INTERNATIONAL JOURNAL OF COMPUTERS
Issue 4, Volume 4, 2010

Embedding Conditional Knowledge Bases into Question Answering Systems and Java Implementation

Nicolae Ţăndăreanu, Mihaela Colhon, Cristina Zamfir

Abstract—A conditional schema is a graph-based structure which is able to represent conditional knowledge. This structure was introduced in [11]. The inference mechanism corresponding to the conditional schema representations was developed in [12]. In this paper we propose a question answering system that can represent and process conditional knowledge using these mechanisms. In order to accomplish this task we refine the concept of conditional schema by introducing the concepts of XML-conditional structure generated by a conditional schema and XML-conditional knowledge base for such a structure. We describe the architecture of a question answering systems that uses these structures. An implementation by means of Java platform is briefly described.

Keywords—conditional schema, conditional knowledge, interface by voice, graphical user interface, XML, dialogue system, question answering system

I. Introduction

A Spoken Dialogue System (SDS) is a software system that accepts natural language as input and produces natural language as output engaging in a conversation with an user. To successfully manage the interaction with users, SDS usually carry out five main tasks: *Automatic Speech Recognition* (ASR), *Natural Language Understanding* (NLU), *Dialogue Management* (DM), *Natural Language Generation* (NLG) and *Text-to Speech Synthesis* (TSS). These tasks are usually implemented in different modules. In general by a *Question Answering System* we understand a SDS designed to provide answers to questions that are formulated by user in natural language. To find the answer to a question, a question answering system may use a database, a collection of natural language documents or a knowledge base. Various such systems were developed. The system `START` ([10]) extracts the information contained by an English text and obtains a knowledge base. The user has access to information by querying the knowledge base. The system formulates the answer in English. The architecture of the `AskMSR` question answering system is presented in [4], where the strategies for predicting when the question answering system is likely to give an incorrect answer are also explored. `Snowball` ([1]) introduces novel strategies for generating patterns and extracting tuples from plain-text documents. This system was developed for extracting structured data from plain-text documents with minimal human participation. `AnswerBus` ([13]) is an open-domain question answering system based on sentence level Web information retrieval. From the Web pages, AnswerBus extracts sentences that are determined to contain answers. A mixture of Natural Language Processing and Information Retrieval can be encountered in `Quanda` system ([2], [3]). `MAYA` ([9]) is a question answering system in Korean that uses a predictive answer indexer. A model for answer extraction component of a question answering system called `Sbuqa` is presented in [14]. The Lexical Functional Grammar, a meaning based grammar that analyses sentences in a deeper level than syntactic parsing are used to represent the question and candidate answers. The papers [6], [7] and [8] belong to a sequence of papers for Romanian case. The reader interested to use the question answering systems for knowledge based on the Web can find an interesting approach in [5].

In this paper we describe an architecture and a Java implementation of a spoken question answering system based on conditional knowledge. The main features of this system are the following ones:

- The user-system communication is based on a voice user interface.
- The information is represented into a knowledge base that uses conditional knowledge.
- The implementation is platform independent because Java technology is used.
- The system can be extended to obtain dialogue systems based on conditional knowledge;
- The product can be used successfully in automatic training, to build case-based consultancies or systems for diagnosing the malfunctions of a device.

This paper is organized as follows: in Section II a short presentation of the conditional schema components is given, the inference mechanism is briefly described and the concepts of XML-conditional structure generated by a conditional schema and XML-conditional knowledge base for such a structure are introduced; Section III contains the description of the architecture for a spoken question answering system based on the concepts defined in Section II; in Section IV we describe a Java implementation of such a system; the last section includes several possible extensions of our study.

N. Ţăndăreanu, Faculty of Mathematics and Computer Science, University of Craiova, http://inf.ucv.ro/~ntand/en/), email: ntand@rdslink.ro

M.Colhon, Faculty of Mathematics and Computer Science, University of Craiova, email: mcolhon@inf.ucv.ro

C.Zamfir, Faculty of Mathematics and Computer Science, University of Craiova, email: cristina.zamfir@star-storage.ro

This article was produced under the project "Supporting young Ph.D students with frequency by providing doctoral fellowships", co-financed from the EUROPEAN SOCIAL FUND through the Sectoral Operational Program Development of Human Resources

II. CONDITIONAL KNOWLEDGE

In the first subsection we recall the concept of conditional schema as was introduced in [11]. This is the basic concept which allows to develop the concepts presented in the second section and then on this basis we can present the architecture of a question answering system and several details concerning the corresponding implementation.

A. Conditional schemas and the inference

Most of the rule-based systems are developed starting with a given set of rules. Thus, various inputs of the system are applied on the same rules in the reasoning process. This is an important restriction that can be avoided by designing systems for which the inference rules are extracted from the inputs and furthermore applied on data specified also by the user. Such inputs are specialized forms of knowledge pieces, named *conditional knowledge pieces* which, as opposite to classical knowledge pieces, can contain sentences that describe rules. The conditional knowledge representation and processing mechanism was developed under the name of *conditional schema*.

The formal concept of conditional schema was introduced in [11]. The inference mechanism of this structure is treated in [12]. A conditional schema is a tuple $\mathcal{S} = (Ob, C_s, E_r, A, V, B_{cr}, h, f)$ such that:

- $Ob = Ob_{ind} \cup Ob_{abstr}$, where Ob_{ind} is the set of the individual objects and Ob_{abstr} is the set of the abstract objects; we suppose that $Ob_{ind} \cap Ob_{abstr} = \emptyset$;
- C_s is the set of conditional mappings;
- E_r is the set of the symbols for conditional binary relations;
- A is the set of attribute names for the elements of Ob_{ind};
- V is a set of values for the elements of A;
- $B_{cr} \subseteq 2^{((Ob\times I)\times(Ob\times I))\times C_s}$ is the set of conditional binary relations and $I = \{i, a\}$, where i is used to designate individual objects and a is used to specify abstract objects. Thus an individual object is specified as (x, i) and an abstract object has the form (x, a).
- $h : E_r \longrightarrow B_{cr}$ maps a conditional binary relation for every symbol of E_r;
- $f : Ob_{ind} \longrightarrow 2^{A\times V}$ assigns declarative knowledge to the individual objects of Ob_{ind}.

The conditional graph ([11]) generated by $\mathcal{S}$ is the system $G_{\mathcal{S}} = (X \cup Z, \Gamma_X \cup \Gamma_Z)$, where:

- $X \subseteq Ob \times I$ is the set of nodes such that $x \in X$ if and only if there are $r \in E_r$, $y \in X$ such that $(x, y) \in_c h(r)$ or $(y, x) \in_c h(r)$;
- $\Gamma_X \subseteq X \times E_r \times X$ and $((n, \omega_1), r, (m, \omega_2)) \in \Gamma_X$ if and only if $((n, \omega_1), (m, \omega_2)) \in h(r)$; the elements of Γ_X are named *arcs of first category*;
- $Z = \{f(x) \mid x \in Ob_{ind}\}$ and $\Gamma_Z = \{(f(x), x) \mid x \in Ob_{ind}\}$; the elements of Γ_Z are named *arcs of the second category*.

This inference mechanism is based on a graph-based structure, the conditional graph, and therefore is a path-driven reasoning mechanism. The conditional graph of a conditional schema contains two kinds of nodes: individual nodes and abstract nodes. An individual node is characterized by pairs of the form *(attribute,value)*, where *attribute* represents an attribute name and *value* gives the value of the corresponding attribute. A path from n_1 to n_{k+1} is a pair $d = ([(n_1, \omega_1), \ldots, (n_{k+1}, \omega_{k+1})], [a_1, \ldots, a_k])$, where $n_1, \ldots, n_{k+1} \in X$, $\omega_1, \ldots, \omega_{k+1} \in \{a, i\}$ and $a_1, \ldots, a_k \in E_r$. The knowledge engineer must define a partial mapping Φ such that $\Phi([a_1, \ldots, a_k])$, $k \geq 2$, is a new label representing a "compound" label of $a_1, \ldots, a_k$. We denote by E_r^* the union set of E_r with the set of the compound labels. In order to apply the inference mechanism we suppose that:

- there is $j \in \{1, \ldots, k+1\}$ such that $w_j = i$
- $[a_1, \ldots, a_k] \in dom(\Phi)$

where by $dom(\Phi)$ we denote the domain of the mapping Φ. In order to interrogate such a system, the user specifies two nodes α_1 and α_2. The node α_1 must be an individual node. Each arc (k_i, k_j) of a given path contains two kinds of labels:

- a label to specify a binary relation between k_i and k_j; this is the set E_r from the definition of a conditional schema;

- a label that identifies a certain condition which must be verified/satisfied by the nearest individual object of k_i; the set of these labels is C_s and the connection between an element of E_r and the conditional binary relations is given by the mapping h from the definition of a conditional schema;

The condition imposed on the arc (k_i, k_j) can be viewed as a semaphore. The value of a semaphore is computed by means of some rule of the form IF-THEN-ELSE, where the attribute values of the individual objects are used. If the condition is $true$ then the semaphore is "on", otherwise is "off". If all semaphores of the path d are "on" then some conclusion is obtained by the inference mechanism. If some semaphore is "off" then either no conclusion is obtained or a "negative" conclusion is specified.

The answer mapping is defined using a semantic function Sem. If G is the grammar mapped on the Recursive Transition Network defined for the user-system dialogue processing and $L(G)$ is the language generated by G then $Sem : Ob \times E_r^* \times Ob \times \{\text{on, off}\} \longrightarrow L(G)$. This mapping is defined by the knowledge engineer.

Let us consider the path $d = ([(\alpha_1, \omega_1), \ldots, (\alpha_{k+1}, \omega_{k+1})], [a_1, \ldots, a_k])$. We suppose that:

- there is $j \in \{1, \ldots, k+1\}$ such that $w_j = i$
- $[a_1, \ldots, a_k] \in dom(\Phi)$
- $[t_1, \ldots, t_k]$ is the list of all conditional symbols of d, where the order of the arcs of d is preserved by this list.

We define $ans(d)$ as follows:

- If $t_1[d] = \ldots = t_k[d] =$ on and $(\alpha_1, \Phi([a_1, \ldots, a_k]), \alpha_{k+1}, on) \in dom(Sem)$ then $ans(d) = Sem(n_1, \Phi([a_1, \ldots, a_k]), n_{k+1}, \text{on})$.
- If there is $u \in \{1, \ldots, k\}$ such that $t_u[d] =$off and $(\alpha_1, \Phi([a_1, \ldots, a_k]), \alpha_{k+1}, \text{off}) \in dom(Sem)$ then $ans(d) = Sem(\alpha_1, \Phi([a_1, \ldots, a_k]), \alpha_{k+1}, \text{off})$
- $ans(d) = unknown$ otherwise.

B. Conditional Knowledge Bases

In this subsection we refine the concept of semantic schema. Because the implementation presented in this paper uses elements of XML, the next two definitions establishes the *general entities* of a conditional schema and the *particular entities* of this concept. We consider a conditional schema $\mathcal{S} = (Ob, C_s, E_r, A, V, B_{cr}, h, f)$.

Definition 2.1: An **XML-conditional structure** generated by $\mathcal{S}$ is a pair (U, D) such that

- $U = (I_{ob}, A_{ob}, C_s, E_r, A, V, h, f, u)$, where
  - I_{ob} is the set of symbols for all individual objects of $\mathcal{S}$;
  - A_{ob} is the set of symbols for all abstract objects of $\mathcal{S}$;
  - $u : E_r \longrightarrow (I_{ob} \cup A_{ob}) \times (I_{ob} \cup A_{ob})$;
  - C_s, E_r, A, V, h, f are specified in $\mathcal{S}$.
- D is the XML-description of the structure U as follows:

```
<?xml version="1.0" encoding="UTF-8"
standalone="no"?>
<conditional_schema>
<iobjs>
i1
i2
.......
</iobjs>
<aobjs>
a1
a2
.......
</aobjs>
<CS>
s1
s2
.......
</CS>
<ER>
r1
r2
.......
</ER>
<Attributes>
at1
at2
</Attributes>
<Values>
v1
v2
.......
</Values>
<h_mapping>
h(r1)={(((i1,i),(i2,i)),T(i1))}
h(r2)={(((i1,i),(i3,i)),T(i1))}
  .......
</h_mapping>
<f_mapping>
f(i1)=(at1,v1)
f(i1)=(at1,v2)
.......
</f_mapping>
<u>
u(r1)=(i1,i2)
u(r2)=(i1,i3)
.......
</u>
</conditional_schema>
```

Definition 2.2: An **XML-conditional knowledge base** for the structure (U, D) is an XML file which establishes values for the entities of this structure and defines also the mappings ϕ, Φ and Sem:

```
<?xml version="1.0" encoding="UTF-8"
standalone="no"?>
<knowledge_base>
<natural_language_text>
  .......
  text of the knowledge piece
  .......
</natural_language_text>
<iobjs>
i1=Helen
.......
</iobjs>
<aobjs>
a1=pizza
a2=cheese cake
.......
</aobjs>
<CS>
s1
s2
</CS>
<ER>
r1=is mother
r2=is sister
.......
</ER>
<Attributes>
at1=age
.......
</Attributes>
<Values>
v1=40
.......
</Values>
<Reg>
s1=R1(i3)=IF(y,i3) in u(r2) and
f(y)=(at2,v2) then s1(i3)=on else
s1(i3)=off
  .......
</Reg>
<phi>
phi(r1,r2)=b1
.......
</phi>
<Big_phi>
```

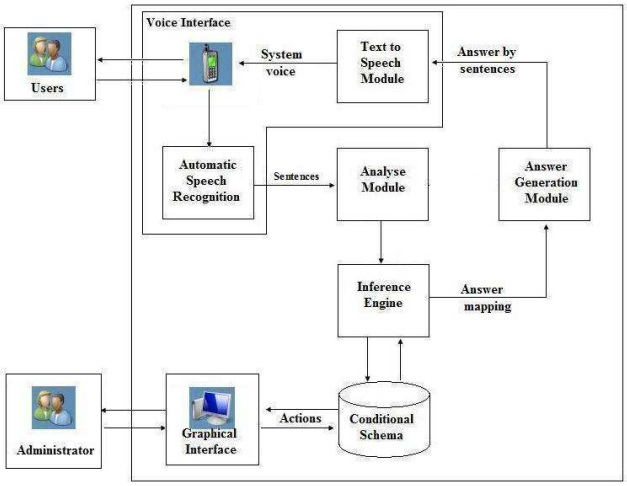

Fig. 1. An architecture for conditional schema implementation

```
PHI([r1,r2])=b1
.......
</Big_phi>
<sem>
Sem(x,r1,y,on) = x is siter of y
Sem(x,r1,y,on) = x is not sister of y
.......
</sem>
</knowledge_base>
```

III. The system architecture

A possible architecture is shown in Figure 1. In this section the components of the system and the connection between them are described.

- The communication between user and system is based on a *voice interface* which includes the module of speech recognition and the module of speech synthesis; a Recursive Transition Network is used to guide the communication in natural language. A conditional knowledge piece that can be represented and processed by our model can contain two kinds of information:

○ *declarative knowledge* - propositions or facts describing the current state of the problem (facts that are known to be true); the user can submit questions which refer only to declarative knowledge.

○ *procedural knowledge* - the logical rules: IF *condition* THEN *action* ELSE *action*; these rules are expressed in natural language and are used to compute the values of the conditional symbols.

- The main task of the *Analyse Module* is to extract the semantics from the received sentences.

In this module, the text transferred from the Automatic Speech Recognition is analyzed and the semantics of the text are extracted. This is done by means of a Recursive Transition Network mechanism for natural language processing. We consider that, in time, due to the different use of the application, the network must addapt according to the needs of the end users. This is why the Recursive Transition Network components are saved in a file, and in this manner the network can be easily reconfigured at a later point in time. If the text obtained by the automatic speech module is grammatically correct then, using a dedicated algorithm, the objects and relations can be extracted from the sentences. The Analyse Module sends to Inference Engine module the objects to perform the inference. If the text is not grammatically correct then the message received from the user will be considered as not valid.

- The *Conditional Schema* module contains the conditional schema component of the system;

- The *Inference Engine* module performs the inferences (receives two nodes and gives the result of the computation).

- The *Answer Generation* module receives the values of the answer mapping generated by the Interface Engine module and produces the answer sentences. We relieve the following facts:

○ $Sem(x, a, y, \text{on})$ specifies the semantics of the relation between the objects x and y, which is identified by the symbol *a*;

○ $Sem(x, a, y, \text{off})$ specifies the converse property.

For example, if $Sem(Peter, is\_a, student, \text{on})$ = *Peter is a student* then $Sem(Peter, is\_a, student, off)$ = *Peter is not a student.*

Frequently the answer given by this module is not the same as the sentence received from inference engine. This can be viewed from the following example. Suppose that the user ask the system ”Does Maria like to eat pizza?” If the value of the mapping ans is the sentence ”A nephew of Maria likes to eat pizza” then the answer given by this module is the sentence ”No, a nephew of Maria likes to eat pizza”.

- The *Graphical Interface* module is an interface for the administrator. By means of this interface the knowledge engineer or the administrator performs the following tasks:

- create a conditional schema;
- add new components to the conditional schema;
- modify the existing components of the conditional schema;
- delete the components of the conditional schema; delete the conditional schema; visualize a conditional schema;
- define, modify and visualize the mapping Φ;
- define, modify and visualize the mapping Sem.

IV. Java implementation

The implementation was performed by means of Java platform. The *Java Speech API* is a standard extension to the Java platform that enables Java applications to use speech input and output. This API was used to implement the module *Speech to Text*. Sphinx-4 is a speech recognition system written entirely in Java and this product was used to implement the module *Automatic Speech Recognition*. This section gives several details concerning this implementation, the application execution stages and some application captures.

After the application is installed the first step is to configure the conditional schema. This is done by an user with administrative capabilities through a graphical interface.

As shown in Figure 2, the administrator can:

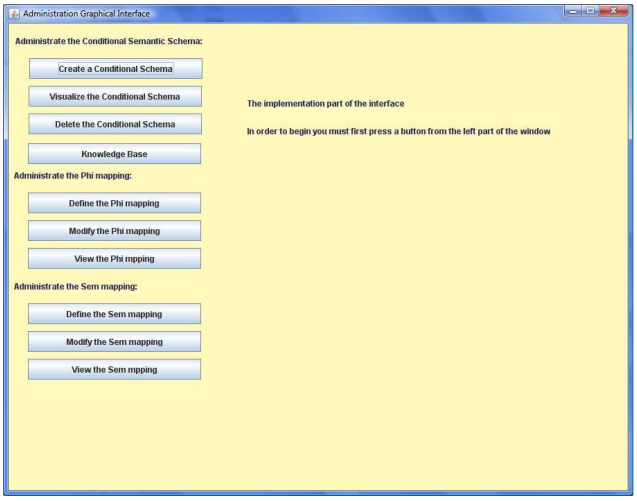

Fig. 2. Capture of the Administrator Graphical Interface

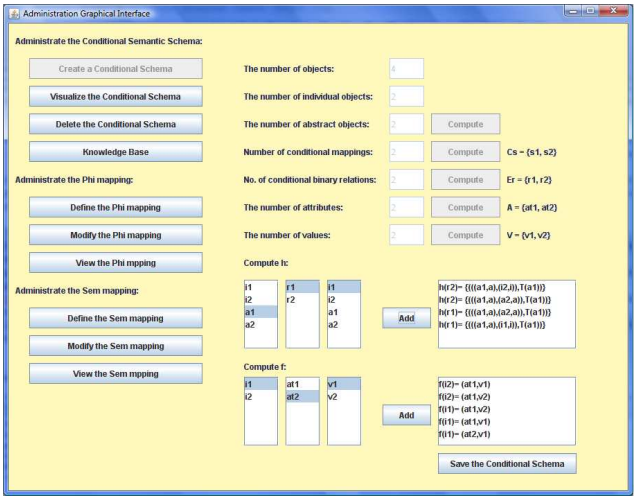

Fig. 3. Create a conditional schema

- Create a Conditional Schema.
- View the Conditional Schema.
- Delete the Conditional Schema.
- Create a Knowledge Base.
- Define, Edit and Delete the Φ mapping.
- Define, Edit and Delete the Sem mapping.

When creating a Conditional Schema the administrator will introduce the number of individual and abstract objects. The application will create unique symbols for objects. The next step is to define the conditional mappings, that is the elements of the set C_s, the sets E_r and A and the mapping h. After completing this operations the user can save the conditional schema. By pressing the button "Visualize the Conditional Schema" the administrator can further view and modify the prior created schema. Modifying the schema is done by adding or removing symbols of objects or of conditional mappings or by adding or removing elements of the sets C_s, E_r or A.

In Figure 3 is presented the graphical controls by means of which the user can construct a Conditional Schema. When the process is finished the user must press the "Save Conditional Schema" button. This will save the schema in an `xml` file format that will be used later on by the application when interacting with regular users.

The administrator can view or modify the schema later on by pressing the "Visualize Conditional Schema" button. The schema can be viewed and modified directly from the xml file or through the application graphical interface. This file shows as follows:

```
<?xml version="1.0" encoding="UTF-8"
standalone="no"?>
<conditional_schema>
<iobjs>
i1
i2
i3
i4
</iobjs>
<aobjs>
a1
a2
a3
</aobjs>
<CS>
s1
s2
</CS>
<ER>
r1
r2
r3
r4
</ER>
<Attributes>
at1
at2
</Attributes>
<Values>
v1
v2
</Values>
<h_mapping>
h(r1)={(((i1,i),(i2,i)),T(i1))}
h(r2)={(((i1,i),(i3,i)),T(i1))}
h(r2)={(((i2,i),(i4,i)),T(i2))}
h(r3)={(((i3,i),(a2,a)),s1)}
h(r3)={(((i4,i),(a3,a)),s2)}
h(r4)={(((i1,i),(a1,a)),T(i1))}
</h_mapping>
<f_mapping>
f(i1)=(at1,v1)
f(i1)=(at1,v2)
f(i1)=(at2,v1)
f(i1)=(at2,v2)
f(i2)=at1,v1)
f(i2)=(at2,v2)
f(i2)=(at2,v1)
f(i2)=(at1,v2)
```

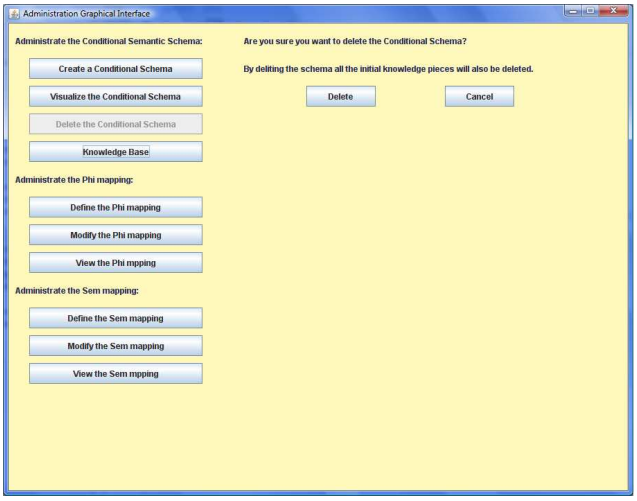

Fig. 4. Delete a conditional schema

```
</f_mapping>
<u>
u(r1)=(i1,i2)
u(r2)=(i1,i3)
u(r2)=(i2,i4)
u(r3)=(i3,a2)
u(r3)=(i4,a3)
u(r4)=(i1,a1)
</u>
</conditional_schema>
```

When the knowledge engineer wants to delete the conditional schema it is very important to remember that also the initial knowledge pieces represented on this schema will be deleted. The graphical interface by means of which the conditional schema can be deleted is shown in Figure 4.

When the ”Knowledge Base” button is pressed the interface used to built a knowledge base is shown. Through this interface the administrator establishes connections between the symbols used in the conditional schema representations and the names of the represented entities. Also he can define the Φ and Sem mappings.

The administrator introduces the symbols of the necessary relations, and based on them the application creates the symbols for the composed relations. Is then the task of the administrator to attach semantics, by means of the Sem function, to the resulted relation symbols. The Φ and Sem mappings are also saved in an xml file format. This file can show as follows:

```
<?xml version="1.0" encoding="UTF-8"
standalone="no"?>
<knowledge_base>
<natural_language_text>
Helen is the mother of Peter. She is 40.
Helen made cheese cake. Peter likes to
eat cheese cake if this is made by his
mother. Susan is Helen’s sister and
George is her son. George likes to eat
fruits if they are bought by his mother.
Susan is 30. She bought some bread.
Helen likes to eat pizza.
</natural_language_text>
<iobjs>
i1=Helen
i2=Susan
i3=Peter
i4=George
</iobjs>
<aobjs>
a1=pizza
a2=cheese cake
a3=fruits
</aobjs>
<CS>
s1
s2
</CS>
<ER>
r1=is mother
r2=is sister
r3=likes
r4=eats
</ER>
<Attributes>
at1=age
at2=made
at4=bought
</Attributes>
<Values>
v1=40
v2=cheese cake
v3=30
v4=fruits
</Values>
<Reg>
s1=R1(i3)=IF(y,i3) in u(r2) and
f(y)=(at2,v2) then s1(i3)=on else s1(i3)=off
s2=R2(i4)=IF(y,i4) in u(r2) and
f(y)=(at4,v4) then s2(i4)=on else s2(i4)=off
</Reg>
<phi>
phi(r1,r2)=b1
phi(b1,r3)=b2
phi(r2,r3)=b3
</phi>
<Big_phi>
PHI([r1,r2])=b1
PHI([r1,r2,r3])=b2
PHI([r2,r3])=b3
</Big_phi>
<sem>
Sem(x,r1,y,on) = x is siter of y
Sem(x,r1,y,on) = x is not sister of y
Sem(x,r2,y,on) = x is mother of y
Sem(x,r2,y,off) = x is not mother of y
Sem(x,r3,y,on) = x eats of y
```

```
Sem(x,r3,y,off) = x does not eat of y
Sem(x,r4,y,on) = x likes to eat of y
Sem(x,r4,y,off) = x does not like to
eat of y
Sem(x,b1,y,on) = x is aunt of y
Sem(x,b1,y,off) = x is not aunt of y
Sem(x,b2,y,on) = the nephew of x
eats y
Sem(x,b2,y,off) = the nephew of x
does not eat y
Sem(x,b3,y,on) = The sun of x eats y
Sem(x,b3,y,off) = The sun of x does
not eat y
</sem>
</knowledge_base>
```

The Java API for XML Processing (JAXP) is for processing XML data using applications written in the Java programming language. JAXP leverages the parser standards Simple API for XML Parsing (SAX) and Document Object Model (DOM).

To process all the xml files that this application uses we have worked with DOM and SAX. DOM is the acronym for Document Object Model, which is a API component of the Java API for XML Processing. A DOM is a garden-variety tree structure, where 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 Simple API for XML (SAX) is the event-driven, serial-access mechanism that does element-by-element processing. The API for this level reads and writes XML to a data repository or the web. The interfaces provided in the SAX package is an important part of our toolkit for handling XML.

After the administration part of the application is properly configured, the end users can interrogate the application. An user will connect to the application by phone or through a microphone. The application will transform the verbal signal into text using the Sphinx4 module. The obtained text will be submitted to the following algorithm:

- Parsing - Dividing the proposition into words;
- Semantical and grammatical evaluation - In order to analyze a text first of all we must determine if the text is grammatically correct (from the defined grammar point of view);
- Extracting objects and relations - in this step the names of the objects and of the relations described in the input sentences are extracted;
- Applying Simb - in order to align the names of the objects indicated by the user to the internal representations of the system, stored in the conditional schema, the names of the objects are assigned to the corresponding symbols of Ob_{ind} or Ob_{abstr}.

Once the message introduced by the user is interpreted, the inference can be performed.

Using the initial knowledge pieces saved in xml format, the objects and the relations that where described by the user through a spoken message will be identified in the system's conditional schema.

If an object or an relation can not be identified in the conditional schema, the Inference Engine will send an error message to the Answer Generation Module that will transform it in a spoken message in which the user will be asked to contact the administrator by e-mail or telephone. This approach will ensure that the user will receive the answer for his question and that the administrator will be informed of the issue that needs to be corrected.

If all the objects and relations indicated by the user can be identified in the conditional schema, the inference mechanism and the answer generation module will calculate and formulate the natural language answers to the user questions.

More precisely, the answer generated by the Inference Engine will be sent to the Answer Generation Module where the corresponding natural language response can be generated. The text will be then transmitted to the Text to Speech module that will transform the text into a spoken message.

For this application we used the following java packages:

- org.w3c.dom.*, org.xml.sax.SAXException, javax.xml.parsers.*, javax.xml.parsers.*, java.io.*, javax.xml.transform.*, javax.xml.transform.dom.DOMSource, javax.xml.transform.stream.* for working with the xml files;
- java.util.StringTokenizer and java.util.Vector for the parsing of the text;
- java.awt.*, java.awt.event.*, javax.swing.* for constructing and working with the graphical interface;

V. Conclusions and future work

In this paper an architecture of a question answering system is proposed. The inference engine and the knowledge base are built taking into account a knowledge representation method named conditional schema. The implementation uses the concepts of XML-conditional structure generated by a conditional schema and XML-conditional knowledge base introduced in this paper.

The user can interrogate the system only by sentences concerning the declarative facts. An interesting extension of our study refers to the case when the user asks the system in the area of procedural knowledge. In this way we are led to the idea to add a way to explain how the system obtained the conclusion. This can be performed using a *Module of Explanations*. By endowing the system with this feature, we will obtain an improved system that can be used in automatic training. In a future work we will exemplify the use of such a system in automatic training, to build health systems consultancy and systems for diagnosing the malfunctions of a device. In a possible future version of the architecture presented in Figure 1 the Administrator module and its Graphical Interface can be removed such that all the rules and the declarative knowledge to be given by the user. A special module of the system will extract the rules from the user spoken message and will formalize these entities by means of conditional schema representations.

The knowledge representation method presented in this paper can be used to build dialogue systems based on conditional schemas. This is a possible future task.

REFERENCES

[1] **E.Agichtei, L.Gravano**:- Snowball: Extracting Relations from Large Plain-Text Collections. Proceeding of the 5th ACM International Conference on Digital Libraries (DL'00). San Antonio, TX. June 2-7, 2000.

[2] **E.Breck, J.Burger, L.Ferro, D.House, M.Light, Inderjeet Mani**:- A sys called Qanda, Eighth Text REtrieval Conference (TREC-8), Gaithersburg, MD. November 17-19, 1999

[3] **E.Breck, J.Burger, L.Ferro, W.Greiff, M.Light, I.Mani, J.Rennie**:- Another sys called Qanda, Ninth Text REtrieval Conference(TREC-9), Gaithersburg, MD. November 13-16, 2000

[4] **Eric Brill, Susan Dumais, Michele Banko** :- An analysis of the AskMSR question-answering system, Proceedings of the ACL-02 conference on Empirical methods in natural language processing - Volume 10, p.257-264, 2002

[5] **A.Fujii, T.Ishikawa**:- Organizing Encyclopedic Knowledge based on the Web and its Application to Question Answering, Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics (ACL-EACL 2001), Toulouse, France, July 6-11, 2001.

[6] **S.Harabagiu, D.Moldovan, C.Clark, M.Bowden, A.Hickl, P.Wang**:- Employing Two Question Answering Systems in TREC-2005, In: Text Retrieval Conference (TREC-14),Gaithersburg, Maryland (2005)

[7] **D.Moldovan, M.Bowden, M.Tatu**: A Temporally-Enhanced PowerAnswer in TREC 2006, In: Text Retrieval Conference (TREC-15), Gaithersburg, Maryland (2006)

[8] **D.I.Moldovan, C.Clark, S.M.Harabagiu, D.Hodges**:- COGEX: A semantically and contextually enriched logic prover for question answering. J. Applied Logic 5(1): 49-69, 2007

[9] **Harksoo Kim, Kyungsun Kim, Gary Geunbae Lee, Jungyun Seo**:- MAYA: a fast Question-answering system based on a predictive answer indexer, Proceedings of the workshop on Open-domain question answering - Volume 12, Toulouse, France, p.1 - 8, 2001

[10] **B.Katz**:- Using English for Indexing and Retrieving, in Artificial Intelligence at MIT: Expanding Frontiers, v. 1, Cambridge, Massachusetts, 1990.

[11] **N.Ţăndăreanu, M.Colhon**:- Conditional graphs generated by conditional schemas, Annals of the University of Craiova, Mathematics and Computer Science Series, Vol.36, No.1, p.1-11, 2009

[12] **M.Colhon, N.Ţăndăreanu**:- The Inference Mechanism in Conditional Schemas, Annals of the University of Craiova, Mathematics and Computer Science Series, Vol.37, No.1, p.55-57, 2010

[13] **Zhiping Zheng**:- AnswerBus Question Answering System. Proceeding of HLT Human Language Technology Conference (HLT 2002). San Diego, CA. March 24 - 27, 2002

[14] **Mahsa A. Yarmohammadi, Mehrnoush Shamsfard, Mahshid A. Yarmohammadi, Masoud Rouhizadeh** :- SBUQA Question Answering System, Advances in Computer Science and Engineering, 13th International CSI Computer Conference, CSICC 2008 Kish Island, Iran, March 9-11, 2008 Revised Selected Papers, Springer Berlin Heidelberg, 316-323, 2009