Annals of the University of Craiova, Mathematics and Computer Science Series
Volume 37(1), 2010, Pages 55–70
ISSN: 1223-6934

Inference Mechanism in Conditional Schemas

Mihaela Colhon and Nicolae Ţăndăreanu

Abstract. 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 knowledge piece applied on their inputs. Such inputs are called *conditional knowledge* ([14]). By the name of *conditional schema* we understand a graph based structure that can represent conditional knowledge ([14]). Such a structure does not include proper rules as in the case of the rule-based systems. The rules are extracted from conditional knowledge. In this paper we formalize the inference in conditional schemas.

2010 Mathematics Subject Classification. Primary 68T30; Secondary 68T50.
Key words and phrases. conditional knowledge, rule-based reasoning, partial algebra, morphism of partial algebras, inference.

1. Introduction

Various aspects related to the concept of conditional knowledge can be relieved in the literature. This concept can involve a probabilistic aspect. In [16] a Bayesian higher-order probability logic reasoning approach with interval probability parameters to the problem of making inference from conditional knowledge is proposed. A logical approach to non monotonic reasoning based on the notion of a non monotonic consequence relation is proposed in [8]. A conditional knowledge concept intended to help the reasoning process about distributed simulations is given in [3]. Conditional knowledge bases have been proposed as belief bases that include defaults rules of the form "*generally, if* α *then* β". The complexity of default reasoning from conditional knowledge bases is treated in [5]. The concepts of nested conditional knowledge bases and co-nested conditional knowledge bases were introduced in [6]. The tractable classes presented in [6] can be recognized in linear time. Possibilistic and probabilistic semantics of conditional knowledge bases are analyzed in [1]. The inference with conditional knowledge bases is also investigated in [2].

The concept of conditional knowledge is connected also by rule based systems. Today, most of the successful knowledge-based systems are such systems. These are computer programs that operate using a set of IF-THEN rules (production rules). The knowledge in the rule-based systems or production systems is often classified as follows ([12]):

- *declarative* - propositions or facts describing the current state of the problem (e.g. facts that are known to be true)
- *procedural* - the logical rules: IF *condition* THEN *action*

A comprehensive collection of state-of-the-art advancements in rule languages, containing methodologies for building rule-based applications can be found in [7].

Received December 20, 2009. Revision received February 08, 2010.

There are two ways to work with rules of the form IF-THEN:

- Many systems contain their proper set of rules and declarative knowledge are applied on these rules. This is the classic way. The basic control architecture of such rule-based system consists of two modules called "recognition" and "action" ([4]). The recognition module involves selecting a single rule for execution which is further passed to the selection and conflict resolution module. Two strategies were developed for these systems: *forward chaining* also known as data-driven reasoning (LISP70, DENDRAL) and *backward chaining* or goal-driven reasoning (MYCIN).
- The second way is to describe the rules in input texts and thus, the control architecture of such systems is very dynamic as it changes for each input. The concept of conditional schema introduced in [14] belongs to this category.

In [13] we introduced the concept of conditional relation, the algebra of these relations is studied and an intuitive connection with knowledge representation is presented. The concepts of *conditional schema* and *conditional graph* were defined in [14]. We proved that every conditional schema generates one and only one conditional graph.

In this paper we describe the inference mechanism in a conditional schema. As we proved in [14], a conditional schema generates one and only one conditional graph and thus the concepts of graph theory can be successfully used. The inference process of conditional schema is defined by means of the conditional graph structure. It uses the rules extracted from inputs in order to assign values for some "conditional mappings". In this manner, the rules can be considered as "semaphores" that allow to use some parts of the paths in the corresponding conditional graph.

The advantages of using a graph-based structure in the representation and reasoning mechanism consist in the fact that we can model complex flow and loop patterns, that cannot be directly translated into block structures ([9]).

The paper is organized as follows: in Section 2 we recall the main concepts and results treated in [14], which are used by the subsequent sections; in Section 3 we formalize the inference in a conditional schema; Section 4 defines the answer function of a system based on conditional schemas; In Section 5 all the concepts are exemplified and the computations are illustrated; the last section includes the conclusions and future work. As we mentioned in Section 6 the model of knowledge representation proposed in this paper lies at the confluence of mathematical logic, rule-based systems and graph theory.

2. Previous concepts and results

The conditional schema proposed in [14] is designed to represent declarative and procedural knowledge given in a natural language. The set of objects from such texts together with their properties and the relations existing between objects form the *declarative knowledge* of the schema. The set of sentences of the form if-then defines the context of *procedural knowledge*. In what follows we will suppose that the relations between objects are binary ones. A relation is obtained by extracting from the text all the ordered pairs of objects satisfying the same property.

If we note by Ob the set of objects then, each binary relation σ extracted from the text is a subset $\sigma \subseteq Ob \times Ob$. In [13] two types of binary relations are considered:

- *classical binary relations* which are unconditionally true, eg. "*Bob is a fish*" represented by:

$$is\_a = \{(Bob, fish), T(Bob)\}$$

where T stands for *true*.

- *conditional binary relations* that are true under some conditions; for example, the sentence "*If Bob lives in a fishbowl then it is a fish*" has the representation:

$$is\_a = \{(Bob, fish), p(Bob)\}$$

where $p(x)$ has the meaning "*x lives in a fishbowl*"

Conditional mappings like p are defined in terms of declarative knowledge ([13]). If C_s denotes the set of these mappings then:

- each $t \in C_s$ is defined as $t : Ob \to \{true, false\}$
- if $t = T \in C_s$ then $T(n) = true$, for every $n \in Ob$.

Intuitively, for each conditional mapping $t \in C_s$ and for an arbitrary object n we say that "*t is on*" for n if the condition attached to t is satisfied for n and "*t is off*" otherwise.

A conditional schema is a tuple $\mathcal{S} = (Ob, C_s, E_r, A, V, B_{cr}, h, f)$ such that ([14]):

- $Ob = Ob_{ind} \cup Ob_{abstr}$, 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 attributes for the elements of Ob_{ind} while V is the set of their values;
- $B_{cr} \subseteq 2^{((Ob \times I) \times (Ob \times I)) \times C_s}$ 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}.

A pair $(attr, val) \in f(x)$, $x \in Ob_{ind}$, specifies the value val of the attribute $attr$ for the object x. The set of all pairs

$$\{(attr, val) \in A \times V \mid (attr, val) \in f(x)\}$$

can be considered as the set of slots for the object $x \in Ob_{ind}$ (as it is defined in [11]). In what concerns the conditional mappings, we use the notations:

$$((n, \omega_1), (m, \omega_2)) \in_c h(r) \tag{1}$$

$$p(n) = Cond_r((n, \omega_1), (m, \omega_2)) \tag{2}$$

for the fact that $((n, \omega_1), (m, \omega_2), p(n))$ belongs to the conditional relation $h(r)$.

As an example we consider the following knowledge piece: *Helen and George are students. Every student plays basketball if the student is tall.*

We consider $E_r = \{r_1, r_2\}$ and $C_s = \{p\}$. From the case presented above we can take:

$$h(r_1) = \{(((Helen, i), (student, a)), T(Helen)), \\ (((George, i), (student, a)), T(George))\}$$

$$h(r_2) = \{(((student, a), (basketball, a)), p(student))\}$$

where $Ob = \{Helen, George, student, basketball\}$ and $p : Ob \longrightarrow \{true, false\}$ is defined as follows: $p(n) = true$ if n *is tall* and $p(n) = false$ otherwise.

For two arbitrary mappings $p, q : Ob \longrightarrow \{true, false\}$ we define the mapping $p \wedge q : Ob \longrightarrow \{true, false\}$ by $(p \wedge q)(n) = true$ if and only if $p(n) = true$ and $q(n) = true$.

FIGURE 1. Graphical representation of a node

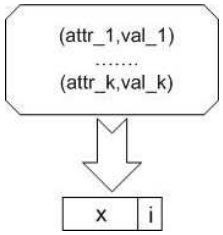

FIGURE 2. Additional information for individual node

The components of a conditional schema can be uniquely represented in a particular directed labeled graph named *conditional graph* ([14]).

Let $\mathcal{S} = (Ob,\ C_s,\ E_r,\ A,\ V,\ B_{cr},\ h,\ f)$ be a conditional schema. The conditional graph ([14]) 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 $((n, \omega_1), (m, \omega_2)) \in_c h(r)$; 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.*

The graphical representation of a conditional graph $G_{\mathcal{S}} = (X \cup Z, \Gamma_X \cup \Gamma_Z)$ is obtained as follows:

- Each element of X is represented by a rectangle as in Figure 1;
- We draw an arc from the node $x \in X$ to $y \in X$ and we put the label $r \in E_r$ on this arc if and only if $(x, r, y) \in \Gamma_X$;
- For each individual node $x \in X$ we append the additional information $f(x)$. This information is collected in a rounded rectangle linked by x as in Figure 2.

3. The inference mechanism

The inference mechanism in a conditional schema is based on the paths of the corresponding conditional graph. In what follows we formalize this path-based inference mechanism.

Definition 3.1. *Let $G_{\mathcal{S}} = (X \cup Z, \Gamma_X \cup \Gamma_Z)$ be the conditional graph generated by the conditional schema $\mathcal{S} = (Ob,\ C_s,\ E_r,\ A,\ V,\ B_{cr},\ h,\ f)$. Let $n_1,\ n_{k+1} \in Ob$ be two arbitrary objects. A path from n_1 to n_{k+1} in $G_{\mathcal{S}}$ is a pair of the form:*

$$([(n_1, \omega_1), \ldots, (n_{k+1}, \omega_{k+1})], [r_1, \ldots, r_k])$$

such that the following conditions are fulfilled:

- $(n_j, \omega_j) \in X,\ j \in \{1, \ldots, k+1\}$
- *for each* $j \in \{1, \ldots, k\}$ *we have* $r_j \in E_r$
- $((n_j, \omega_j), r_j, (n_{j+1}, \omega_{j+1})) \in \Gamma_X,\ j \in \{1, \ldots, k\}$

We denote by $Path(n_1, n_{k+1})$ the set of all paths from n_1 to n_{k+1} in $G_{\mathcal{S}}$.

Remark 3.1.
(1) *Because* $Ob_{ind} \cap Ob_{abstr} = \emptyset$, $pr_1 X = Ob$ *and* $Ob = Ob_{ind} \cup Ob_{abstr}$, *there is no element* $n \in Ob$ *such that* $(n, i) \in X$ *and* $(n, a) \in X$. *It follows that in Definition 3.1 the elements* $n_1, n_{k+1} \in Ob$ *determine uniquely the elements* $(n_1, w_1) \in X$ *and* $(n_{k+1}, w_{k+1}) \in X$.
(2) *A path in* $G_{\mathcal{S}}$ *can not contain arcs of the second category.*

If we denote by $D = 2^{Ob \times Ob}$ the set of all classical binary relations over D then we define the mapping $\mu_0 : E_r \longrightarrow D$ as follows:

$$\mu_0(r) = \{(n, m) \in Ob \times Ob \mid \exists \omega_1, \omega_2 \in I : ((n, \omega_1), r, (m, \omega_2)) \in \Gamma_X\}$$

As in the theory of partial algebras, we say that a mapping g is an *extension* of the mapping f and we denote this property by $f \prec g$ if the following conditions are satisfied:
$f : dom(f) \longrightarrow A$
$g : dom(g) \longrightarrow A$
$dom(f) \subseteq dom(g)$
$g(x) = f(x)$ for all $x \in dom(f)$.
We consider a superset E_r^* of E_r, that is:

$$E_r \subseteq E_r^* \tag{3}$$

and an extension of the mapping μ_0:

$$\mu : E_r^* \longrightarrow D \tag{4}$$

We consider also a partial binary operation:

$$\varphi : E_r^* \times E_r^* \longrightarrow E_r^* \tag{5}$$

that satisfies the following property:

$$\mu(\varphi(e_1, e_2)) = \mu(e_1) \circ \mu(e_2) \tag{6}$$

where $\circ$ is the product operation on D. It is understood that the equality (6) holds for every $(e_1, e_2) \in E_r^* \times E_r^*$ such that $\varphi(e_1, e_2)$ is defined. We can restate the previous conditions as in the next proposition.

Proposition 3.1. *The pairs* (E_r^*, φ) *and* $(D, \circ)$ *are partial algebras and* μ *is a morphism of partial algebras. In other words, for* $(u, v) \in dom(\rho)$ *then*

$$\mu(\varphi(u, v)) = \mu(u) \circ \mu(v) \tag{7}$$

Many times in the theory of partial algebras the properties are stated in the language of diagrams. In this way an intuitive image of the property is obtained. In this language, Proposition 3.1 can be restated as in the next proposition.

Proposition 3.2. *If we can go along the path* $(E_r^* \times E_r^*, E_r^*, D)$ *in the diagram from Figure 3 then we can go also along the path* $(E_r^* \times E_r^*, D \times D, D)$ *and we obtain the same result.*

Definition 3.2. *We denote by* $List(E_r^*)$ *the set of all nonempty lists containing elements from* E_r^*. *We define the partial mapping* $\Phi : List(E_r^*) \longrightarrow E_r^*$ *as follows:*
(1) $\Phi([a_1]) = a_1$
(2) *For* $k \geq 2$ *we take* $\Phi([a_1, \ldots, a_k]) = b_k$, *where* b_k *is obtained by the following procedure:*

$$b_1 = a_1,\ b_2 = \varphi(b_1, a_2),\ \ldots,\ b_k = \varphi(b_{k-1}, a_k)$$

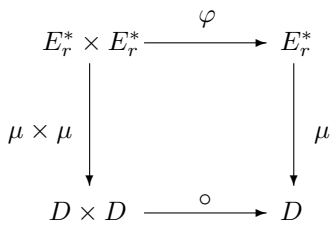

FIGURE 3. The diagram for μ and φ

Proposition 3.3. *If* $[a_1, \ldots, a_k] \in dom(\Phi)$ *and* $(\Phi([a_1, \ldots, a_k]), a_{k+1}) \in dom(\varphi)$ *then* $[a_1, \ldots, a_{k+1}] \in dom(\Phi)$ *and* $\Phi([a_1, \ldots, a_{k+1}]) = \varphi(\Phi([a_1, \ldots, a_k]), a_{k+1})$

Proof: Suppose that $[a_1, \ldots, a_k] \in dom(\Phi)$. It follows that $\Phi([a_1, \ldots, a_k]) = b_k$, where b_k is obtained from the sequence $b_1 = a_1$, $b_2 = \varphi(b_1, a_2)$, ..., $b_k = \varphi(b_{k-1}, a_k)$. If $(\Phi([a_1, \ldots, a_k]), a_{k+1}) \in dom(\varphi)$ then $(b_k, a_{k+1}) \in dom(\varphi)$. If we denote $b_{k+1} = \varphi(b_k, a_{k+1})$ then $[a_1, \ldots, a_{k+1}] \in dom(\Phi)$ and $\Phi([a_1, \ldots, a_{k+1}]) = b_{k+1}$.

Proposition 3.4. *If* $[a_1, \ldots, a_{k+1}] \in dom(\Phi)$ *then* $[a_1, \ldots, a_k] \in dom(\Phi)$ *and* $\mu(\Phi([a_1, \ldots, a_{k+1}])) = \mu(\Phi([a_1, \ldots, a_k]) \circ \mu(a_{k+1})$

Proof: Applying Proposition 3.3 we obtain:

$$\mu(\Phi([a_1, \ldots, a_{k+1}])) = \mu(\varphi(\Phi([a_1, \ldots, a_k]), a_{k+1}))$$

and further, from Proposition 3.1 we obtain

$$\mu(\varphi(\Phi([a_1, \ldots, a_k]), a_{k+1})) = \mu(\Phi([a_1, \ldots, a_k])) \circ \mu(a_{k+1})$$

and the proposition is proved.

Proposition 3.5. *Let us consider* $d = ([(n_1, \omega_1), \ldots, (n_{k+1}, \omega_{k+1})], [a_1, \ldots, a_k])$ *be a path in the conditional graph* $G_{\mathcal{S}}$*, where* $k \geq 1$*. If* $[a_1, \ldots, a_k] \in dom(\Phi)$ *then:*

$$(n_1, n_{k+1}) \in \mu(\Phi([a_1, \ldots, a_k])) \tag{8}$$

Proof: We proceed by induction on k. Let us verify the proposition for $k = 1$. In this case we have the path

$$d = ([(n_1, \omega_1), (n_2, \omega_2)], [a_1])$$

and from Definition 3.1 we deduce that:

$$((n_1, \omega_1), a_1, (n_2, \omega_2)) \in \Gamma_X$$

Taking into consideration the definition of μ_0 we obtain:

$$(n_1, n_2) \in \mu_0(a_1)$$

But $\mu(a_1) = \mu_0(a_1)$ because $a_1 \in E_r^*$ and $\Phi([a_1]) = a_1$. Thus (8) is verified for $k = 1$. Suppose the relation (8) is verified for k. Consider a path:

$$d = ([(n_1, \omega_1), \ldots, (n_{k+2}, \omega_{k+2})], [a_1, \ldots, a_{k+1}])$$

such that $[a_1, \ldots, a_{k+1}] \in dom(\Phi)$. By the inductive assumption we have:

$$(n_1, n_k) \in \mu(\Phi([a_1, \ldots, a_k])) \tag{9}$$

But $d_1 \in Path(n_1, n_{k+2})$ therefore from Definition 3.1 we deduce that:

$$((n_j, \omega_j), a_j, (n_{j+1}, \omega_{j+1})) \in \Gamma_X$$

for $j \in \{1, \ldots, k+1\}$. Particularly, we have:

$$((n_{k+1}, \omega_{k+1}), a_{k+1}, (n_{k+2}, \omega_{k+2})) \in \Gamma_X$$

Applying the definition of $\mu_0(a_{k+1})$ we obtain:

$$(n_{k+1}, n_{k+2}) \in \mu_0(a_{k+1}) \tag{10}$$

Combining (9) and (10) we obtain:

$$(n_1, n_{k+2}) \in \mu(\Phi([a_1, \ldots, a_k])) \circ \mu(a_{k+1})$$

Now, from Proposition 3.4 we obtain

$$(n_1, n_{k+2}) \in \mu(\Phi([a_1, \ldots, a_{k+1}]))$$

and the proposition is proved.

4. The answer mapping

The answers of the conditional schema to the received interrogations are sentences in natural language (English). If we denote by G the grammar used to generate natural language constructions of the system then by $L(G)$ we note the language generated by the grammar rules.

An answer is constructed by means of the following function:

$$Sem : Ob \times E_r^* \times Ob \times \{on, off\} \longrightarrow L(G) \tag{11}$$

such as:

∘ $Sem(x, a, y, \text{on})$ specifies the semantics of the relation given by $\mu(a)$ existing between the objects x and y;

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

For example, if

$$Sem(Peter, is\_a, student, \text{on}) = Peter\ is\ a\ student$$

then

$$Sem(Peter, is\_a, student, \text{off}) = Peter\ is\ not\ a\ student$$

The values of the conditional mappings are expressed in terms of declarative knowledge about the individual objects. Thus, the interrogations and the answers are defined in the system with respect to the individual objects $x \in Ob_{ind}$, represented in the conditional graph by pairs $(x, i) \in X$.

Remark 4.1. *Based on the previous results and using the notation (2) we observe that if* $d = ([(n_1, \omega_1), \ldots, (n_{k+1}, \omega_{k+1})], [a_1, \ldots, a_k]) \in Path(n_1, n_{k+1})$ *then, there are* $t_1, \ldots, t_k \in C_s$ *such that:*

$$t_j(n_j) = Cond_{a_j}((n_j, \omega_j), (n_{j+1}, \omega_{j+1})) \tag{12}$$

In what follows, we will note by $CS(d)$ *the list of all conditional mappings of the path* d*, that is,* $CS(d) = [t_1, \ldots, t_k]$.

Remark 4.2. *As the values for the conditional mappings act like semaphores in the reasoning process of the conditional schema, in what follows we will use the value "on" instead of "true" and "off" instead of "false".*

Definition 4.1. *Let* $d = ([(n_1, w_1), \ldots, (n_{k+1}, w_{k+1})], [r_1, \ldots, r_k])$ *be a path in* $G_{\mathcal{S}}$. *We define:*

- *Suppose* $(n_{i_2}, a) \in X$*; the node* $(n_{i_1}, i) \in X$ *is the* **nearest individual object** *of* n_{i_2}*, where* $1 \leq i_1 < i_2 \leq k+1$ *if there is no* s *such that* $i_1 < s < i_2$ *and* (n_s, i) *is in* d*. In this case we denote* $Near(n_{i_2}) = n_{i_1}$*. By extension, if* $(n_{i_2}, i) \in X$ *then we define* $Near(n_{i_2}) = n_{i_2}$*.*
- *Suppose that* $CS(d) = [t_1, \ldots, t_k]$*. For* $1 \leq j \leq k$ *we write* $t_j[d] = on$ *if and only if either* $t_j = T$ *or* $t_j(x) = on$ *for* $x = Near(n_j)$*.*

For an individual object x if $(attr, value)$ represents an initial knowledge about it, then we shall denote:

$$V_x(attr) = value$$

in order to specify that *value* is referred to x for the property *attr*. For example the sentence *The general score of Peter is 9.50* can be represented by $V_{Peter}(gen\_score) = 9.50$.

The conditions attached to each symbol of C_s can be transposed in IF-THEN-ELSE rules. For example the following rules set values *on* and *off* for each element of the set $C_s = \{p_1, p_2, p_3, p_4\}$:

$R_1(x)$: IF $V_x(gen\_score) > 9$ THEN $p(x) =$ on ELSE $p_1(x) =$ off
$R_2(x)$: IF $8 < (V_x(score1) + V_x(score2))/2 \leq 9$ THEN $p_2(x) =$ on ELSE $p_2(x) =$ off
$R_3(x)$: IF $(V_x(score1) + V_x(score2))/2 > 9$ THEN $p_3(x) =$ on ELSE $p_3(x) =$ off
$R_4(x)$: IF $V_x(height) = tall$ THEN $p_4(x) =$ on ELSE $p_4(x) =$ off

where R_j is the name of the rule and $R_j(x)$ denotes the fact that the rule R_j is applied for the individual object x, where $j \in \{1, 2, 3, 4\}$.

Definition 4.2. *Let* $\mathcal{S} = (Ob, C_{cs}, E_r, A, V, B_{cr}, h, f)$ *be a conditional schema. A system* $V_{\mathcal{S}} = (\mathcal{S}, E_r^*, L(G), R, \varphi, \mu, Sem)$ *is named a* **valuation system** *for* $\mathcal{S}$ *if the conditions (3)-(6) and (11) are satisfied and* R *is the set of rules that compute the values of the conditional symbols.*

Definition 4.3. *Let us consider* $d = ([(n_1, \omega_1), \ldots, (n_{k+1}, \omega_{k+1})], [a_1, \ldots, a_k]) \in Path(n_1, n_{k+1})$ *such that:*

- *there is* $j \in \{1, \ldots, k+1\}$ *such that* $\omega_j = i$
- $[a_1, \ldots, a_k] \in dom(\Phi)$
- $CS(d) = [t_1, \ldots, t_k]$

We define $ans(d)$ *such that:*

- *If* $t_1[d] = \ldots = t_k[d] = on$ *and*

$$(n_1, \Phi([a_1, \ldots, a_k]), n_{k+1}, on) \in dom(Sem)$$

then

$$ans(d) = Sem(n_1, \Phi([a_1, \ldots, a_k]), n_{k+1}, on)$$

- *If there is* $u \in \{1, \ldots, k\}$ *such that* $t_u[d] =$*off and*

$$(n_1, \Phi([a_1, \ldots, a_k]), n_{k+1}, off) \in dom(Sem)$$

then

$$ans(d) = Sem(n_1, \Phi([a_1, \ldots, a_k]), n_{k+1}, off)$$

- $ans(d) = unknown$ *otherwise.*

We define the interrogations for conditional schemas as pairs of the form

$$(x, y) \in Ob \times Ob \tag{13}$$

and as consequence the answer mapping is constructed as follows.

Definition 4.4. *Let $\mathcal{S} = (Ob, C_s, E_r, A, V, B_{cr}, h, f)$ be a conditional schema and a valuation system $V_{\mathcal{S}}$ for $\mathcal{S}$. The* ***answer mapping*** *generated by the pair $(\mathcal{S}, V_{\mathcal{S}})$ is the mapping*

$$Ans : Ob \times Ob \longrightarrow 2^{L(G)} \tag{14}$$

defined as follows:

(1) *If there is $d \in Path(n_1, n_{k+1})$ such that $ans(d) \neq unknown$ then*

$$Ans(n_1, n_{k+1}) = \bigcup_{d \in Path(n_1, n_{k+1})} \{ans(d) \mid ans(d) \neq unknown\}$$

(2) *Otherwise, $Ans(n_1, n_{k+1}) = unknown$*

The algorithm by means of which we can implement the deduction process of a conditional schema is the following:

Begin algorithm

Input: A knowledge piece given in a natural language, denoted by KP

Step 1: Extract from KP the components of the conditional system $\mathcal{S} = (Ob, C_{cs}, E_r, A, B_{cr}, h, f)$ associated to KP.

Step 2: Define the components of a valuation system $(\mathcal{S}, E_r^*, L(G), R, \varphi, \mu, Sem)$ for $\mathcal{S}$.

Output $Ans(n_1, n_{k+1})$ for every pair (n_1, n_{k+1}).

End of algorithm

5. Example of computations

In order to exemplify the computations, firstly we consider the following knowledge piece KP_1:

"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."

The conditional graph corresponding to KP_1 is illustrated in Figure 4.

Remark 5.1. *In order to simplify the writing, the conditional binary relations will be denoted by $((n, m), p)$ instead of $((n, m), p(n))$ for $p \in C_s$, $n, m \in Ob$, the argument of the conditional mapping being self-evident.*

Remark 5.2. *In the graphical representation of a conditional graph $G_{\mathcal{S}} = (X \cup Z, \Gamma_X \cup \Gamma_Z)$, each arc $((n, w_1), r, (m, w_2)) \in \Gamma_X$ is labeled with (r, p) where $p = Cond_r((n, w_1), (m, w_2)) \in C_s$.*

The conditional schema representations of the information of KP_1 are as follows:

- $Ob = \{Helen, Peter, Susan, George, pizza, cheese\_cake, fruits\}$
- $V_{Helen}(made) = cheese\_cake$, $V_{Helen}(age) = 40$
  $V_{Susan}(bought) = fruits$, $V_{Susan}(age) = 30$
- $C_s = \{p, q\}$, where $p(x)$ stands for the condition *"if mother of x made cheese cake"*, $q(x)$ stands for *"if mother of x bought fruits "*, $x \in Ob$;
- $E_r = \{is\_mother, is\_sister, likes, eats\}$, where
  $h(is\_sister) = \{(((Helen, i), (Susan, i)), T)\}$
  $h(is\_mother) = \{(((Helen, i), (Peter, i)), T), (((Susan, i), (George, i)), T)\}$
  $h(likes) = \{(((Helen, i), (pizza, a)), T)\}$
  $h(eats) = (((Peter, i), (cheese\_cake, a)), p), (((George, i), (fruits, a)), q)\}$

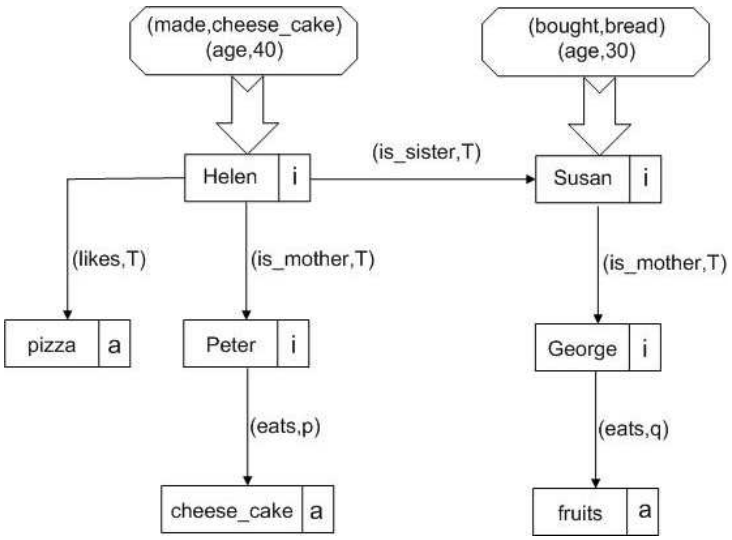

FIGURE 4. Graphical representation of KP_1

$\mu_0(is\_sister) = \{(Helen, Susan)\}$
$\mu_0(is\_mother) = \{(Helen, Peter), (Susan, George)\}$
$\mu_0(likes) = \{(Helen, pizza)\}$
$\mu_0(eats) = \{(Peter, cheese\_cake), (George, fruits)\}$

- In order to define the values of the conditional symbols p and q we can use the following rules, which give the rules set R:
  $R_1(x)$: IF $(y, x) \in \mu_0(is\_mother) \wedge V_y(made) = cheese\_cake$ THEN $p(x) =$ on ELSE $p(x) =$ off
  $R_2(x)$: IF $(y, x) \in \mu_0(is\_mother) \wedge V_y(bought) = fruits$ THEN $q(x) =$ on ELSE $q(x) =$ off
- $A = \{made, bought, age\}$
- The mapping φ is defined as follows:
  $\varphi(is\_sister, is\_mother) = b_1$
  $\varphi(b_1, eats) = b_2$
  $\varphi(is\_mother, eats) = b_3$
- $E_r^* = E_r \cup \{b_1, b_2, b_3\}$
- The extension μ of μ_0 is obtained by computation as follows:
  $\mu(b_1) = \mu(\varphi(is\_sister, is\_mother)) = \mu(is\_sister) \circ \mu(is\_mother) = \mu_0(is\_sister) \circ \mu_0(is\_mother) = \{(Helen, George)\}$
  $\mu(b_2) = \mu(\varphi(b_1, eats)) = \mu(b_1) \circ \mu(eats) = \mu(b_1) \circ \mu_0(eats) = \{(Helen, fruits)\}$
  $\mu(b_3) = \mu(\varphi(is\_mother), eats)) = \mu(is\_mother)) \circ \mu(eats) = \mu_0(is\_mother)) \circ \mu_0(eats) = \{(Helen, cheese\_cake)\}$
- We define now the mapping Sem as follows:
  $Sem(x, eats, y, \text{on}) =$ *"x eats y"*,
  $Sem(x, eats, y, \text{off}) =$ *"x does not eat y"*,
  $Sem(x, is\_mother, y, \text{on}) =$ *"x is the mother of y"*,
  $Sem(x, is\_mother, y, \text{off}) =$ *"x is not the mother of y"*,
  $Sem(x, is\_sister, y, \text{ on}) =$ *"x is the sister of y"*,

$Sem(x, is\_sister, y, \text{off}) =$ *"x is not the sister of y"*,
$Sem(x, likes, y, \text{on}) =$ *"x likes to eat y"*,
$Sem(x, likes, y, \text{off}) =$ *"x does not like to eat y"*,
$Sem(x, b_1, y, \text{on}) =$ *"x is the aunt of y"*,
$Sem(x, b_1, y, \text{off}) =$ *"x is not the aunt of y"*,
$Sem(x, b_2, y, \text{on}) =$ *"The nephew of x eats y"*,
$Sem(x, b_2, y, \text{off}) =$ *"The nephew of x does not eat y"*,
$Sem(x, b_3, y, \text{on}) =$ *"The son of x eats y"*,
$Sem(x, b_3, y, \text{off}) =$ *"The son of x does not eat y"*,

We illustrate the computations of the mapping **Ans** for two cases: the first one will produce an affirmative answer and the second one will produce a negative answer.

(1) In order to compute **Ans(Helen, cheese_cake)** we have to find all the paths from $Helen$ to $cheese\_cake$, that is to calculate $Path(Helen, cheese\_cake)$. There is only one element in this set, namely

$$d_1 = ([(Helen, i), (Peter, i), (cheese\_cake, a)], [is\_mother, eats])$$

Based on d_1 we construct the sequence (a_1, a_2) where:
- $a_1 = is\_mother$
- $a_2 = eats$
- $\Phi([a_1, a_2]) = \varphi(is\_mother, eats) = b_3$

Thus we have $\Phi([a_1, a_2]) = b_3$ and $d_1 = ([(Helen, i), (Peter, i), (cheese\_cake, a)], [a_1, a_2])$.
For the path d_1 we observe that
$t_1(Helen) = Cond_{a_1}((Helen, i), (Peter, i)) = T(Helen) = \text{on}$
$t_2(Peter) = Cond_{a_2}((Peter, i), (cheese\_cake, a)) = p(Peter)$
therefore $CS(d_1) = [T, p]$.
In order to compute

$$ans(d_1) = Sem(Helen, b_3, cheese\_cake, \text{on}) \tag{15}$$

we have to verify that $p[d_1] = on$. This condition is satisfied for $p(Peter) = on$. Applying the rule R_1 we have to evaluate

$$R_1(Peter) : IF\ (y, Peter) \in \mu_0(is\_mother) \wedge V_y(made) = cheese\_cake$$
$$THEN\ p(Peter) = \text{on}\ ELSE\ p(Peter) = \text{off}$$

But $\mu_0(is\_mother) = \{(Helen, Peter), (Susan, George)\}$ therefore we have to take $y = Helen$ and to verify whether or not $V_{Helen}(made) = cheese\_cake$. This is a true condition and thus $p(Peter) = \text{on}$. By our notation we can write that $p[d_1] = \text{on}$.
Using now the mapping Sem we obtain

$ans(d_1)$="*The son of Helen eats cheese cake*"

and $Ans(Helen, cheese\_cake) = \{ans(d_1)\}$

(2) In order to compute **Ans(Helen, fruits)** we observe that:

$$Path(Helen, fruits) = \{d_2\}$$

where:

$$d_2 = ([(Helen, i), (Susan, i), (George, i), (fruits, a)], [is\_sister, is\_mother, eats])$$

and $CS(d_2) = [T, T, q]$ because

$$(((Helen, i), (Susan, i)), T) \in h(is\_sister)$$

$(((Susan, i), (George, i)), T) \in h(is\_mother)$
$(((George, i), (fruits, a)), q(George)) \in h(eats)$

We take the sequence a_1, a_2 and a_3 of labels specified in d_2:
$a_1 = is\_sister$
$a_2 = is\_mother$
$a_3 = eats$

and we compute $\Phi([a_1, a_2, a_3])$:
$a_1 = is\_sister$
$b_1 = \varphi(a_1, is\_mother)$
$b_2 = \varphi(b_1, eats)$

therefore $\Phi([a_1, a_2, a_3]) = b_2$. Now for the path d_2 we observe that:
$t_1(Helen) = Cond_{a_1}((Helen, i), (Susan, i)) = T(Helen) = on$
$t_2(Susan) = Cond_{a_2}((Susan, i), (George, i)) = T(Susan) = on$
$t_3(George) = Cond_{a_3}((George, i), (eats, a)) = q(George)$

therefore $CS(d_2) = [T, T, q]$.

In order to compute $ans(d_2)$ we have to verify whether or not $q[d_2] = on$. This condition is satisfied if $q(George) = on$. Applying the rule R_2 we obtain:

$$R_2(George) : IF\ (y, George) \in \mu_0(is\_mother) \wedge V_y(bought) = fruits$$
$$THEN\ q(George) = on\ ELSE\ q(George) = off$$

But $\mu_0(is\_mother) = \{(Helen, Peter), (Susan, George)\}$ therefore we have to take $y = Susan$. We have $V_{Susan}(bought) = bread$, therefore $q(George) = off$. By our notation we can write that $q[d_2] = off$. We have the second case of the Definition 4.3, therefore we have $ans(d_2) = Sem(Helen, b_2, fruits, off)$= *"The son of Helen does not eat fruits"*. It follows that $Ans(Helen, fruits) = \{ans(d_2)\}$.

Remark 5.3. *The case previously exemplified does not use the entity* ***Near*** *introduced in Definition 4.1. In fact the case presented is a particular one: the predecessor of an abstract node is an individual one. More precisely, if* $((n_1, \omega_1), r, (n_2, a)) \in \Gamma_X$ *then* $\omega_1 = i$ *and thus* $Near(n_2) = n_1$.

In order to relieve the use of the entity $Near$ we exemplify this case by considering the following knowledge piece KP_2:
Peter is a student. He is a boy. A student is a competitor if has an initial score greater that 15. The initial score of Peter is 18. Peter is tall. A competitor plays tennis if is female and plays basketball if is tall.
In order to represent this piece we use the conditional graph from Figure 5.

We obtain:

- $Ob = \{Peter, student, competitor, tennis, basketball\}$
- $V_{Peter}(initial\_score) = 18$
- $C_s = \{p_1, p_2, p_3\}$, where $p_1(x)$ stands for the condition *"if initial score of x is greater than 15"*, $p_2(x)$ stands for *"if x is female "* and $p_3(x)$ for *"if x is tall"*;
- $E_r = \{c_1, c_2, c_3\}$, where
  $h(c_1) = \{(((Peter, i), (student, a)), T)\}$
  $h(c_2) = \{((student, a)(competitor, a)), p_1\}$
  $h(c_3) = \{(((competitor, a), (tennis, a)), p_2),$
  $(((competitor, a), (basketball, a)), p_3)\}$
- We use the following rules:
  $R_1(x)$: IF $V_x(initial\_score) > 15$ THEN $p_1(x) = on$ ELSE $p_1(x) = off$
  $R_2(x)$:IF $V_x(sex) = female$ THEN $p_2(x) = on$ ELSE $p_2(x) = off$

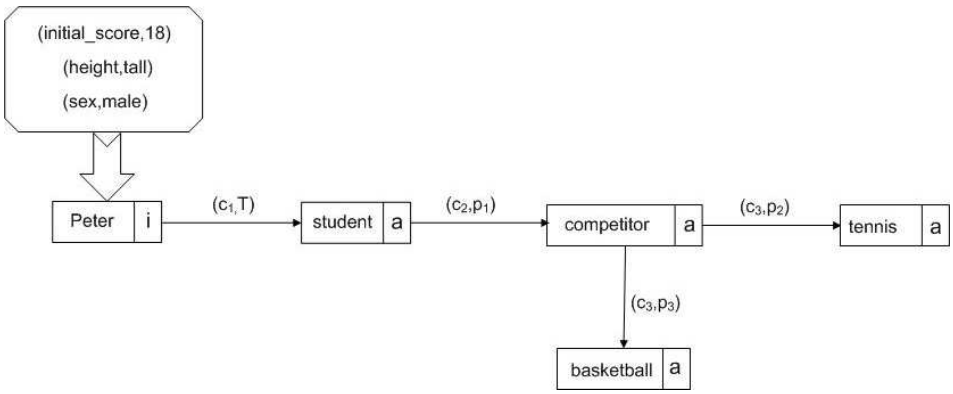

FIGURE 5. Conditional graph for KP_2

$$R_3(x)\text{:IF } V_x(height) = tall \text{ THEN } p_3(x) = on \text{ ELSE } p_3(x) = off$$

- Consider the path

$$d_3 = ([(Peter, i), (student, a), (competitor, a), (basketball, a)], [c_1, c_2, c_3])$$

Obviously we have $CS(d_3) = [T, p_1, p_3]$. In order to establish whether or not $p_1[d_3]$ is on or off we have to valuate $p_1(x)$ for $x = Near(student)$. We observe that $Near(student) = Peter$, therefore we have to compute the value of $p_1(Peter)$. Based on the rule R_1 we obtain $p_1[d_3] = on$. In a similar manner we obtain $Near(competitor) = Peter$ and by R_3 we obtain $p_3[d_3] = on$. If we consider the path

$$d_4 = ([(Peter, i), (student, a), (competitor, a), (tennis, a)], [c_1, c_2, c_3])$$

then we have $Near(student) = Peter$ and $Near(competitor) = Peter$. Taking into account the attribute values of the individual node $Peter$ we obtain $p_1[d_4] = on$ and $p_2[d_4] = off$.

6. Conclusions and future work

The model of knowledge representation proposed in this paper lies at the confluence of logic, rule-based systems and graph theory. A less obvious connection with mathematical logic is related to the individual nodes. It is not difficult to observe that by introducing the attributes for these nodes we intended to model a common reasoning rule of mathematical logic, known as *modus ponens*. In order to exemplify this case we consider a short knowledge piece represented in Figure 6: *Peter is tall. If Peter is tall then he plays basketball.* In mathematical logic, from p and $p \to q$ we deduce q. Based on our inference we deduce *Peter plays basketball.*

The present paper formalizes the inference mechanism of the conditional schemas. This mechanism is a path-driven one. The conditional graph of a conditional schema contains two kinds of nodes: individual nodes and abstract nodes. An individual node enjoys proper values for a set of attributes that characterizes the class of node while the abstract nodes do not have such values.

In order to interrogate such a system, the user specifies two nodes n_1 and n_{k+1}. All the paths d from n_1 to n_{k+1} are computed, where:

$$d = ([(n_1, w_1), \ldots, (n_{k+1}, w_{k+1})], [r_1, \ldots, r_k])$$

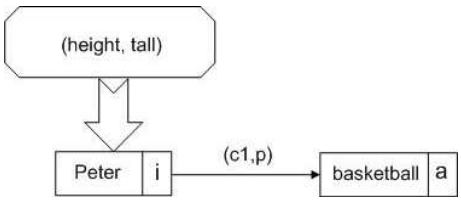

FIGURE 6. A model of modus ponens

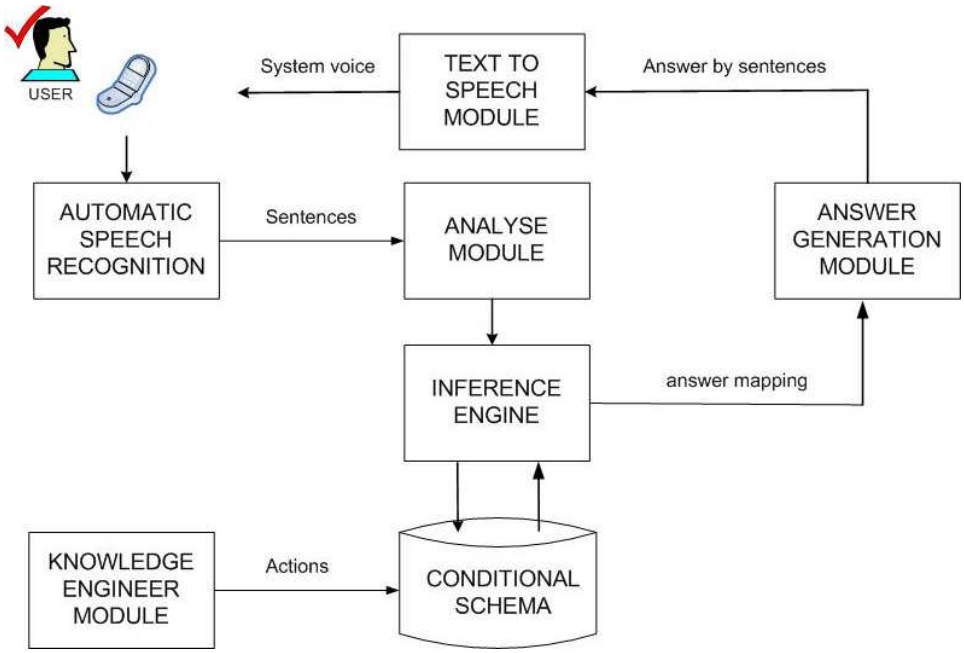

FIGURE 7. An architecture for conditional schema implementation

Each arc $((n_j, w_j), r_j, (n_{j+1}, w_{j+1}))$ of a given path d from n_1 to n_{k+1} contains two kinds of labels:

(1) a label to specify a binary relation between n_j and n_{j+1}, that is r_j, $1 \leq j \leq k$
(2) a label to specify certain condition which is satisfied/not satisfied by the nearest individual object of n_j, that is $Cond_{r_j}((n_j, w_j), (n_{j+1}, w_{j+1}))$.

The node n_1 must be an individual node. The condition from n_j to n_{j+1} can be viewed as a semaphore. The value of a semaphore is computed by means of certain rule, 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 certain conclusion is obtained by the inference mechanism. If a semaphore is "off" then either no conclusion is obtained or a "negative" conclusion is specified. This depends on the semantics mapping Sem of the conditional schema. If d contains only abstract nodes then the inference can not be performed for d. A future research work will include this case.

We intend to implement the inference mechanism of the conditional schema in a spoken dialogue system. To successfully manage the interaction with users, spoken

dialogue systems 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).

A possible architecture for such a dialogue system that uses conditional schema for knowledge representation and reasoning is shown in Figure 7. The communication between user and the system is designed by means of a Voice User Interface through the Speech Recognition Module. The output of such a module, that is a natural language text is passed to the Analyse Module which extracts the semantics of the text.

The system has a second interface, for the knowledge engineer. His task is to keep updated the conditional schema by performing actions like the following ones: introduce the components of a conditional schema, modify these components, erase a component, erase the entire conditional schema etc.

The inference engine is a stand-alone module. This module performs deductions in the system (receives an interrogation and gives back the result of the computation). The answer generation module receives the value of the answer mapping and produces natural language answers which are further passed to the speech synthesis module.

The query-answering GUI can be constructed using graphical controls from AWT toolkit and Swing toolkit while the inference engine can be written using Prolog programming. In order to connect SWI-Prolog reasoning engine with Java Native Interface, Java Prolog Library platform ([10]) is the best choice. A possible idea for further research is to extract the components of a conditional schema from a text description. Such an attempt was made to semantic schemas ([17]).

References

[1] S.Benferhat, D.Dubois and H.Prade, Possibilistic and standard probabilistic semantics of conditional knowledge bases, *Journal of Logic and Computation* **9** (1999), no. 6, 873–895.

[2] S.Benferhat and L.Garcia, A local approach to reasoning with conditional knowledge bases, *Proceedings of Eighth IEEE International Conference on Tools with Artificial Intelligence (ICTAI)* (1996), 404–407.

[3] K.Mani Chandy and J.Misra, Conditional Knowledge as a Basis for Distributed Simulation, Computer Science Department, California Institute of Technology, 5251:TR:87, *Technical Report* (1987).

[4] R.Davis and J.King, The Origin of Rule-Based Systems in AI, Chapter 2 of *Rule-Based Expert Systems: The MYCIN Experiments of the Stanford Heuristic Programming Project*, Edited by Bruce G. Buchanan and Edward H. Shortliffe.

[5] T.Eiter and T.Lukasiewicz, Default reasoning from conditional knowledge bases: Complexity and tractable cases, *Artificial Intelligence* **124** (2000), no. 2, 169–241.

[6] B.B.Garcia, New tractable classes for default reasoning from conditional knowledge bases, *Annals of Mathematics and Artificial Intelligence*, **45** (2005), no. 3-4, 275–291.

[7] A.Giurca, D.Gasevic and K.Taveter, Handbook of Research on Emerging Rule-Based Languages and Technologies: Open Solutions and Approaches, *Information Science Reference*, ISBN: 978-1-60566-402-6, 2009.

[8] D. Lehmann and M. Magidor, What Does a Conditional Knowledge Base Entail?, *Artificial Intelligence* **55** (1989), 1–60.

[9] O. Nicolae, M.Diaconescu and G.Wagner, Simulation Modelling for Businesses using BPMN and AORS, *Annals of University of Craiova, Math. Comp. Sci. Ser.* **36** (2009), no. 2, 69–78.

[10] P. Singleton, F. Dushin and J. Wielemaker, JPL: A bidirectional Prolog/Java interface, `www.swi-prolog.org/packages/jpl`

[11] C.Popîrlan and N.Ţăndăreanu, An Extension of Inheritance Knowledge Bases and Computational Properties of their Answer Functions, *Annals of the University of Craiova, Math. Comp. Sci. Ser.*, **35** (2008), 149–170.

[12] V. A.Thompson, Conditional reasoning: The necessary and sufficient conditions, *Canadian Journal of Experimental Psychology*, **49** (1995), 1–60.
[13] N. Ţăndăreanu and M. Ghindeanu, Towards a Mathematical Modelling of Conditional Knowledge, *Research Notes in Artificial Intelligence and Digital Communications*, **103** (2003), 5–15.
[14] N. Ţăndăreanu and M. Colhon, Conditional graphs generated by conditional schemas, *Annals of University of Craiova, Math. Comp. Sci. Ser.*, **36** (2009), no. 1, 1–11.
[15] J. White and A. Nechypurenko, Intelligence Frameworks for Assisting Modelers in Combinatorically Challenging Domains, *Proceedings of Workshop on Generative Programming and Component Engineering for QoS Provisioning in Distributed Systems* (2006).
[16] Y. Li and W. Liu, Deduction from Conditional Knowledge on Bayesian Networks with Interval Probability Parameters, *International Conference on Computer Science and Software Engineering*, **1** (2008), 594–597.
[17] C.Zamfir, From Text Description to Semantic Schema, *Annals of University of Craiova, Math. Comp. Sci. Ser.*, **36** (2009), no. 2, 97–108.

(Mihaela Colhon and Nicolae Ţăndăreanu) Department of Informatics, University of Craiova,
Al.I. Cuza Street, No. 13, Craiova RO-200585, Romania, Tel. & Fax: 40-251412673
E-mail address: mcolhon@inf.ucv.ro, ntand@rdslink.ro