The Knowledge Representation and Reasoning (KR&R) is based on the idea that propositional content can be rigorously represented in formal languages, such that the resulted representations can be productively reasoned over by humans and machines ([1]). Conditional Knowledge Representation and Reasoning represents a new brand of KR&R, for which several formalisms have been developed. In this paper we define XML Language Specifications for such knowledge in order to develop a standardized XML-based formalism for Expert Systems.

**Keywords**—Binary Relation, Conditional Knowledge, Conditional Schema, XML Schema Language.

**I. INTRODUCTION**

The Semantic Web is the abstract representation of data on the World Wide Web, based on the Resource Description Framework (RDF) standards and other standards to be defined. It is being developed by the World Wide Web Consortium (W3C) in collaboration with a large number of researchers and industrial partners.

XML - derived from eXtensible Markup Language - is a general purpose markup language which supports the sharing of structured data across different information systems. As the popularity of XML increases substantially, the importance of XML languages to describe the structure and semantics of human knowledge also increases ([5]). Although there have been proposed about dozen XML languages, no comprehensive mathematical analysis of them have been made.

The database and web community use terms such as schema and XML document (or file). XML Schema is considered to be an abstract definition of the set of conforming XML documents.

XML Schema is the XML schema language recommended by the W3C in order to define constraints which are further used to describe a class of documents ([6]).

The spirit and the development approach behind the Semantic Web require as much as possible formal data/knowledge to be provided in formats that others can read and interpret for unforeseen purposes. In other words:

- Automatically processable meta-data;
- Presented in a standard form;
- Allow flexible and dynamic interpretation for unforeseen purposes.

We believe that providing a framework in abstract terms is important to understand the various aspects of a XML language description in order to facilitate its efficient implementation. Towards this goal, in the present paper we propose a XML Schema formalism for representing and processing conditional knowledge texts. We also believe that is important to have a mathematical framework to study when an efficient operation is possible and when it is not.

**A. Our purpose**

Let us consider a collection of knowledge pieces, more precisely of conditional knowledge pieces (texts in natural languages that contains if-then sentences), for which a conditional schema formalism has been developed ([2], [8] and [9]). One may want to find the union of two (or more) such schemas. Then, it could be interesting in finding out if a specific knowledge piece (KP) that is valid with respect to both schemas remains valid in the resulted union schema, and conversely, if a KP that is invalid with respect to both schemas remains invalid in the union schema.

Another problem could appear if a schema representation would evolve in a new version during an update process. In this case, one can be interested in constructing the intersection of the old and new schema representation to determine, for example, which knowledge remains unchanged (the closure properties of conditional schemas under boolean operations: union, intersection and difference).

In XML query processing, computing answers from multiple documents (in our case, knowledge pieces) may require to compute union of the internal representation systems – that is, union of the conditional schemas. Such issues are directly related with the efficient implementation of an XML language and following this idea, we consider that the formalism we propose here meet the needs of this formalism.

**B. Conditional schema structure**

The conditional knowledge representation and processing mechanism was developed under the name of conditional schema. The formal concept of conditional schema was introduced in [8], being defined by means of a tuple of eight components:

\[ S = (Ob, C_s, E_s, A, V, B_s, h, f) \]

such that:

- \( Ob \) is a set of the object names. This set is divided into two subsets corresponding to the individual and abstract objects as follows:
The conditional symbols can be of two types:

- one type points out to the existence of a relation between the object to which the conditional symbols is applied and another one (noted here with object2)

```
<condition>
  <type> relation </type>
  <object2> object2 id </object2>
  <relation> relation id </relation>
</condition>
```

- the second kind request a particular value attribute for the object the conditional symbol is applied or another object of the conditional schema

```
<condition>
  <type> attr/value </type>
  <attribute> attribute name </attribute>
  <value> attribute value </value>
</condition>
```

As XML Conditional Schema representations are organized in a nested data structure, each element having several attributes. The top-most elements are the objects and the relations that exist between them. The rules attached to the
A. XML-conditional knowledge base

As an exemplification for the formalisms already introduced, in this section we give an example of a XML-conditional knowledge base using the XML Conditional Schema specifications.

A XML-conditional knowledge base is an XML file which contains the instances for the XML Conditional Schema entities and their corresponding values. As every XML file, such knowledge base begins with a prolog which contains a declaration that identifies the document as a XML document:

```xml
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
```

Java implementation. To process the xml files we have worked with DOM and SAX.

Simple API for XML Parsing (SAX) offers an interface based on events in order to work with XML files. The XML files are processed in order to find out the XML elements treated by the application. Depending on the tag encountered in the XML we can decide what operation to perform.

Document Object Model (DOM) offers a set of predefined interfaces by W3C DOM Working Group. The interfaces facilitate the tree representation of XML in applications. The specifications for these interfaces are defined trough the Interface Definition Language (IDL) that is independent of the programming language. In DOM each node contains one of the components from an XML structure. The two most common types of nodes are element nodes and text nodes. Using DOM functions we can create nodes, remove nodes, change their contents, and traverse the node hierarchy.

The packages used to process the XML files corresponding to the proposed Conditional Knowledge Bases representation are:

- **javax.xml.parsers** – it contains the classes for bought SAX and DOM;
- **org.xml.sax** – it contains the interfaces for SAX processing;
- **org.xml.sax.helpers** – contains the classes that aid with using SAX;
- **org.xml.sax.ext** – offers two interfaces to extend SAX;
- **org.w3c.org** – offers interfaces that can be used to represent the XML in a tree like structure.

**Example of XML Conditional Knowledge Base.** We provide the XML file also with the natural language text of the Conditional Knowledge Piece. An example of such a XML-conditional knowledge base is shown below:

```xml
<?xml version="1.0" encoding="UTF-8" standalone="no"?>

<knowledge_base>
  <natural_language_text>
    All birds are animals and all penguins are birds. Birds have wings and two legs. Generally, birds fly but penguins do not fly. Every animal is a flying animal if it has wings and flies. Every flying animal is a mobile animal. Also every animal which has legs is considered mobile animal. Bob is a penguin and Tweety is a small bird. Bob is friend with Tweety.
  </natural_language_text>

  <node>
    <type> individual </type>
    <object_id> 11 </object_id>
    <object> Bob </object>
    <parameters>
      <parameter>
        <type> attr/value </type>
        <name> fly </name>
        <value> no </value>
      </parameter>
      <parameter>
        <type> attr/value </type>
        <name> wings </name>
        <value> yes </value>
      </parameter>
      <parameter>
        <type> attr/value </type>
        <name> legs </name>
        <value> yes </value>
      </parameter>
    </parameters>
  </node>

  <node>
    <type> individual </type>
    <object_id> 12 </object_id>
    <object> Tweety </object>
    <parameters>
      <parameter>
        <type> attr/value </type>
        <name> fly </name>
        <value> yes </value>
      </parameter>
      <parameter>
        <type> attr/value </type>
        <name> wings </name>
        <value> yes </value>
      </parameter>
      <parameter>
        <type> attr/value </type>
        <name> legs </name>
        <value> yes </value>
      </parameter>
      <parameter>
        <type> attr/value </type>
        <name> size </name>
        <value> small </value>
      </parameter>
    </parameters>
  </node>

</knowledge_base>
```
B. XML query language

The main role of an XML query language is to allow the formulation of queries and determine the result set of the XML elements that should be returned ([10]). We assume that a query input is a set of known nodes within multiple XML Conditional Knowledge Bases. Following this idea, in order to execute an XML query, the query engine should be supplied with

1. the query string that describes the objects for which the relations existing between them have to be identified

   <object1> name object </object1>
   <object2> name object </object2>

   For this version, we consider that object1 is always an individual object and for this reason all the inputs address queries expressed for particular cases.

2. optionally, the URL of the Conditional Knowledge Base on which the query must perform; if it is not included in the query, then all the Conditional Knowledge Bases will be considered.

C. Processing XML Conditional Schema data

The XML file corresponding to a certain Conditional Knowledge Base is processed with respect to the user interrogation in order to determine the corresponding answer.
from knowledge it represents. For this task, a path-driven reasoning is performed using an inference mechanism developed in [2] to perform reasoning based on conditional schema representations. Because conditional relations are used, determining the relationships between pairs of objects usually implies to verify the conditions identified by the conditional symbols. Thus, if we have an attr/value type condition, then the value(s) indicated must be fulfilled by the corresponding pairs (attr, value) of the individual object of the interrogation. The other case corresponds to the relation type condition in which a specific relation must exist between the individual object of the interrogation and another object (individual or abstract) indicated by the tag.

We have that for each input an XML file is generated, this file containing the conditional schema representation of the Conditional Knowledge Piece text together with the new relations deduced by the inference engine in order to construct the answer to the user interrogation.

The conditions attached to the conditional schema relations can be used also for providing explanations regarding the generated answers. Indeed, if we consider the following interrogations for the system:

**Query_1:**

```xml
<object1> Bob </object1>
<object2> mobile animal </object2>
```

**Query_2:**

```xml
<object1> Tweety </object1>
<object2> mobile animal </object2>
```

the system will attach to the deduced relations, the conditional symbols what were fulfilled during the inference:

```xml
<relation>
  <from> i1 </from>
  <to> a5 </to>
  <relation_id> r9 </relation_id>
  <label> is_a_kind_of </label>
  <cond_symb> q2 </cond_symb>
</relation>
```

for the Query_1 and, respectively,

```xml
<relation>
  <from> i2 </from>
  <to> a5 </to>
  <relation_id> r10 </relation_id>
  <label> is_a </label>
  <cond_symb> q1 </cond_symb>
</relation>
```

for Query_2. The natural language explanations for the conditional symbols q1 and q2 corresponding to the considered interrogations, that is, to the individual objects, can be formulated as follows:

- for Query_1: Because Bob has legs.
- for Query_2: Because Tweety flies and has wings.

The algorithm for computing the answer is a Breadth-First routine, which searches for a path between the two objects specified in the interrogation. It uses a FIFO queue (noted here with q) in order to store the temporal paths in the considered conditional schema.

```plaintext
Procedure Query(Schema, obj1, obj2)
q.push({obj1})
while (!q.isEmpty())
  TmpPath ← q.pop()
  obj ← TmpPath[len(TmpPath)-1]
  If (obj-obj2) then
    composeRelation(TmpPath)
    return;
  EndIf
  For each (link_obj in Schema\TmpPath)
    If (rel-relation(link_obj, obj) and (rel.cond_sym = null or evalCond(obj1, rel) = true)) then
      NewPath ← TmpPath + [link_obj]
      q.push(NewPath);
  EndIf
EndWhile
EndProcedure
```

The routine would be called like this:

```plaintext
NameRel ← Query(Schema, obj1, obj2);
```

where Schema is the set of nodes (individual and abstract objects) corresponding to a particular conditional schema.

III. CONCLUSION

This paper proposes an XML Schema based-formalism for conditional knowledge representation and processing. The present article follows our work dedicated to this type of knowledge by mapping XML representations to conditional schema components and implementing the inference process developed for this schema with respect to the XML specifications. We consider that our proposal covers all kinds of the conditional schema matching problems.

ACKNOWLEDGMENT

The author Mihaela Colhon has been founded for this research by the strategic grant POSDRU/89/1.5/S/61968, Project ID 61986 (2009), co-financed by the European Social Fund within the Sectorial Operational Program Human Resources Development 2007-2013.

The author Cristina Zamfir has produced this article under the project “Supporting young Ph.D students with frequency by providing doctoral fellowships”, co-financed from the EUROPEAN SOCIAL FUND through the Sectorial Operational Program Development of Human Resources.

REFERENCES

Nicolae Țăndâreanu received the degree in mathematics and computer science in 1970 and the Ph.D. in mathematics in 1975, from the University of Bucharest, Romania. He is currently Professor of computer science at the Faculty of Mathematics and Computer Science, University of Craiova, Romania.

N. Țăndâreanu has published over 70 papers in refereed journals, lectures and monographs in the following domains: automata theory, formal languages, algorithms, numerical computation, generalized Boolean functions and logic programming, knowledge bases, knowledge representation and expert systems. He is quoted in the following encyclopedic works: International Book of Honor, Fourth World Edition (p. 223), Dictionary of International Biography, Twenty Fourth Edition (p. 328).