Int. J. Advanced Networking and Applications
Volume: 02, Issue: 04, Pages: 723-731 (2011)

XML Solving Problem of Expert System

Yasser A. Nada
Department of computer science,
Faculty of Computers and Information Systems,
Taif University -Kingdom of Saudi Arabia,
Email: y_nada@yahoo.com

----------ABSTRACT----------

The Extensible Markup Language (XML) is a subset of SGML that is completely described in this paper. Its goal is to enable generic SGML to be served, received, and processed on the Web in the way that is now possible with HTML. XML has been designed for ease of implementation and for interoperability with both SGML and HTML.

An expert system is a computer program designed to simulate the problem-solving behavior of a human who is an expert in a narrow domain or discipline. Expert Systems (ES), also called Knowledge Based System (KBS), are computer application programs that take the knowledge of one or more human experts in a field and computerize it so that it is readily available for use.

The main objective of this paper was to investigate the usage of different refinement methodologies for different layers of knowledge base modeling and investigate the possibility of building an expert system development and refinement tool. In our work we used XML as a knowledge representation to represent the knowledge base. Therefore we used the mathematical model to refinement of a knowledge base.

Key Words: Expert System, Refinement Knowledge Base, XML

Date of Submission: October 10, 2010 Date of Acceptance: November 11, 2010

1. Introduction

Today, XML is invading the world of computers and occupying most of its fields. It is widely spreading over the internet, networks, information systems, software and operating systems, DBMS, search tools, web development and services, communication protocols and other fields. As a result, XML data are floating within and between different applications and systems all over the internet and intranets. Due to the huge amount of XML structured data being circulated, controlling XML data becomes imperative for various purposes and aims [1].

XML (Extensible Markup Language) has emerged as the most important format for data exchange and storage over systems across the greatest variety of tools and platforms. It is a rapidly maturing technology with powerful real-word applications, particularly for the management, display and organization of data. Together with its associated tools (XSL, XSLT, XPath, XLink, XPointer, DOM DTD, Schemas, …etc), it is an essential technology for anyone looking for more efficient and cost effective ways of both managing and transferring data.

Perhaps the most well known applications are web related, but there are many other non-web based applications where XML is useful–for example as a replacement for (or to complement) traditional database [2]. A lot of languages that are based on XML are around, e.g., RDF [3], OIL [4], DAML+OIL, OWL, etc. Further, XML tools have been used to develop a library of inference methods to be used for reasoning on different representations.

Expert System (ES), also called a Knowledge Based System (KBS), is computer application programs that take the knowledge of one or more human experts in a field and computerize it so that it is readily available for use. A central problem of expert system is knowledge base refinement [5]. The knowledge base refinement is concerned with improving an incorrect, inconsistent, and incomplete domain theory, and thus a uniform refinement framework can be designed to support knowledge base validation and maintenance [6]. The knowledge base optimization aims at improving the performance of the knowledge-based system by reducing the response time of knowledge based expert system and minimizing the number of generated questions. The optimized system is guaranteed to compute correct results independent of the scheduling strategy and execution environment [7].

The outline of the paper is as follows. Section 2 gives a brief introduction to XML. Section3 presents the XML structure. Section 4 components of an expert system. Section 5 shows the importance of knowledge. Section 6 explains knowledge in the expert systems. Section 7 explains knowledge base refinement. Section 8 Knowledge Representation Methodology based XML how the proposed system works and complete example. Finally, section9 summarizes this paper.

2. What is XML?

Extensible Markup Language, abbreviated XML, describes a class of data objects called XML documents and partially describes the behavior of computer programs which process them. XML is an application profile or restricted form of SGML, the Standard

Generalized Markup Language. By construction, XML documents are conforming SGML documents.

XML documents are made up of storage units called entities, which contain either parsed or unparsed data. Parsed data is made up of characters, some of which form character data, and some of which form markup. Markup encodes a description of the document's storage layout and logical structure. XML provides a mechanism to impose constraints on the storage layout and logical structure.

A software module called an XML processor is used to read XML documents and provide access to their content and structure. It is assumed that an XML processor is doing its work on behalf of another module, called the application. This specification describes the required behavior of an XML processor in terms of how it must read XML data and the information it must provide to the application.

2.1 The main difference between XML and HTML

XML is not a replacement for HTML. XML and HTML were designed with different goals:

XML was designed to describe data and to focus on what data is.
HTML was designed to display data and to focus on how data looks.
HTML is about displaying information, XML is about describing information.

XML specifies neither semantics nor a tag set. In fact XML is really a meta-language for describing markup languages. In other words, XML provides a facility to define tags and the structural relationships between them. Since there's no predefined tag set, there can't be any preconceived semantics. All of the semantics of an XML document will either be defined by the applications that process them or by stylesheets.

The HyperText Markup Language is the Web language of choice, although it is problematic and limiting. XML solves many of the problems Web authors have experienced with HTML and is responsible for XHTML, a recasted HTML, in XML. Web authors and other publishers will be using XML for many years because it offers them an effective and powerful multi-media publishing solution.

2.2 XML is a complement to HTML

It is important to understand that XML is not a replacement for HTML. In the future development of the Web it is most likely that XML will be used to structure and describe the Web data, while HTML will be used to format and display the same data.

2.3 XML in future Web development

We have been participating in XML development since its creation. It has been amazing to see how quickly the XML standard has been developed, and how quickly a large number of software vendors have adopted the standard.

We strongly believe that XML will be as important to the future of the Web as HTML has been to the foundation of the Web. XML is the future for all data transmission and data manipulation over the Web.

3. XML Structure

Each XML document has both a logical and a physical structure. Physically, the document is composed of units called entities. An entity may refer to other entities to cause their inclusion in the document. A document begins in a "root" or document entity. Logically, the document is composed of declarations, elements, comments, character references, and processing instructions, all of which are indicated in the document by explicit markup.

A software module called an XML processor is used to read XML documents and provide access to their content and structure. It is assumed that an XML processor is doing its work on behalf of another module, called the application. This specification describes the required behavior of an XML processor in terms of how it must read XML data and the information it must provide to the application.

3.1 Documents

XML documents are similar to HTML documents. They contain information and markup tags that define the information and are saved as ASCII text. The name of the XML document has an XML extension 'xyz.xml'. A data object is an XML document if it is well-formed. A well-formed XML document may in addition be valid if it meets certain further constraints.

- Well formed XML documents contain text and XML tags which conform with the XML syntax.
- Valid XML documents must be well formed and are additionally error checked against a Document Type Definition (DTD). A DTD is a set of rules outlining which tags are allowed, what values those tags may contain and how the tags relate to each other. Typically a valid document is used when documents require error checking, and enforced structure, or are working within a company/ industry wide environment in which many documents need to follow the same guidelines.

3.2 Well-Formed XML Documents

Well-formed documents are well-formed because they do not have to be created in a structured environment, against a pre-defined set of structural rules, but merely have to comply with XML well-formedness constraints. These constraints require that elements, which are named content containers, properly nest within each other and use other markup syntax correctly. Well-formed XML elements are defined by their use, not by a rigid structural definition, allowing authors to create elements in response to their development. This flexibility offers authors greater control over document processing and design than in traditional SGML environments, in which structure must be formally defined in a DTD before any documents can be written.

Well-formed XML frees Web authors from the fixed nature of HTML, allowing imaginations to prevail over restraint. HTML is a fixed document type, meaning that

Volume: 02, Issue: 04, Pages: 723-731 (2011)

it cannot be expanded or altered to improve its description power.

Well formed XML documents simply markup pages with descriptive tags. You don't need to describe or explain what these tags mean. In other words a well formed XML document does not need a DTD, but is must conform to the XML syntax rules. If all tags in a document are correctly formed and follow XML guidelines, then a document is considered as well formed.

3.2.1 An example XML document

```
<?xml version="1.0"?>
<note>
<to>Tove</to>
<from>Jani</from>
<heading>Reminder</heading>
<body>Don't forget me this weekend!</body>
</note>
```

The first line in the document: The XML declaration should always be included. It defines the XML version of the document. In this case the document conforms to the 1.0 specification of XML.

3.3 Valid XML

The primary difference between valid and well formed XML is their relationship to a document type definition. Well formed XML is designed for use without a DTD, whereas valid XML explicitly requires it. A DTD is a set of rules that a document follows, which software may need to read before processing and displaying a document. These rules generally state the name and contents of each element and in which contexts it can and must exist. Paragraph elements might be defined as containing keyword and code elements and as existing within section and note elements. Valid XML documents may employ certain advanced features of XML, which are not accessible to well formed documents, because of their lack of a DTD. These features can significantly improve the usability of a document, including: linking mechanisms, entities and attributes. Most XML Web sites are likely to be composed of valid XML documents, conforming with customized DTDs, allowing their creators the freedom to structure their sites and use much greater feature sets than HTML has traditionally allowed.

A "Valid" XML document is a "Well Formed" XML document which conforms to the rules of a Document Type Definition (DTD).

The following is the same document as above but with an added reference to a DTD:

```
<?xml version="1.0"?>
<!DOCTYPE note SYSTEM "InternalNote.dtd">
<note>
<to>Tove</to>
<from>Jani</from>
<heading>Reminder</heading>
<body>Don't forget me this weekend!</body>
</note>
```

Valid XML is a more rigid, or formal, form of XML. All XML Documents are well formed documents (otherwise they would not be XML documents). Some XML documents are additionally valid. Valid documents must conform not only to the syntax, but also to the DTD (Document Type Definition). DTD is a set of rules that defines what tags appear in a XML document, so that viewers of an XML document know what all the tags mean. DTDs also describe the structure of a document.

The main difference between valid and well formed is that Valid XML requires a DTD and whereas Well formed XML does not. Still it is advisable for an XML document to have a DTD, because if several people are authoring the document, the DTD will set out the ground rules that they can all work by, and more importantly they can use a parser (the validity checker) to make sure that they are not violating the rules. The XML DTD can either be in the prolog of the document, or it can be in a separate file that is referred to in the prolog. More on DTD in the section on DTD.

3.4 The Parser

An XML parser is a processor that reads an XML document and determines the structure and properties of the data. If the parser goes beyond the XML rules for well-formdness and validates the document against an XML DTD, the parser is said to be a "validating" parser. A generalized XML parser reads XML files and generates a hierarchically structured tree, then hands off data to viewers and other applications for processing. A validating XML parser also checks the XML syntax and reports errors.

3.4.1 Parsing Examples

These parsing examples illustrate common mistakes and proper use of elements. The first two examples are well-formed XML, whereas the following ones would produce parsing errors, as they are not well-formed. Model your elements after the first two examples, obeying well-formed rules.

```
<PRICE>$57.80</PRICE>
<PET><CAT type="Cornish Rex">Cat nests properly
within PET.</CAT></PET>
<WEATHER>Foggy
<LEVEL>Intermediate<LEVEL>
<PASSWORD>planetB612</PASSWD>
<DISTANCE TYPE=KM 120</DISTANCE>
<CAR><engine>engine does not nest properly within
CAR</CAR></engine>
```

3.5 XML Declaration

These XML declarations are commonly used for various types of XML authoring. The first two declarations can be used to describe well-formed and valid XML documents, respectively. The third declaration can be considered the default XML declaration, stating that it is an XML version 1.0 document, that it cannot standalone from external markup declarations and that it is encoded in UTF-8, an 8 bit Unicode character encoding. Use the second XML declaration when creating your own valid XML documents.

```
<?xml version="1.0" standalone="yes"?>
```

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

3.6 General example

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE MEMO [
<!ELEMENT MEMO
(TO,FROM,SUBJECT,BODY,SIGN)>
<!ATTLIST MEMO     importance
(HIGH|MEDIUM|LOW) "LOW">
<!ELEMENT TO       (#PCDATA)>
<!ELEMENT FROM     (#PCDATA)>
<!ELEMENT SUBJECT  (#PCDATA)>
<!ELEMENT BODY     (P+)>
<!ELEMENT P        (#PCDATA)>
<!ELEMENT SIGN     (#PCDATA)>
<!ATTLIST SIGN     signatureFile CDATA #IMPLIED
    email          CDATA #REQUIRED>
]>
<MEMO importance="HIGH">
<TO>Paper Takers</TO>
<FROM>Paper Writer</FROM>
<SUBJECT>Your impressions</SUBJECT>
<BODY>
<P>Now that you are almost done the paper, you must be getting an idea of what XML is about. These emerging technologies sometimes take time though before they catch on.</P>
<P>Did you find the paper helpful? Which areas did you find confusing? How would you improve them?</P>
</BODY>
<SIGN email="y_nada@yahoo.com">Yasser Nada</SIGN>
</MEMO>
```

4. Components of an Expert System

All expert systems are composed of several basic components: a user interface, a database, a knowledge base, and an inference mechanism. Moreover, expert system development usually proceeds through several phases including problem selection, knowledge acquisition, knowledge representation, programming, testing and evaluation. Expert systems have a number of major system components and interface with individuals in various roles. These are illustrated in figure (1). The major components are [8]:

- **Knowledge base** - a declarative representation of the expertise, often in IF THEN rules;
- **Working storage** - the data which is specific to a problem being solved;
- **Inference engine** - the code at the core of the system which derives recommendations from the knowledge base and problem-specific data in working storage;
- **User interface** - the code that controls the dialog between the user and the system.

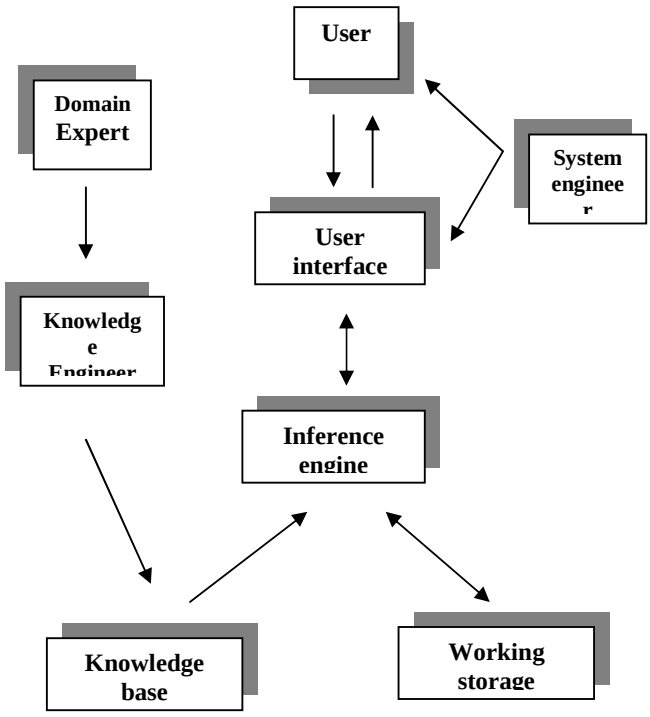

Figure 1 the Expert System Component

5. Importance of Knowledge

Knowledge can be defined as the body of facts and principles accumulated by human kind or the act, fact, or state of knowing. The meaning of knowledge is closely related to the meaning of intelligence. Intelligence requires the possession of and access to knowledge. And a characteristic of intelligence people is that they possess much knowledge.

Thus, we can say that knowledge includes and requires the use of data and information. But it is more. It combines relationships, correlations, dependencies, and the notion of gestalt with data and information.

6. Knowledge in the Expert System

The knowledge the expert uses to solve a problem must be represented in a fashion that can be used to code into the computer and then be available for decision making by the expert system. There are various formal methods for representing knowledge and usually the characteristics of a particular problem will determine the appropriate representation techniques employed.

KBS are computer programs which capture and retain expertise that has been gained over many years of engineering experience and also employ knowledge gained from other (than human) knowledge sources. KBS can reason intelligently about necessary action to take in real time, thus freeing operational staff [9].

Moreover, separation of the knowledge base from the control elements allows the inference engine and algorithms to be generic so they can be applied to a variety of processes. This means that it is possible to begin operating a process with an empty knowledge base

Volume: 02, Issue: 04, Pages: 723-731 (2011)

and create a new knowledge base for the particular process [10].
Knowledge bases can be represented by production rules. These rules consist of a condition or premise followed by an action or conclusion **(IF condition...THEN action)**.

6.1. Knowledge Acquisition

Knowledge acquisition is a time consuming process in which the knowledge engineer works along side the participating expert and extracts, structures and organizes the information to be represented in the expert system.

Knowledge acquisition requires no standard methodology for extracting knowledge. However, it usually involves a progressive number of personal interviews of the expert(s) to record information pertinent to the knowledge-base. Occasionally, the role of the knowledge engineer can be significantly reduced if the understanding of the development processes by the participating experts are substantial and they are willing, able to organize and express all the necessary information to develop facts or rules based on their personal heuristics.

Consistency in the naming conventions of facts or rules is vital, and the ability to develop a language which is familiar to the end users is also important. Acquired knowledge should be played back to experts, perhaps using a different medium than the one used to acquire it. During the knowledge acquisition phase, the knowledge engineer should identify the conclusions that the expert system should render and verify this knowledge as it is acquired. Knowledge acquisition should also be supplemented with a thorough review of current literature to provide the most available up-to-date information.

In the most widely supported view of knowledge acquisition stages subdivide knowledge acquisition activities throughout the life of an expert system development project into the following major tasks:

- Initial entering knowledge.
- Reducing or avoiding erroneous knowledge.
- Augmenting acquired knowledge.

To explain this view we present a framework for knowledge acquisition that identifies the major stages as identification, conceptualization, formalization, implementation, and testing [11].

6.2. Knowledge Engineer

A knowledge engineer has the job of extracting this knowledge and building the expert system knowledge base. Having decided that your problem is suitable you need to extract the knowledge from the expert and represent it using your expert system shell. This is the job of the *knowledge engineer*, but involves close collaboration with the expert(s) and the end user(s).

The knowledge engineer is the AI language and representation expert. He should be able to select a suitable expert system shell (and other tools) for the project, extract the knowledge from the expert, and implement the knowledge in a correct and efficient knowledge base. The knowledge engineer may initially have no knowledge of the application domain.

To extract knowledge from the expert the knowledge engineer must first become at least somewhat familiar with the problem domain, maybe by reading introductory texts or talking to the expert. After this, more systematic interviewing of the expert begins. Typically experts are set a series of example problems, and will explain aloud their reasoning in solving the problem. The knowledge engineer will abstract general rules from these explanations, and check them with the expert [12].

In order to develop the initial prototype the knowledge engineer must make provisional decisions about appropriate knowledge representation and inference methods (e.g., rules, or rules + frames; forward chaining or backward chaining). To test these basic design decisions, the first prototype may only solve a small part of the overall problem. If the methods used seem to work well for that small part it's worth investing the effort in representing the rest of the knowledge in the same form.

6.3. Knowledge Representation

After the domain has been identified and knowledge acquired from a participating expert, a model for representing the knowledge must be developed. Numerous techniques for handling information in the knowledge-base are available; however, most expert systems utilize rule-based approaches. The knowledge engineer, working with the expert, must try to define the best structure possible. Other commonly used approaches include decision trees, blackboard systems and object oriented programming.

Knowledge representation is crucial. One of the clearest results of artificial intelligence research so far is that solving even apparently simple problems requires lots of knowledge. Really understanding a single sentence requires extensive knowledge both of language and of the context. Really understanding a visual scene similarly requires knowledge of the kinds of objects in the scene. Solving problems in a particular domain generally requires knowledge of the objects in the domain and knowledge of how to reason in that domain - both these types of knowledge must be represented.

Also, knowledge representation has been defined as "A set of syntactic and semantic conventions that make it possible to describe things. The syntax of a representation specifies a set of rules for combining symbols to form expressions in the representation language. The semantics of a representation specify how expressions so constructed should be interpreted i.e. how meaning can be derived from a form" [13].

During the last thirty five years several different representation schemes appeared. These schemes have been classified into four categories [14]

7. Knowledge Base Refinement

Knowledge base refinement aims to improve and maintain the performance of knowledge base system (KBS). It is obvious that at no stage of KBS development or exploitation, a perfect performance can be achieved by

Volume: 02, Issue: 04, Pages: 723-731 (2011)

any KBS (except for small-scale application, where exhaustive testing can be performed at the development stage). Therefore, knowledge base refinement is expected to take place constantly through KBS development, and periodically throughout KBS exploitation. During KBS development, knowledge base refinement is concerned with the following two tasks:

Knowledge base revision: This takes place if a structural or function error is revealed in the Knowledge base-theory during its validation, and it consists of introducing appropriate generalizations and/or specializations in Knowledge base-theory to correct the set of conclusions generated by the KBS. The knowledge base revision aims to improve the inferential accuracy of the Knowledge base-theory, and is required when an inconsistency and/or incompleteness is detected during KBS validation.

Knowledge base restructuring: This aims to improve the operating characteristics of the KBS by recognizing and removing sources of potential performance inefficiencies. A common cause for such inefficiencies is redundant, circular or subsumed rules. The restructuring of the Knowledge base-theory is intended to get rid of such rules. It does not change the set of conclusions generated by the system, but it may change the way in which some conclusions are derived, i.e. the knowledge base restructuring aims to eliminate redundancies, subsumptions and circularities from the Knowledge base-theory to assure its convergence and efficiency.

7.1. Refinement, Verification and Validation

The refinement should be thought of as complementary to verification and validation. When a verification and validation phase has been completed, a refinement phase should be executed, with the benefit of the information assembled by verification and validation. A refinement system responds to the existence of evidence suggesting the need for change. So what types of evidence can be provided? Traditionally the evidence takes the form of examples provided by the expert. Here, it is suggested that the faults identified by verification and validation tools are also suitable triggers for refinement. In many cases, evidence consists of the effects of the faults and the major effort within refinement is precisely: ‘‘identifying exactly what should be changed, and how’’. Here it is argued that knowledge refinement is a natural extension to verification and validation systems, and may even be considered as a collaborating system, which gains from the analysis undertaken during faultfinding. Thus refinement, also, becomes an ongoing process throughout the life cycle. [15].

7.2. Refinement and Theory Revision

The field of knowledge refinement is closely related to the subfield of Machine Learning known as theory revision [16], [17]. Traditionally, refinement has been identified as a subfield of Knowledge Acquisition. Early in the KBS development, knowledge acquisition consists of new knowledge being assembled and integrated in the KB. However, as the KB evolves, it is often more appropriate to change existing knowledge rather than always acquire new knowledge. This process of altering knowledge, and possibly incorporating new knowledge, is known as knowledge refinement and is often a distinct step in the final phases of development; most of the KB is acceptable, but small changes are made to the content of the knowledge (not the structure) so that unwanted behavior does not reoccur. As refinement generation becomes more automated it becomes an emerging field in Machine Learning, where it is often referred to as theory revision[15].

7.3. Characterizations of Knowledge Refining Systems

The knowledge refining systems will be characterized in terms of their basic knowledge representation, consistency in refining process compared to initial knowledgebase development, domain knowledge dependence, and the direction of refinement (generation, specialization, or both by adding or deleting conditions). The basic ways of representing knowledge are two-fold: propositional (binary) or predicate (continuous) value representation. While SEEK [17], SEEK2 [19], and KBANN [20] employ propositional attribute value representation, FOIL [21] and GOLEM [22] turn to predicate representation. The main strength of the propositional representation lies in its simplicity. But, this convenience leads to implementation limitations - a new proposition is created every time new attribute values are added. It is also difficult to establish relationships between propositions since they only have true or false values. In terms of maintenance, predicate value representation is more efficient and flexible. Yet there is no standard process for creating predicates, and generalization and/or specialization of knowledge is not straightforward as in the case of propositional representation. If knowledge is represented as propositions, simply checking the existence of propositions is enough; but for predicate representation, the existence of predicates as well as the domain value range must be checked.

8. Knowledge Representation Methodology based XML

The primitives used for knowledge representation (KR) are disorder names, and rule attributes definitions. The attribute definitions are the possible signs and symptoms. The generated signs and symptoms are called findings. In fact, a finding is a symptom or a sign observed by the user. So, the finding consists of three parts: a concept, characteristic or a property, and an associated value(s).

The findings together with the disorder names are used to configure rules. In the other word each rule consists of a disorder name and a set of findings.

XML (Extensible Markup Language) has emerged as the most important format for data exchange and storage over systems across the greatest variety of tools and platforms. It is a rapidly maturing technology with powerful real-word applications, particularly for the

management, display and organization of data. Together with its associated tools (*XSL, XSLT, XPath, XLink, XPointer, DOM DTD, Schemas,* …etc), it is an essential technology for anyone looking for more efficient and cost effective ways of both managing and transferring data. Perhaps the most well known applications are web related, but there are many other non-web based applications where XML is useful–for example as a replacement for (or to complement) traditional database [23]. A lot of languages that are based on XML are around, e.g., RDF [24], OIL [25], DAML+OIL, OWL, etc. Those importances of those languages are related to semantic web [26]. To realize the semantic web vision, it was necessary to express semantics of data using those languages. In our work, we propose a new language to express the semantic of the well-known knowledge representation schemes as for instance: rules, dependency graph, frame, semantic network, etc. Further, XML tools have been used to develop a library of inference methods to be used for reasoning on different representations.

An XML document is a hierarchically structured and self-describing piece of information, and consists of atomic elements or complex elements (elements with nested subelements). An XML document incorporates structure and data in one entity. To this extent, XML data is semistructured data.

The processing and management of XML data are popular research issues. However, operations based on the structure of XML data have not received strong attention. These operations involve, among others, the grouping of structurally similar XML documents. Such grouping results from the application of clustering methods with distances that estimate the similarity between tree structures.

The advantages of using the XML as a knowledge representation scheme are:

- Providing compatibility between different tools and even between different versions of the same tool.
- Preparing reports for reviewing of knowledge base by domain experts
- Deploying expert systems applications efficiently on the Web
- Providing a readable and elegant explanation of deducted conclusions.
- Facilitating knowledge base refinement and optimization.
- Supporting knowledge base verification, validation and maintenance.
- Any reasoning over this representation improves the performance of the knowledge-based system by reducing the response time of knowledge-based expert system and minimizing the number of generated questions.
- Facilitating easy transmission of structured data over existing network protocol [27].

8.1. Rule Representation

We propose an XML-based to represent the knowledge. In this representation the rule is represented as parent node and its findings are represented as a set of child nodes (subnodes).

So the parent node consists of four parts: *Rule* tag, *Name* attribute, *Disorder* attribute, and *NoTrueFindings* attribute, where:

The *Name* attribute represents the rule id.

The *Disorder* attribute represents the disorder name.

The *NoTrueFindings* attribute represents the number of the findings in the rule and which are in working memory.

The parent node is expressed as:

<Rule *Name*=value *Disorder*=value *NoTrueFindings*=value>.

Every element in the set of subnodes consists of five parts: *Finding* tag, *Cpt* attribute, *Prop* attribute, *Val* attribute, *Equal* attribute, and ExistInWM attribute where, *Cpt, Prop, Va, Equal,* and *ExistInWM* the concept of the finding, the property of the finding, the value of the finding, the present or absence of the finding, the existence of the finding in the working memory; respectively.

Every child node is expressed as:

<Finding *Cpt*="Concept" *Prop*="Property" *Val*="value" *Equal*="Yes/No" *ExistInWM*="Yes/No" />

The following example explains how rule structure is represented using XML format.

8.1.1 Working Example

Suppose that the Red Spider disorder existed when the color of the leaf spot is yellowish green, the position of the leaf spot is upper surface, and the color of the leaf is dirty. We can represent this rule using our representation as follow:

<*Rule* *Name*="rule19" *Disorder*="Red_spider" *NoTrueFindings* ="0">
<*Finding* *Cpt*="Leaves_obs" *Prop*="Spots_color" *Val*="yellowish green" *Equal*="Yes" *ExistInWM*="No" />
<*Finding* *Cpt*="Leaves_obs" *Prop*="Spots_position" *Val*="upper surface"" *Equal*="Yes" *ExistInWM*="No" />
<*Finding* *Cpt*="Leaves_obs" *Prop*="Color" *Val*="dirty" *Equal*="Yes" *ExistInWM*="No" />
</Rule>

8.2. Complete Example

Suppose the original knowledgs base contained the following rules:

R1: F1, F3, F6 ⟶ D1
R2: F1, F8 ⟶ D1
R3: F2, F5, F6 ⟶ D2
R4: F4, F6, F7 ⟶ D3
R5: F3, F7, F9 ⟶ D4

R6: F5, F7, F8 ⟶ D5

Initially the refined knowledge base (RKB) is empty, so the following table illustrates the steps of the algorithm.
Rule no = R1
FindingsSet = [F1,F3,F6]
PowerSetOfFindingsSet =
[[F1],[F3],[F6],[F1,F3],[F1,F6],[F3,F6],[F1,F3,F6]]

SubSumsionExist	***Disorders List***	***IdenticalSet***	***RKB***
(F1,RK)=False	[D1, D1]	True	F1 → D1
(F3,RK)=False	[D1,D5].	False	
(F6,RK)=false	[D1, 2,D5]	False	
([F1,F3],RK)=True			
([F1,F6],RK)=True			
([F3,F6],RK)=False	[D1]	True	F3, F6 →D1
([F1,F3,F6],RK)=True			

Finally the refined knowledge base contains the following rules

RR1: F1 ⟶ D1
RR2: F3, F6 ⟶ D1
RR3: F2 ⟶ D2
RR4: F4 ⟶ D3
RR5: F6,F7 ⟶ D3
RR6: F9 ⟶ D4
RR7: F3, F7 ⟶ D4
RR8: F5, F7 ⟶ D5
RR9: F5, F8 ⟶ D5
RR10: F7, F8 ⟶ D5

9. Conclusions

In this paper, most of the major features of the XML Language have been discussed. Hopefully the reader will have enough background to pick up and read the XML specifications without difficulty.

The primary objective of this paper is to develop a computer-based diagnostic expert system that can be used in the diagnosis and treatment of diseases. In other words, developing our expert system is web using XML Language.

Thus, we developed three different rule-based systems, each designed to take XML as an input, and produce XML as an output and manipulate intermediate facts as XML. They used very different methods of representing the XML during rule processing.

So we developed this XML program to be used on the internet, support wide variety of applications and make it easy to write programs which process XML documents which will be human- legible and reasonably clear and easy to create.

The Knowledge refinement has an important role to play in the process of knowledge acquisition and maintenance. However, there is a difference of emphasis; knowledge refinement is more concerned with improving the performance of existing knowledge base systems.

An optimal knowledge base will consist of a set of rules, which have a minimal set of findings. The number of rules and their findings in the initial knowledge base is directly proportional to the optimization ratio. In this work, the verified, and refined knowledge bases cover the initial knowledge bases. The refined, and verified knowledge bases do not only have an optimized number of rules, and findings, but they also have optimized and enhanced three types of reasoning mechanism.

References

[1] Gilbert Tekli, Richard Chbeir Towards an XML Adaptation/Alteration Control Framework. 2010 Fifth International Conference on Internet and Web Applications and Services. IEEE pp 248-255, 2010.

[2] Awad H. Khalil. A framework for Security Enforcement in XML-based B2B Applications. In Proceedings of The Eleventh International Conference On Artificial Intelligence Application, ICAIA, pp: 77-84, 2003.

[3] J.Broekstra et al. Enabling Knowledge Representation on the web by Extending RDF Schema. Proc. 10th Int'l World Wide Web Conf., Hong Kong, 2001.

[4] Ora Lassila and Ralph R. Swick. Resource Description Framework (RDF) Model and Syntax Specification. Recommendation REC-rdf-syntax-19990222, W3C, February 1999.

[5] Buchanan, B.G. and Shortliffe, E.H.. Rule-Based Expert System: The MYCIN Experiments of the Stanford Heuristic Programming Project. Reading, MA: Addison-Wesley, 1984.

[6] Zlatareva, N.. A Refinement Framework to Support Validation and Maintenance of Knowledge-Based System. Expert System with Application, vol. 15, pp.245-252, 1998.

[7] Blaž Zupan , Albert Mo Kim Cheng. Optimization of Rule-Based Systems Using State Space Graphs. IEEE Transactions On Knowledge And Data Engineering, Vol. 10, No. 2, March/April,1998.

[8] Schnupp, P.H.; "Building Expert System in Prolog ", Amiz, 2000.

[9] S.G. Tzafestas, H.B. Verbruggen, Artificial intelligence in industrial decision making control, and automation: an introduction, in: S.G. Tzafestas, H.B. Verbruggen (Eds.),, Artificial Intelligence in Industrial Decision Making, Control and Automation, Kluwer, The Netherlands, , pp. 1–39, 1995.

[10] H. Wang, D. Linkens, Intelligent supervisory control. A qualitative bond graph reasoning approach, World Scientific, Singapore, 1996.

[11] Minsky M.; " Semantic Information Processing ", Cambridge, Mass. :MIT Press , 1968.

[12] Jain, L. and A. M. Fanelli; "Recent Advances In Artificial Neural Networks Design And Applications", CRC Press, 2000.

[13] Barr A. and E. A. Feigenbaum, eds.; "The Handbook of Artificial Intelligence",Vol. 1, Los Altos, Calif: Morgan Kaufman, 1981.
[14] George F. Luger and William, A. Stubblefield; " Artificial Intelligence and the Design of Expert Systems", The Benjamin/ cummings publishing Co., Inc., 1989.
[15] Susan Craw.. Refinement complements verification and validation. Int.J.Humam-Computer Studies, 44(2), 245-256, 1996.
[16] Adé, H., Malfait, B. & De Raedt, L. RUTH: an ILP Theory Revision System. Eighth International Symposium on Methodologies for Intelligent Systems (ISMIS94), 336-345, 1994.
[17] Ourston, D. & Mooney, R. J. Theory Refinement Combining Analytical and Empirical Methods. Artificial Intelligence, 66, 273-309, 1994.
[18] Politakis, P., and Weiss, S. Using Empirical Analysis to Refine System Knowledge Bases. Artificial Intelligence, Vol. 22, pp. 23-48, 1984.
[19] Ginsberg, A., Weiss, S., and Politakis, P. A Generalized Approach to Automate Knowledge Based Refinement. Proceedings of 9th IJCAI, Vol. I, Los Angeles, California, pp. 18-23, 1985.
[20] Shavlik, J. W., and Towell, G. G. An Approach to Combining Explanation-Based and Neural Learning Algorithms. Connection Science, Vol. 1, pp. 233-255, 1989.
[21] Quinlan, J. R. Learning Logical Definitions from Relations. Machine Learning,Vol. 5,pp.239-266, 1990.
[22] Muggleton and Feng. Efficient Induction of Logic Programs. Proceedings of the First Conference on Algorithmic Learning Theory, Ithaca, New York, 1990.
[23] Awad H. Khalil. A framework for Security Enforcement in XML-based B2B Applications. In Proceedings of The Eleventh International Conference On Artificial Intelligence Application, ICAIA, pp: 77-84, 2003.
[24] J.Broekstra et al. Enabling Knowledge Representation on the web by Extending RDF Schema. Proc. 10th Int'l World Wide Web Conf., Hong Kong, 2001.
[25] Ora Lassila and Ralph R. Swick. Resource Description Framework (RDF) Model and Syntax Specification. Recommendation REC-rdf-syntax-19990222, W3C, February 1999.
[26] 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, 284(5): 34-43, May 2001.
[27] Bray, T., Paoli, J., Sperberg-McQueen, C. M. Eds. Extensible Mark-up Language, W3 Consortium recommendation paper., http://www.w3.org/TR/1998/ REC-xml-19980210, Feb. 1998.

Authors Biography

DR. Yasser Ahmed Nada Was born in Ismailia, Egypt, in 1968. He received the BSc degree in pure Mathematics and Computer Sciences in 1989 and MSc degree for his work in computer science in 2003, all from the Faculty of Science, Suez Canal University, Egypt. In 2007, he received his Ph.D. in Computer Science from the Faculty of Science, Suez Canal University, Egypt. From September 2007 until now, he worked as a lecturer of computer science, Faculty of Computers and Information Systems Taif University, KSA. His research interests include Expert Systems, Artificial Intelligence, Object Oriented Programming, Computer Vision, and Genetic.