes in **Dr**. What is more, it is inferred that: if R (**Vr**, **Dr**) = **true**, then **Vr** must be valid in **Dr**.*

Definition 4 (OWL based Constraints). *We define **Cr** as OWL based Constraints which is a set of OWL classes. In the view of knowledge management, **Cr** is a knowledge base. It is inferred that: if **R (Vr, Cr) = true**, then **Vr** must be satisfied **Cr**.*

Definition 5 (Ocr, and Original CSP). *CSP on OWL Reasoning (Ocr) is defined as triple <**R**, **Dr**, **Cr**>, where **R**, **Dr**, and **Cr** are defined above. We apply Ocr to be an OWL based semantic presentation of CSP. Oppositely, we defined Original CSP as: Let c is a Ocr and p is a CSP; if c is the Ocr of p, then p is the Original CSP of c.*

Definition 6 (Ocr Solution). *Ocr Solution is defined as a special kind of **Vr** in Ocr as: Let s to be one Ocr, s = <**R**, **Dr**, **Cr** >; if **R (Vr, Dr)** × **R (Vr, Cr) =true**, then **Vr** is one Ocr Solution of s. We use Ocr Solution to present the variables satisfy all the constraints of original CSP.*

3. ENUMERATING-REASONING ALGORITHM

Enumerating-Reasoning algorithm (***era***) aims to search the constrains-satisfied solutions in solution space. Towards a Ocr solution, solution space refers to ***Dr*** of the Ocr. It is impossible for us to enumerate each ***Vr*** in ***Dr*** by using traditional enumerating methods; in order to solve that, we support the Comparing Enumerating Function as follow:

Definition 7 (Comparing Enumerating Function). *We define **Fe** as Comparing Enumerating Function to enumerate each **Vr** in **Dr**. **Fe** is defined as: Let **d** is a set of **Dr**, p and **q** are two sets whose elements are sets of **Vr**, then **Fe**: **d** × **p** → **q**; for **Si** □ **q**, **Rs (Si, d) = true** AND **Si** is NOT the member of **p.***

In practice, the implementation of ***Fe*** depends on the OWL reasoning machine that we choose and the actual CSP which we meet. According to our research, how to construct a ***Fe*** is still a puzzle, and an awkward ***Fe*** must decline the performance of ***era***. Towards known Ocr and ***Fe***, ***era*** could be described in Pseudocode as below:

```
PROCEDURE era (R, Fe, Dr, Cr)
    Vr ← empty
    Set result_set, enum_set
    InitialSet (result_set)          //InitialSet (set) initialize a set
    InitialSet (enum_set)
    WHILE Dr! = enum_set
      Vr ← Fe (Dr, enum_set)
      IF R (Vr, Cr) ==true
        Append (result_set, Vr)   //Append (set, element) append a new elements into a set
    Append (enum_set, Vr)
    END
    RETURN result_set
END PROCEDURE
```

4. SOLVING FOUR COLOR PROBLEM WITH OCR

Towards the practical four color problem shown in Figure 1., the problem can presents as a traditional CSP < ***X, D, C*** >: ***X*** = {*R1, R2, R3, R4, R5*}, *Ri* in Reg stand for the color of region *i*. ***D*** = {*D1, D2, D3, D4, D5*}, *Di* ≡ {*Red, Yellow, Green, Blue*}. ***C***= {“R1 is different from R2, R4, and R5”, “R2 is different from R3 and R5”, “R3 is different from R4 and R5”, “R4 is different from R5”}.

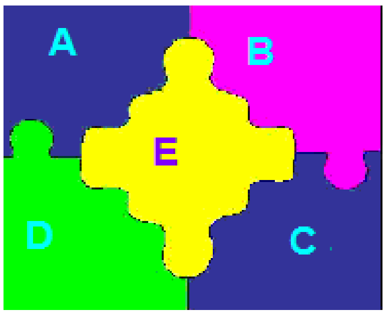

Figure 1. A Practical Four Color Problem

We turn this problem into a Ocr. And we define OWL class ColorableBoard to describe generic characters of colored board; OWL classes ColorableBoardA, ColorableBoardB, ColorableBoardC, ColorableBoardD and ColorableBoardE is defined describe each colored board in this problem; OWL classes ColorA ColorB, ColorE, ColorD and ColorE are defined as constraints for painting. In this problem ***Dr*** = { ColorableBoardA, ColorableBoardB, ColorableBoardC, ColorableBoardD, ColorableBoardE }, and ***Cr*** = {ColorA, ColorB, ColorC, ColorD, ColorE}. Regarding to the length of paper, we just show part of OWL codes.

```
<owl:Class rdf:about="#ColorableBoard">
     <rdfs:subClassOf>
       <owl:Restriction>
         <owl:onProperty>
           <owl:ObjectProperty rdf:about="#hasColor"/>
         </owl:onProperty>
         <owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
         >1</owl:cardinality>
       </owl:Restriction>
     </rdfs:subClassOf>
     <rdfs:subClassOf rdf:resource="http://www.w3.org/2002/07/owl#Thing"/>
</owl:Class>

<owl:ObjectProperty rdf:about="#hasColor">
   <rdfs:range rdf:resource="#FouColor"/>

    <rdfs:domain rdf:resource="#ColorableBoard"/>
</owl:ObjectProperty>
…
```

```
<FouColor rdf:ID="A">
        <owl:sameAs rdf:resource="#Green"/>
        <owl:differentFrom rdf:resource="#E"/>
        <owl:differentFrom rdf:resource="#D"/>
        <owl:differentFrom rdf:resource="#B"/>
</FouColor>
<FouColor rdf:ID="B">
        <owl:differentFrom rdf:resource="#E"/>
        <owl:differentFrom rdf:resource="#C"/>
        <owl:sameAs rdf:resource="#Blue"/>
</FouColor>
…
<owl:Class rdf:about="#ColorA">
        <rdfs:subClassOf rdf:resource="#FouColor"/>
        <owl:disjointWith>
          <owl:Class rdf:about="#ColorE"/>
        </owl:disjointWith>
        <owl:disjointWith rdf:resource="#ColorD"/>
        <owl:disjointWith>
          <owl:Class rdf:about="#ColorB"/>
        </owl:disjointWith>
</owl:Class>
…
…
<owl:Class rdf:ID="ColorableBoardA">
       <rdfs:subClassOf>
         <owl:Restriction>
           <owl:allValuesFrom>
             <owl:Class rdf:ID="ColorA"/>
           </owl:allValuesFrom>
           <owl:onProperty>
             <owl:ObjectProperty rdf:ID="hasColor"/>
           </owl:onProperty>
         </owl:Restriction>
        </rdfs:subClassOf>
       <rdfs:subClassOf>
         <owl:Class rdf:ID="ColorableBoard"/>
        </rdfs:subClassOf>
</owl:Class>
…
```

As the OWL shown above, we use OWL operators owl:disjointWith and owl:diffrentFrom to present the complex logics in constraints. We build the ***Fe*** on principle of permutation and combination. Then we enumerate each ***Vr***, such as:

```
…
<ColorableBoardA rdf:ID="enum1">
     <hasColor>
       <ColorB rdf:ID="Green">
         <rdf:type rdf:resource="#ColorA"/>
        </ColorB>
     </hasColor>
</ColorableBoardA>
<ColorableBoardB rdf:ID="enum2">
     <hasColor>
       <ColorB rdf:ID="Blue">
```

```
        <rdf:type rdf:resource="#ColorB"/>
       </ColorB>
     </hasColor>
</ColorableBoardB>
<ColorableBoardC rdf:ID="enum3">
     <hasColor>
       <ColorC rdf:ID="Green">
        <rdf:type rdf:resource="#ColorC"/>
       </ColorC>
     </hasColor>
</ColorableBoardC>
<ColorableBoardD rdf:ID="enum4">
     <hasColor>
       <ColorD rdf:ID="Red">
        <rdf:type rdf:resource="#ColorD"/>
       </ColorD>
     </hasColor>
</ColorableBoardD>
<ColorableBoardE rdf:ID="enum5">
     <hasColor>
       <ColorE rdf:ID="Yellow">
        <rdf:type rdf:resource="#ColorE"/>
       </ColorE>
     </hasColor>
</ColorableBoardA>
     …
```

Finally we program the conflicts detection function ***R*** for this problem by using Java class ValidityReport in Package com.hp.hpl.jena.reasoner. Partial codes of tester are shown below:

```
     ValidityReport validity = ontModel.validate();   // ontModel means ontology elements in Vr, Cr, Dr
     if(validity.isValid()){
this.fail("There should be no conflict");
}else{
               …        // append this Vr into solution set.
for (Object report:validity.getReports())
       System.out.println(" - " + report);
}
```

With this program, we can simply evaluate the solution of this four color problem. If the ***Vr*** satisfied ***Cr***, Vr would append into the set of Ocr Solutions. Other wise, application will report the track of conflicts as below:

```
       - Error (conflict): Description of Conflict…, may be
          The reason of Conflict…
        ... + number of other similar report
```

We could analysis the Ocr processing by those tracks. So far, the performance of calculating on this problem by Ocr is much lower than the one by traditional CSP solving. However, as the demo above show us, Ocr is a universal method of solving towards the CSPs which could be described by OWL.

5. CONCLUSION AND FUTURE WORK

In conclusion, Ocr we mentioned above support us a general platform and methodology for CSP solving. But there are still some limitations:

- At the most of time, CSPs focus on the application in specific domains; however OWL class is difficult to built up and maintain for most domain experts of non-computer-science majors. Natural language and mathematics expressions are both widely used in any filed of science and engineering; however natural language lacks the ability of complex logistics description, nowadays, we attempt to extract OWL classes from some of mathematics expression and convert them into a Ocr.

- The performance of Enumerating-Reasoning algorithm needs improvement for large scaled enumerating and conflicts detecting, especially the construction of comparing enumerating function.

The limitations listed here are also the focuses of the future work for the author.

ACKNOWLEDGEMENT

This work belongs to open source software project—LighTECHs which focuses on semantic web oriented applications.

REFERENCES

Book

Alexander Smirnov, Mikhail Pashkin, Nikolai Chilov, Tatiana Levashova and Andrew Krizhanovsky. *Ontology-Driven Knowledge Logistics Approach as Constraint Satisfaction Problem, LNCS Volume 2888/2003.* Springer Berlin / Heidelberg

Apt, Krzysztof (2003). *Principles of constraint programming.* Cambridge University Press. ISBN 0-521-82583-0

Davies, John (2006-07-11). *Semantic Web Technologies: Trends and Research in Ontology-based Systems.* Wiley, ISBN 0470025964

Passin, Thomas B. (2004-03-01). *Explorer's Guide to the Semantic Web.* Manning Publications, ISBN 1932394206

Wilson, Robin, 2002. *Four Colours Suffice.* Penguin Books Ltd, London UK

Changqing Li and Tok Wang Ling. *OWL-Based Semantic Conflicts Detection and Resolution for Data Interoperability LNCS Volume 3289/2004.* Springer Berlin / Heidelberg

Journal

Tim Berners-Lee, James Hendler and Ora Lassila. The Semantic Web--A new form of Web content that is meaningful to computers will unleash a revolution of new possibilities. *Scientific American*, 2005-5-17, pp 1-18

Eric Miller. An Introduction to the Resource Description Framework. *D-Lib Magazine*, 1998-5, pp 2-32

Alavi, M. and Leidner, D. (2001). Review: Knowledge Management and Knowledge Management Systems: Conceptual Foundations and Research Issues. *MIS Quarterly*, 25, 1, pp 107-136

Conference paper or contributed volume

DL McGuinness, F van Harmelen. OWL Web Ontology Language Overview. *World Wide Web Consortium*, 2004

J.H.M. Lee, H.F. Leung and H.W. Won. Towards a More Efficient Stochastic Constraint Solver. *Second International Conference on Principles and Practice of Constraint Programming*, Cambridge, Massachusetts, USA, August, 1996

Allaire, F. *Another proof of the four colour theorem—Part I*, *7th Manitoba Conference on Numerical Mathematics and Computing*, Congr. Numer. 1977

J.H.M. Lee and T.W. Lee, A WAM-Based Abstract Machine for Interval Constraint Logic Programming. *Sixth IEEE International Conference on Tools with Artificial Intelligence*, New Orleans, USA, November, 1994

Holger Knublauch. An AI tool for the real world Knowledge modeling with Protégé, *JavaWorld.com*, USA, 2006

Ian Dickinson. Jena Ontology API. *At http://jena.sourceforge.net/ontology/index.htm*l. 2008

Racer Systems GmbH &Co.KG. RacerPro documentation. *At http://www.racersystems.com/products/racerpro/manual. phtml. 2006*