International Journal of Information, Control and Computer Sciences
ISSN: 2517-9942
Vol:3, No:8, 2009

Computational Networks for Knowledge Representation

Nhon Van Do

***Abstract*—**In the artificial intelligence field, knowledge representation and reasoning are important areas for intelligent systems, especially knowledge base systems and expert systems. Knowledge representation Methods has an important role in designing the systems. There have been many models for knowledge such as semantic networks, conceptual graphs, and neural networks. These models are useful tools to design intelligent systems. However, they are not suitable to represent knowledge in the domains of reality applications. In this paper, new models for knowledge representation called computational networks will be presented. They have been used in designing some knowledge base systems in education for solving problems such as the system that supports studying knowledge and solving analytic geometry problems, the program for studying and solving problems in Plane Geometry, the program for solving problems about alternating current in physics.

***Keywords*—**Artificial intelligence, artificial intelligence and education, knowledge engineering, knowledge representation.

I. Introduction

IN artificial intelligence science, models and methods for knowledge representation play an important role in designing knowledge base systems and expert systems. Nowadays there are many various knowledge models which have already been suggested and applied. In the books [1], [2], [3], and [4] we have found popular methods for knowledge representation in designing knowledge base systems (KBS) such as predicate logic, semantic nets, frames, deductive rules. Many new methods and techniques were presented in [11], [12], [13], and [14]. Among these methods neural networks and fuzzy logic can be used for computational intelligence. Some methods are suitable for representing and processing semantics such as conceptual graphs in [8], [9] and [10].

The above methods are very useful for designing intelligent systems, and for solving complex problems. However, they are not suitable to represent knowledge in the domains of reality applications in many cases, especially the systems that can solve problems in practice based on the knowledge base. The ontology called COKB-ONT presented in [7] is also a good and useful tool for developing knowledge base systems in practice. This ontology was used to construct some intelligent systems in education, and these systems were introduced in [5], [6] and [15]. Although COKB-ONT is very useful and suitable for representing knowledge, it is not strong enough for representing knowledge in the domains of reality applications. Therefore, it is needed to develop new models to represent problems with knowledge. In this paper, we present the models for knowledge representation that are called computational networks. They have been used in designing some knowledge base systems in education for solving problems such as the system that supports studying knowledge and solving analytic geometry problems, the program for studying and solving problems in Plane Geometry, the program for solving problems about alternating current in physics. These applications have been implemented by using programming tools and computer algebra systems such as C++, JAVA, and MAPLE. They are very easy to use for students in studying knowledge, to solve automatically problems and give human readable solutions agree with those written by teachers and students. Problems are also modeled easily using the computational networks, together with the algorithms for solving problems automatically and propose a simple language for specifying them.

II. Computational Networks with Simple Valued Variables

In this part a simple model of computational nets will be presented together related problems and techniques for solving them. Although this model is not very complicated, but it is a very useful tool for designing many knowledge base systems in practice.

A. Definitions

Definition 2.1: A *computational network (CN) with simple valued variables* is a pair (M, F), in which $M = \{x_1, x_2, ..., x_n\}$ is a set of variables with simple values (or unstructured values), and $F = \{f_1, f_2, ..., f_m\}$ is a set of computational relations over the variables in the set M. Each computational relation $f \in F$ has the following form:

(i) An equation over some variables in M, or

(ii) Deductive rule $f : u(f) \rightarrow v(f)$, with $u(f) \subseteq M$, $v(f) \subseteq M$, and there are corresponding formulas to determine (or to compute) variables in v(f) from variables in u(f).We also define the set $M(f) = u(f) \cup v(f)$.

Remark: In many applications equations can be represented as deduction rules.

Example 2.1: The computational knowledge over elements of a triangle named ABC can be represented by a computational networks (M, F) with M = {A, B, C, a, b, c, R, S, p, …} (the set of all attributes of triangle ABC) and

$F = \{ f_1: A + B + C = \pi, \quad f_2 : \frac{a}{\sin(A)} = \frac{b}{\sin(B)},$

$f_3 : \frac{c}{\sin(C)} = \frac{b}{\sin(B)}, f_4: \frac{a}{\sin(A)} = \frac{c}{\sin(C)},$

$f_5 : p = (a+b+c)/2, f_6 : S = a.h_a / 2, f_7 : S = b.h_b / 2,$

$f_8 : S = c.h_c / 2, f_9 : S = a.b.\sin(C) / 2, ...\}.$

B. Problems

Given a computational net (M, F). The popular problem arising from reality applications is that to find a solution to determine a set $H \subseteq M$ from a set $G \subseteq M$. This problem is denoted by the symbol $H \rightarrow G$, H is the hypothesis and G is the goal of the problem. To solve the problem we have to answer two questions below:

Q1: Is the problem solvable based on the knowledge K = (M, F)?

Q2: How to obtain the goal G from the hypothesis H based on the knowledge K = (M, F) in case the problem is solvable?

Example 2.2: In the knowledge K = (M, F) of example 1, suppose that H = { a=5, b=4, A=π/2 }, Find a solution for the goal G = {S, R}.

Definition 2.2: Given a computational net K = (M, F).

(i) For each $A \subseteq M$ and $f \in F$, denote $f(A) = A \cup M(f)$ be the set obtained from A by applying f. Let $S = [f_1, f_2, ..., f_k]$ be a list consisting relations in F, the notation $S(A) = f_k(f_{k-1}(\dots f_2(f_1(A)) \dots))$ is used to denote the set of variables obtained from A by applying relations in S.

(ii) The list $S = [f_1, f_2, ..., f_k]$ is called a *solution* of the problem $H \rightarrow G$ if $S(H) \supseteq G$. Solution S is called a good solution if there is not a proper sublist S' of S such that S' is also a solution of the problem. The problem is *solvable* if there is a solution to solve it.

Definition 2.3: Given a computational net K = (M, F). Let A be a subset of M. It is easy to verify that there exists a unique set $\overline{A} \subseteq M$ such that the problem $A \rightarrow \overline{A}$ is solvable; the set $\overline{A}$ is called the closure of A.

C. Algorithms and Theorems

The following are some algorithms and results that show methods and techniques for solving the above problems on computational nets. The proofs will be omitted here. They were used in designing knowledge base systems such as those presented in [5], [6], [7] and [15].

Theorem 2.1: Given a computational net K = (M, F). The following statements are equivalent.

(i) Problem $H \rightarrow G$ is solvable.

(ii) $\overline{H} \supseteq G$.

(iii) There exists a list of relations S such that $S(H) \supseteq G$.

Algorithm 2.1: Find a solution of the problem $H \rightarrow G$.

```
Step 1: Solution ← empty;
Step 2: if G ⊆ H then
            begin
            Solution_found ← true;
            goto step 4;
            end
        else
            Solution_found ← false;
Step 3: Repeat
            Hold ← H;
            Select f ∈ F;
            while not Solution_found and (f found) do
                begin
                if (applying f from H produces new facts)
                then
                    begin
                    H ← H ∪ M(f);
                    Add f to Solution;
                    end;
                if G ⊆ H then
                    Solution_found ← true;
                Select new f ∈ F;
                end;{ while }
        Until Solution_found or (H = Hold);
Step 4: if not Solution_found then
            There is no solution found;
        else
            Solution is a solution of the problem;
```

Algorithm 2.2: Find a good solution from a solution $S = [f_1, f_2, ..., f_k]$ of the problem $H \rightarrow G$ on computational net (M, F).

```
Step 1: NewS ← [];
        V ← G;
Step 2: for i := k downto 1 do
            If v(f_k) ∩ V ≠ ∅ then
              Begin
              Insert f_k at the beginning of NewS;
              V ← (V – v(f_k)) ∪ (u(f_k) – H);
              End
Step 3: NewS is a good solution.
```

On a computational net (M, F), in many cases the problem $H \rightarrow G$ has a solution S in which there are relations producing some redundancy variables. At those situations, we must determine necessary variables of each step in the problem solving process. The following theorem shows the way to analyse the solution to determine necessary variables to compute at each step.

Theorem 2.2: Given a computational net K = (M, F). Let $[f_1, f_2, ..., f_m]$ be a good solution of the problem $H \rightarrow G$. denote $A_0 = H$, $A_i = [f_1, f_2, ..., f_i](H)$, with i=1, ..., m. Then there exists a list $[B_0, B_1, ..., B_{m-1}, B_m]$ satisfying the following conditions:

(1) $B_m = G$,

(2) $B_i \subseteq A_i$, with i=0, 1, ..., m.

(3) For i=1,...,m, $[f_i]$ is a solution of the problem $B_{i-1} \rightarrow B_i$ but not to be a solution of the problem $B \rightarrow B_i$, with B is any proper subset B of B_{i-1}.

Example 2.3: For the computational net (M, F) representing the knowledge related to triangles in example 2.1, find a solution of the problem $\{a, B, C\} \rightarrow \{S\}$. The algorithm 2.1 will give us a solution $Sol = [f_1, f_2, f_3, f_5, f_9]$, and this solution

is not a good solution because there exists a redundancy relation such as f_5. From the solution Sol the algorithm 2.2 will give the new solution NewSol = [f_1, f_2, f_9], and the process to solve the problem is as follows:

Step 1: Compute A by applying f_1;
Step 2: Compute b by applying f_2;
Step 3: Compute S by applying f_9;

III. NETWORKS OF COMPUTATIONAL OBJECTS

In many problems we usually meet many different kinds of objects. Each object has attributes and internal relations between them. Therefore, it is necessary to consider an extension of computational nets in which each variable is a computational object.

Definition 3.1: A computational object (or Com-object) has the following characteristics:

(1) It has valued attributes. The set consists of all attributes of the object O will be denoted by **M(O)**.
(2) There are internal computational relations between attributes of a Com-object O. These are manifested in the following features of the object:
- Given a subset A of M(O). The object O can show us the attributes that can be determined from A.
- The object O will give the value of an attribute.
- It can also show the internal process of determining the attributes.

Example 3.1: A triangle with some knowledge (formulas, theorems, etc ...) is an object. The attributes of a "triangle" object are 3 edges, 3 angles, etc. A "triangle" object can also answer some questions such as "Is there a solution for the problem that to compute the surface from one edge and two angles?".

Definition 3.2: A computational relation f between attributes of certain objects is called a *relation between the objects*. A network of Com-objects will consists of a set of Com-objects **O** = {$\mathbf{O_1, O_2, ..., O_n}$} and a set of computational relations **F** = {$\mathbf{f_1, f_2, ... , f_m}$}. This network of Com-objects is denoted by **(O, F)**. The following are some notations:

M(f_i) = the set of attributes of C-objects in the relation f_i.

$$\mathbf{M(F)} = \bigcup_{i=1}^{m} M(f_i).$$

$$\mathbf{M(O)} = \bigcup_{i=1}^{n} M(O_i).$$

M = the set of attributes of C-objects are considered in certain problem.

$\mathbf{M_i} = M \cap M(O_i)$, for i=1,2, ... , m.

By the above notations, M_i is the set of attributes considered of the object O_i.

On the network of Com-objects (O, F), we consider the problem that to determine (or compute) attributes in set G from given attributes in set H. The problem will be denoted by H→G.

Example 3.2: In the figure 1 below, suppose that AB = AC, the values of the angle A and the edge BC are given (hypothesis). ABDE and ACFG are squares. Compute EG.

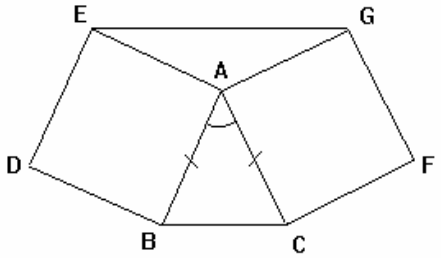

Fig. 1 A problem in geometry

The problem can be considered on the network of Com-objects (O, F) as follows:

O = {O_1: triangle ABC with AB = AC, O_2 : triangle AEG, O_3 : square ABDE, O_4 : square ACFG }, and F = {f_1, f_2, f_3, f_4, f_5} consists of the following relations

f_1 : $O_1.c = O_3.a$
{the edge c of triangle ABC = the edge of the square ABDE}
f_2 : $O_1.b = O_4.a$
{the edge b of triangle ABC = the edge of the square ACFG}
f_3 : $O_2.b = O_4.a$
{the edge b of triangle AEG = the edge of the square ACFG}
f_4 : $O_2.c = O_3.a$
{the edge c of triangle AEG = the edge of the square ABDE}
f_5 : $O_1.A + O_2.A = \pi$.

Definition 3.3: Let (O, F) be a network of Com-objects, and M be a set of concerned attributes. Suppose A is a subset of M.

(a) For each f ∈ F, denote f(A) is the union of the set A and the set consists of all attributes in M deduced from A by f. Similarly, for each Com-object $O_i \in O$, O_i(A) is the union of the set A and the set consists of all attributes (in M) that the object O_i can determine from attributes in A.
(b) Suppose D = [t_1, t_2, ..., t_m] is a list of elements in F ∪ O. Denote
$A_0 = A$, $A_1 = t_1(A_0)$, . . ., $A_m = t_m(A_{m-1})$, and
$D(A) = A_m$.

We have $A_0 \subseteq A_1 \subseteq \ldots \subseteq A_m = D(A) \subseteq M$.

A problem H → G is called *solvable* if there is a list D ⊆ F ∪ O such that D(A) ⊇ B. In this case, we say that D is a *solution* of the problem.

Technically the theorems and algorithms in section II can be develop to obtain the new ones for solving the problem H → G on network of Com-objects (O, F). They will be omitted here except the algorithm to find a solution of the problem. The worthy of note is that the objects may participate in solutions as computational relations.

Algorithm 3.1: Find a solution of the problem H→G on a network of Com-objects.

Step 1: Solution ← empty;

Step 2: **if** $G \subseteq H$ **then**
begin
Solution_found ← true;
goto step 5;
end
else
Solution_found ← false;
Step 3: Repeat
Hold ← H;
Select $f \in F$;
while not Solution_found **and** (f found) **do**
begin
if (applying f from H produces new facts) **then**
begin
$H \leftarrow H \cup M(f)$;
Add f to Solution;
end;
if $G \subseteq H$ **then**
Solution_found ← true;
Select new $f \in F$;
end; { while }
Until Solution_found **or** (H = Hold);
Step 4: **if** not Solution_found **then**
begin
Select $O_i \in O$ such that $O_i(H) \neq H$;
if (the selection is successful) **then**
begin
$H \leftarrow O_i(H)$;
Add O_i to Solution;
if $(G \subseteq H)$ **then**
begin
Solution_found ← true;
goto step 5;
end;
else
goto step 3;
end;
end;
Step 5: **if not** Solution_found **then**
There is no solution found;
else
Solution is a solution of the problem;

Example 3.3: Consider the network (O, F) in example 3.2, and the problem $H \rightarrow G$, where $H = \{O_1.a, O_1.A\}$, and $G = \{O_2.a\}$.
Here we have:
$M(f_1) = \{ O_1.c , O_3.a \}$,
$M(f_2) = \{ O_1.b , O_4.a \}$,
$M(f_3) = \{ O_2.b , O_4.a \}$,
$M(f_4) = \{ O_2.c , O_3.a \}$,
$M(f_5) = \{ O_1.\alpha , O_2.\alpha \}$,
$M = \{ O_1.a, O_1.b, O_1.c, O_1.A, O_2.b, O_2.c, O_2.A , O_2.a, O_3.a, O_4.a \}$.

The above algorithms will produce the solution
$D = \{ f_5, O_1, f_1, f_2, f_3, f_4, O_2\}$,
And the process of extending the set of attributes as follows:

$$A_0 \xrightarrow{f_5} A_1 \xrightarrow{O_1} A_2 \xrightarrow{f_1}$$
$$A_3 \xrightarrow{f_2} A_4 \xrightarrow{f_3} A_5 \xrightarrow{f_4}$$
$$A_6 \xrightarrow{O_2} A_7$$

Where
$A_0 = A = \{O_1.a , O_1.A\}$,
$A_1 = \{O_1.a , O_1.A, O_2.A\}$,
$A_2 = \{ O_1.a , O_1.A, O_2.A, O_1.b, O_1.c \}$,
$A_3 = \{O_1.a , O_1.A, O_2.A, O_1.b, O_1.c, O_3.a\}$,
$A_4 = \{O_1.a , O_1.A, O_2.A, O_1.b, O_1.c, O_3.a, O_4.a\}$,
$A_5 = \{O_1.a , O_1.A, O_2.A, O_1.b, O_1.c, O_3.a, O_4.a, O_2.b\}$,
$A_6 = \{O_1.a , O_1.A, O_2.A, O_1.b, O_1.c, O_3.a, O_4.a, O_2.b, O_2.c\}$,
$A_7 = \{O_1.a , O_1.A, O_2.A, O_1.b, O_1.c, O_3.a, O_4.a, O_2.b, O_2.c, O_2.a\}$.

IV. EXTENSION OF COMPUTATIONAL NETWORKS

Computational Networks with simple valued variables and networks of computational objects can be used to represent knowledge in many domains of knowledge. The basic components of knowledge consist of a set of simple valued variables and a set of computational relations over the variables. However, there are domains of knowledge based on a set of elements, in which each element can be a simple valued variables or a function. For example, in the knowledge of alternating current the alternating current intensity i(t) and the alternating potential u(t) are functions. It requires considering some extensions of computational networks such as *extensive computational networks* and *extensive computational objects networks* that are defined below.

Definition 4.1: An *extensive computational network* is a structure (M, R) consisting of two following sets:

- $M = M_v \cup M_f$ is a set of attributes or elements, with simple valued or functional valued. $M_v = \{x_{v1}, x_{v2}, \ldots, x_{vk}\}$ is the set of simple valued variables. $M_f = \{x_{f1}, x_{f2}, \ldots x_{fm}\}$ is the set of functional valued elements.
- $R = R_{vv} \cup R_{fv} \cup R_{vf} \cup R_{fvf}$ is the set of deduction rules, and R is the union of four subsets of rules R_{vv}, R_{fv}, R_{vf}, R_{fvf}. Each rule r has the form $r: u(r) \rightarrow v(r)$, with u(r) is the hypotheses of r and v(r) is the conclusion of r. A rule is also one of the four cases below.
  - Case 1: $r \in R_{vv}$. For this case, $u(r) \subseteq M_v$ and $v(r) \subseteq M_v$.

- Case 2: $r \in R_{fv}$. For this case, $u(r) \subseteq M_f$ and $v(r) \subseteq M_v$.
- Case 3: $r \in R_{vf}$. For this case, $u(r) \subseteq M_v$ and $v(r) \subseteq M_f$.
- Case 4: $r \in R_{fvf}$. For this case, $u(r) \subseteq M$, $u(r) \cap M_f \neq \emptyset$, $u(r) \cap M_v \neq \emptyset$, and $v(r) \subseteq M_f$.

Each rule in R has the corresponding computational relation in the set $F = F_{vv} \cup F_{fv} \cup F_{vf} \cup F_{fvf}$.

Definition 4.2: An *extensive computational Object* (ECom-Object) is an object O has structure including:

(1) A set of attributes $Attr(O) = M_v \cup M_f$, with M_v is a set of simple valued variables; M_f is a set of functional variables. Between the variables (or attributes) there are internal relations, that are deduction rules or the computational relations.

(2) The object O has behaviours of reasoning and computing on attributes of objects or facts such as:
- Find the closure of a set $A \subset Attr(O)$.
- Find a solution of problems which has the form $A \rightarrow B$, with $A \subseteq Attr(O)$ and $B \subseteq Attr(O)$.
- Perform computations.
- Consider determination of objects or facts.

Definition 4.3: An *extensive computational objects network* is a model **(O, M, F, T)** that has the components below.

(1) $O = \{O_1, O_2, \ldots, O_n\}$ is the set of extensive computational objects.

(2) M is a set of object attributes. We will use the following notations:
$M_v(O_i)$ is the set of simple valued attributes of the object O_i, $M_f(O_i)$ is the set of functional attributes of O_i, $M(O_i) = M_v(O_i) \cup M_f(O_i)$, $M(O) = M(O_1) \cup M(O_2) \cup \ldots \cup M(O_n)$, and $M \subseteq M(O)$.

(3) $F = F(O)$ is the set of the computational relations on attributes in M and on objects in O.

(4) $T=\{t_1, t_2, \ldots, t_k\}$ is set of operators on objects.

On the structure (O, T), there are expressions of objects. Each expression of objects always has its attributes as if it is an object.

The extensions of the computational networks are more powerful in designing knowledge bases in reality.

V. CONCLUSION AND FUTURE WORKS

Computational Networks with simple valued variables and networks of computational objects are useful models. It can be used to represent knowledge in many domains of knowledge. The methods and techniques for solving the problems on the networks will be useful tool for design intelligent systems, especially systems that can solve problems based on a knowledge base.

There are domains of knowledge with functional attributes such as knowledge of alternating current in physics. This motivates another extensions of the above computational networks presented in the previous sections. The new computational networks with its simple valued variables and functional variables will be considered. Also, the computational objects in future works will have functional attributes. On the network of computational objects, operators will be considered. Such future works on computational networks make them more powerful for representing knowledge in practice.

REFERENCES

[1] Stuart Russell & Peter Norvig, *Artificial Intelligence – A modern approach* (second edition), Prentice Hall, 2003.
[2] John F. Sowa. *Knowledge Representation: Logical, Philosophical and Computational Foundations*, Brooks/Cole, 2000
[3] George F. Luger, *Artificial Intelligence: Structures And Strategies For Complex Problem Solving*, Addison Wesley Longman, 2008.
[4] Chitta Baral, *Knowledge Representation, Reasoning and Declarative Problem Solving*, Cambridge University Press, 2003.
[5] Do Van Nhon, "A Program for studying and Solving problems in Plane Geometry", in *Proc. Conf. on Artificial Intelligence 2000*, Las Vegas, USA, 2000, pp. 1441-1447.
[6] Do Van Nhon, "A system that supports studying knowledge and solving of analytic geometry problems", in *Proc. 16th World Computer Congress 2000 conf. on Education Uses of Information and Communication Technologies*, Beijing, China, 2000, pp. 236-239.
[7] Nhon Do, *An ontology for knowledge representation And Applications.* Waset, International Conference on Data, Information and Knowledge Management, Singapore, 2008.
[8] Michel Chein & Marie-Laure Mugnier, *Graph-based Knowledge representation: Computational foundations of Conceptual Graphs*, Springer-Verlag London Limited 2009.
[9] Frank van Harmelem & Vladimir & Bruce, *Handbook of Knowledge Representation*, Elsevier, 2008.
[10] F. Lehmann, *Semantic Networks in Artificial Intelligence*, Elsevier Science Ltd, 2008.
[11] Amit Konar, *Computational Intelligence : Principles, Techniques and Applications*, Springer-Verlag Berlin Heidelberg, 2005.
[12] Leszek Rutkowski, *Computational Intelligence: Methods and Techniques*, Springer-Verlag Berlin Heidelberg, 2008.
[13] ToshinoriMunakata, *Fundamentals of the New Artificial Intelligence: Neural, Evolutionary, Fuzzy and More*, Springer-Verlag London Limited, 2008.
[14] M. Tim Jones, *Artificial Intelligence : A System Approach*, Infinity Science Press LLC, 2008.
[15] Nhon Do & Tuyen Tran T. & Phan Truong H., *Design method for Knowledge Base Systems in Education using COKB-ONT.* Waset, International Conference on Communication and Information technologies in Education, Thailand, 2008.

Nhon Van Do is currently a senior lecturer in the faculty of Computer Science at the University of Information Technology, Ho Chi Minh City, Vietnam. He got his MSc and Ph.D. in 1996 and 2002 respectively, from The University of Natural Sciences – National University of Ho Chi Minh City. His research interests include Artificial Intelligence, computer science, and their practical applications, especially intelligent systems and knowledge base systems.