IJCSI International Journal of Computer Science Issues, Vol. 7, Issue 3, No 8, May 2010
ISSN (Online): 1694-0784
ISSN (Print): 1694-0814

Model for Knowledge Bases of Computational Objects

Nhon Van Do

**Department of Computer Science, University of Information Technology,
Ho Chi Minh City, Vietnam**

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 model for knowledge bases of computational objects will be presented. We also present the model for representing problems and algorithms, design methods using the model to construct applications.

Keywords: *knowledge base systems, knowledge representation, intelligent systems, knowledge processing.*

1. 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], [6] and [10] we have found popular methods for knowledge representation in designing knowledge base systems (KBS) and intelligent systems. They include predicate logic, semantic nets, frames, deductive rules. Many new methods and techniques were presented in [12], [13], [14], and [15]. 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 [6], [10] and [11]. 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 knowledge bases.

Therefore, it is needed to develop new models to represent knowledge in reality domains and also to represent problems with knowledge. In this paper, we present models for knowledge representation and they can be used to design knowledge base systems and intelligent systems in reality. The main model presented here is the model for knowledge bases of computational objects. This model can be used to represent the total knowledge and to design the knowledge base component of systems. Next, networks of computational objects can be used for modeling problems in knowledge domains. These models are tools for designing inference engine of systems. The models 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.

2. Model for Knowledge Bases of Computational Objects

The traditional methods for knowledge representation such as those presented in [2], [4], and [20] are interested and useful for many applications. However, those methods are not enough and not easy to use for constructing intelligent programs or knowledge base systems in different domains of knowledge, especially programs with human readable output. The model for knowledge bases of computational objects (KBCO model) has been established from Object-Oriented approach to represent knowledge together with programming techniques for symbolic computation. There have been many results and tools for Object-Oriented methods, and some principles as well as techniques were presented in [16]. This way also gives us a method to model problems and to design algorithms. The models are very useful for constructing components and the whole knowledge base of intelligent system in practice of knowledge domains.

2.1 Computational Objects

In many problems we usually meet many different kinds of objects. Each object has attributes and internal relations between them. They also have basic behaviors for solving problems on its attributes.

Definition 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.

The structure computational objects can be modeled by **(*Attrs, F, Facts, Rules*)**. *Attrs* is a set of attributes, *F* is a set of equations called computation relations, *Facts* is a set of properties or events of objects, and *Rules* is a set of deductive rules on facts. For example, knowledge about a triangle consists of elements (angles, edges, etc) together with formulas and some properties on them can be modeled as a class of C-objects whose sets are as follows:

Attrs = {A, B, C, a, b, c, R, S, p, ...} is the set of all attributes of a triangle,

F = {A+B+C = π; $\frac{a}{\sin(A)} = 2R$; $\frac{b}{\sin(B)} = 2R$;

$\frac{c}{\sin(C)} = 2R$; $\frac{a}{\sin(A)} = \frac{b}{\sin(B)}$;

$S = \frac{1}{2}bc\sin(A)$; ...},

Facts = {a+b>c; a+c>b; b+c>a ; …}, and

Rules = { {a>b} ⇔ {A>B}; {b>c} ⇔ {B>C};
{c>a} ⇔ {C>A}; {a=b} ⇔ {A=B};
{a^2= b^2+c^2} ⇒ {A=pi/2},
{A=pi/2} ⇒ {a^2 = b^2+c^2, b ⊥ c},
...}.

An object also has basic behaviors for solving problems on its attributes. Objects are equipped abilities to solve problems such as:

1. Determines the closure of a set of attributes.
2. Executes deduction and gives answers for questions about problems of the form: determine some attributes from some other attributes.
3. Executes computations
4. Suggests completing the hypothesis if needed.

For example, when a triangle object is requested to give a solution for problem {a, B, C} ⇒ S, it will give a solution consists of three following steps:

Step 1: determine A, by A = π -B-C;
Step 2: determine b, by b = a.sin(B)/sin(A);
Step 3: determine S, by S = a.b.sin(C)/2;

2.2 Components of the KBCO model

Definition 2: The model for knowledge bases of computational objects (KBCO model) consists of six components:

(C, H, R, Ops, Funcs, Rules).

The meanings of the components are as follows:
- **C** is a set of concepts of computational objects. Each concept in C is a class of Com-objects.
- **H** is a set of hierarchy relation on the concepts.
- **R** is a set of relations on the concepts.
- **Ops** is a set of operators.
- **Funcs** is a set of functions.
- **Rules** is a set of rules.

There are relations represent specializations between concepts in the set *C*; *H* represents these special relations on *C*. This relation is an ordered relation on the set C, and H can be considered as the Hasse diagram for that relation. The figure 1 below represents special relations on the classes of triangles.

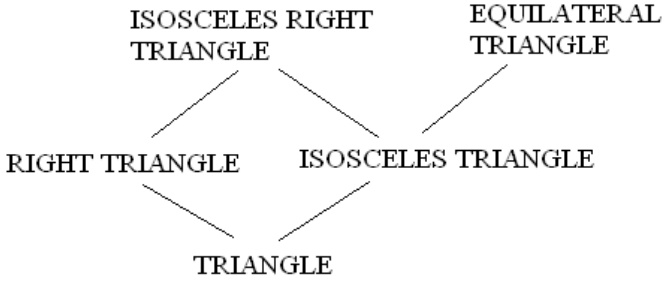

Fig. 1 Specialization relations on the classes of triangles

R is a set of other relations on *C*, and in case a relation r is a binary relation it may have properties such as reflexivity, symmetry, etc…. In plane geometry and analytic geometry, there are many such relations: relation "belongs to" of a point and a line, relation "central point" of a point and a line segment, relation "parallel" between two line segments, relation "perpendicular" between two line segments, the equality relation between triangles, etc.

The set *Ops* consists of operators on *C*. This component represents a part of knowledge about operations on the objects. Almost knowledge domains have a component consisting of operators. In analytic geometry there are vector operators such as addition, multiplication of a vector by a scalar, cross product, vector product; in linear algebra there are operations on matrices. The KBCO model helps to organize this kind of knowledge in knowledge domains as a

component in the knowledge base of intelligent systems.

The set *Funcs* consists of functions on Com-Objects. Knowledge about functions is also a popular kind of knowledge in almost knowledge domains in practice, especially fields of natural sciences such as fields of mathematics, fields of physics. In analytic geometry we have the functions: distance between two points, distance from a point to a line or a plane, projection of a point or a line onto a plane, etc. The determinant of a square matrix is also a function on square matrices in linear algebra.

The set *Rules* represents for deductive rules. The set of rules is certain part of knowledge bases. The rules represent for statements, theorems, principles, formulas, and so forth. Almost rules can be written as the form "if <facts> then <facts>". In the structure of a deductive rule, <facts> is a set of facts with certain classification. Therefore, we use deductive rules in the KBCO model. Facts must be classified so that the component *Rules* can be specified and processed in the inference engine of knowledge base system or intelligent systems.

Base on the KBCO model, the knowledge base can be organized by the following components:

1. The dictionary of concepts about kinds of objects, attributes, operators, functions, relations and related concepts.
2. The table of descriptions for structures and features of objects. For example, we can request a triangle to compute and to give us its attributes.
3. The tables for representing hierarchical relations of concepts.
4. The tables for representing other relations of concepts.
5. The tables for representing knowledge about operators.
6. The tables for representing knowledge about functions.
7. The tables of descriptions for kinds of facts. For example, a relational fact consists of kind of the relation and the list of objects in the relation.
8. The tables of descriptions for rules. For example, a deductive rule consists of hypothesis part and conclusion part. Both of them are lists of facts.
9. The lists or sets of rules.
10. The lists of problem patterns.

2.3 Kinds of facts in KBCO model

In the KBCO model there are 11 kinds of facts accepted. These kinds of facts have been proposed from the researching on real requirements and problems in different domains of knowledge. The kinds of facts are as follows:

- **Fact of kind 1**: information about object kind. The followings are some examples:
  - ABC is a right triangle.
  - ABCD is a parallelogram.
  - The matrix A is a square matrix.
- **Fact of kind 2**: a determination of an object or an attribute of an object. The following problem in analytic geometry gives some examples for facts of kind 2.

  Problem: Given the points E and F, and the line (d). Suppose E, F, and (d) are determined. (P) is the plane satisfying the relations: E $\in$ (P), F $\in$ (P), and (d) // (P). Find the general equation of (P). In this problem we have three facts of kind 3: (1) point E is determined or we have already known the coordinates of E, (2) point F is determined, (3) line (d) is determined or we have already known the equation of (d).
- **Fact of kind 3**: a determination of an object or an attribute of an object by a value or a constant expression. The followings are some examples in plane geometry and in analytic geometry:
  - In the triangle ABC, suppose that the length of edge BC = 5.
  - The plane (P) has the equation $2x + 3y – z + 6 = 0$, and the point M has the coordinate (1, 2, 3).
- **Fact of kind 4**: equality on objects or attributes of objects. This kind of facts is also popular, and there are many problems related to it on the knowledge base. The following problem in plane geometry gives some examples for facts of kind 4.

  Problem: Given the parallelogram ABCD. Suppose M and N are two points of segment AC such that AM = CN. Prove that two triangles ABM and CDN are equal (see figure 2).

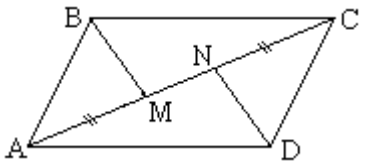

Fig. 2 Problem: prove that Δ ABM = Δ CDN

In the above problem we have to determine equality between two C-objects, a fact of kind 4.

- **Fact of kind 5**: a dependence of an object on other objects by a general equation. An example in geometry for this kind of fact is that w = 2*u + 3*v; here u, v and w are vectors.
- **Fact of kind 6**: a relation on objects or attributes of

the objects. In almost problems there are facts of kind 6 such as the parallel of two lines, a line is perpendicular to a plane, a point belongs to a line segment.

- **Fact of kind 7**: a determination of a function.
- **Fact of kind 8**: a determination of a function by a value or a constant expression.
- **Fact of kind 9**: equality between an object and a function.
- **Fact of kind 10**: equality between a function and another function.
- **Fact of kind 11**: a dependence of a function on other functions or other objects by an equation.

The last five kinds of facts are related to knowledge about functions, the component **Funcs** in the KBCO model. The problem below gives some examples for facts related to functions.

Problem: Let d be the line with the equation 3x + 4y - 12 = 0. P and Q are intersection points of d and the axes Ox, Oy.

(a) Find the central point of PQ

(b) Find the projection of O onto the line d.

For each line segment, there exists one and only one point which is the central point of that segment. Therefore, there is a function MIDPOINT(A, B) that outputs the central point M of the line segment AB. Part (a) of the above problem can be represented as to find the point I such that I = MIDPOINT(P,Q), a fact of kind 9. The projection can also be represented by the function PROJECTION(M, d) that outputs the projection point N of point M onto line d. Part (b) of the above problem can be represented as to find the point A such that A = PROJECTION(O,d), which is also a fact of kind 9.

The above models and kinds of facts can be used to represent knowledge in practical applications. Unification algorithms of facts were designed and used in different applications such as the system that supports studying knowledge and solving analytic geometry problems, the program for studying and solving problems in Plane Geometry, the knowledge system in linear algebra.

2.4 Specification Language

The language for the KBCO model is constructed to specify knowledge bases in intelligent systems with knowledge of the form KBCO model. This language includes the following:

- A set of characters: letter, number, special letter.
- Vocabulary: keywords, names.
- Data types: basic types and structured types.
- Expressions and sentences.
- Statements.
- Syntax for specifying the components of KBCO model.

The followings are some structures of definitions for expressions, C-Objects, relations, facts, and functions.

Definitions of expressions:

```
expr          ::=  expr | rel-expr | logic-expr
expr          ::=  expr add-operator term |
                   term
term          ::=  term mul-operator factor | factor
factor        ::=  – factor |
                   element ^ factor |
                   element
element       ::=  ( expr ) |
                   name |
                   number |
                   function-call
rel-expr      ::=  expr rel-operator expr
logic-expr    ::=  logic-expr OR logic-term |
                   logic-expr IMPLIES logic-term |
                   NOT logic-term |
                   logic-term
logic-term    ::=  logic-term AND logic-primary |
                   logic-primary
logic-primary ::=  expr |
                   rel-expr |
                   function-call |
                   quantify-expr
                   TRUE | FALSE
quantify-expr ::=  FORALL(name <, name>*),
        logic-expr | EXISTS(name), logic-expr
```

Definitions of Com-object type:

```
cobject-type  ::=  COBJECT name;
                       [isa]
                       [hasa]
                       [constructs]
                       [attributes]
                       [constraints]
                       [crelations]
                       [facts]
                       [rules]
                   ENDCOBJECT;
```

Definitions of computational relations:

```
crelations    ::=  CRELATION:
                   crelation-def+
                   ENDCRELATION;
```

crelation-def ::= **CR** name;
MF: name <, name>*;
MFEXP: equation;
ENDCR;
equation ::= expr = expr

Definitions of special relations:
isa ::= ISA: name <, name>*;
hasa ::= HASA:
[fact-def]

Definitions of facts:
facts ::= **FACT**: fact-def+
fact-def ::= object-type | attribute | name | equation | relation | expression
object-type ::= cobject-type **(**name**)** |
cobject-type **(**name <, name>* **)**
relation ::= relation **(** name <, name>+ **)**

Definitions of relations based on facts:
relation-def ::= **RELATION** name;
ARGUMENT: argument-def+
[facts]
ENDRELATION;
argument-def ::= name <, name>*: type;

Definitions of functions – form 1:
function-def ::= **FUNCTION** name;

ARGUMENT: argument-def+

RETURN: return-def;
[constraint]
[facts]
ENDFUNCTION;
return-def ::= name: type;

Definitions of functions – form 2:
function-def ::= **FUNCTION** name;
ARGUMENT: argument-def+
RETURN: return-def;
[constraint]
[variables]
[statements]
ENDFUNCTION;
statements ::= statement-def+
statement-def ::= assign-stmt | if-stmt | for-stmt
asign-stmt ::= name := expr;
if-stmt ::= **IF** logic-expr **THEN** statements+
ENDIF; |
IF logic-expr **THEN** statements+
ELSE statements+
ENDIF;
for-stmt ::= **FOR** name **IN [**range**] DO**
statements+
ENDFOR;

3. Networks of Computational Objects

Definition 3: A computational relation f between attributes of objects or between objects is called a *relation between the objects*. A network of computational objects 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.

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

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 1: In the figure 3 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.

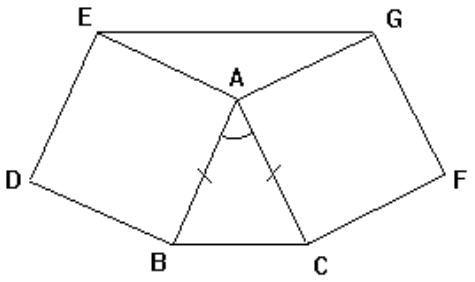

Fig. 3 A problem in geometry.

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 4: 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 $f \in F$, 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 $F \cup O$. Denote
$A_0 = A$, $A_1 = t_1(A_0)$, . . ., $A_m = t_m(A_{m-1})$, and
$D(A) = A_m$.

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

A problem $H \rightarrow G$ is called *solvable* if there is a list $D \subseteq F \cup O$ such that $D(A) \supseteq B$. In this case, we say that D is a *solution* of the problem.

The algorithm below is to find a solution of the problem $H \rightarrow G$ on the network of Com-objects. The objects may participate in solutions as computational relations.

Algorithm 1: Find a solution of the problem $H \rightarrow G$ 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 2: Consider the network (O, F) in example 1, 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 algorithm 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

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

4. Design Method

In this section, we will present a process to construct a knowledge base system for solving problems based on the knowledge base. Besides, techniques in each phase will be presented also.

4.1 Structure of Systems

A knowledge base system, which supports searching, querying and solving problems, has the structure of an expert system. We can design the system which consists of six components:

- The knowledge base.
- The inference engine.
- The explanation component.
- The working memory.
- The knowledge manager.
- The interface.

The figure 4 below shows the structure of the system.

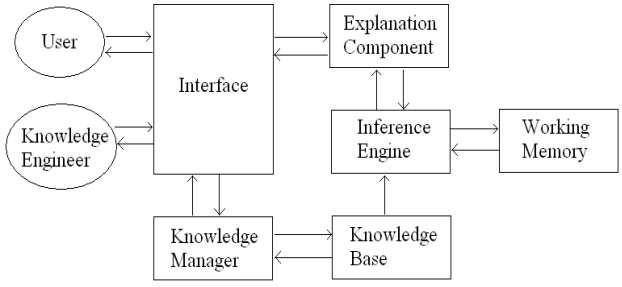

Fig. 4 Structure of a system

The Knowledge Base contains the knowledge for solving some problems in a specific knowledge domain. The Inference engine will use the knowledge stored in knowledge base to solve problems, to search or to answer for the query. It must identify problem and use suitable deductive strategies to find out right rules and facts for solving the problem. The working memory stores the facts and rules in the process of searching and deduction. The explanation component supports to explain the phases, concepts in the process of solving the problem. The knowledge manager aims to support updating knowledge into knowledge base. It also supports to search the knowledge and test consistence of knowledge. The interface component of the system is required to have a specification language for communication between the system and learners, between the system and teachers as well.

4.2 Design Techniques

The process of analysis and design the components of the systems consists of the following stages.

Stage 1: Collecting real knowledge based on KBCO model.

Stage 2: Classifying the knowledge in the Stage 1, to analyze requirements.

Stage 3: Establishing knowledge base organization for the system based on KBCO model and its specification language. Knowledge base can be organized by structured text files. They include the files below.

- File OBJECT_KINDS.txt stores names of concepts.
- File HIERARCHY.txt stores information of the Hasse diagram representing for the component H of COKB model.
- Files RELATIONS.txt and RELATIONS_DEF.txt store the specification of relations (the component R of KBCO model).
- Files OPERATORS.txt and OPERATORS_DEF.txt store the specification of operators (the component Ops of KBCO model).
- Files FUNCTIONS.txt and FUNCTIONS_DEF.txt store the specification of functions (the component Funcs of KBCO model).
- File FACT_KINDS.txt stores the definition of kinds of facts.
- File RULES.txt stores deductive rules.
- File SOMEOBJECTS.txt stores certain objects.

Stage 4: In this stage we do Modeling of problems and designing algorithms. Problems are represented using *networks of Com-Objects*. It consists of three sets below.

$O = \{O_1, O_2, \ldots, O_n\}$,
$F = \{f_1, f_2, \ldots, f_m\}$,
$Goal = \{ g_1, g_2, \ldots, g_m \}$.

In the above model the set O consists of n Com-objects, F is

the set of facts given on the objects, and Goal consists of goals.

The design of deductive algorithms for solving problems and the design of interface of the system can be developed by three steps for modeling:

Step 1: Classify problems such as problems as frames, problems of a determination or a proof of a fact, problems of finding objects or facts, etc…

Step 2: Classify facts and representing them based on the kinds of facts of KBCO model.

Step 3: Modeling kinds of problems from classifying in step 1 and 2. From models of each kind, we can construct a general model for problems, which are given to the system for solving them.

The basic technique for designing deductive algorithms is the unification of facts. Based on the kinds of facts and their structures, there will be criteria for unification proposed. Then it produces algorithms to check the unification of two facts.

The next important work is doing research on strategies for deduction to solve problems on computer. The most difficult thing is modeling for experience, sensible reaction and intuitional human to find the heuristics rules, which were able to imitate the human thinking for solving problems.

Stage 5: Creating a query language for the model. The language helps to design the communication between the system and users by words.

Stage 6: Designing the interface of software and programming the software. Our application has been implemented by using programming tools and computer algebra systems such as Visual Basic.NET or C#, SQL Server. They are very easy to use for students, to search, query and solve automatically problems.

Stage 7: In this stage we do testing, maintaining and developing the application. The work is similar as in other computer systems.

5. Applications

Some practical intelligent systems were produced with designing based on KBCO model and the above specification language. Applications include:

- The system that supports studying knowledge and solving analytic geometry problems. The system consists of three components: the interface, the knowledge base, the knowledge processing modules or the inference engine. The program has menus for users searching knowledge they need and they can access knowledge base. Besides, there are windows for inputting problems. Users are supported a simple language for specifying problems. There are also windows in which the program shows solutions of problems and figures.
- The program for studying and solving problems in plane geometry. It can solve problems in general forms. Users only declare hypothesis and goal of problems base on a simple language but strong enough for specifying problems. The hypothesis can consist of objects, relations between objects or between attributes. It can also contain formulas, determination properties of some attributes or their values. The goal can be to compute an attribute, to determine an object, a relation or a formula. After specifying a problem, users can request the program to solve it automatically or to give instructions that help them to solve it themselves. The program also gives a human readable solution, which is easy to read and agree with the way of thinking and writing by students and teachers. The second function of the program is "Search for Knowledge". This function helps users to find out necessary knowledge quickly. They can search for concepts, definitions, properties, related theorems or formulas, and problem patterns.
- Examples below illustrate the functions of a system for solving problems of analytic geometry and a system for solving problems in plane geometry. The systems are designed by using COKB model, its language and algorithms. The system was implemented in JAVA and MAPLE. Each example presents the problem in natural language, specifies the problem in specification language to input into the system, and a solution produced from the system.

Example 1: Let d be the line with the equation 3x + 4y - 12 = 0. P and Q are intersection points of d and the axes Ox, Oy.

(a) Find the midpoint of PQ

(b) Find the projection of O on the line d.

Specification of the problem:

Objects = {[d,line], [P,point], [Q,point]}.

Hypothesis = { d.f = (3*x+4*y-12 = 0), Ox.f = (y = 0),
O = [0, 0], P = INTERSECT(Ox, d),
Q = INTERSECT (Oy, d),
H = PROJECTION(O, d), Oy . f = (x = 0) }.

Goal = { MIDPOINT(P, Q), H }.

Solution found by the system:

Step 1: {d.f = (3*x+4*y-12 = 0), Ox.f = (y = 0),
Oy.f = (x = 0)} → {d.f, Ox.f , Oy.f }.
Step 2: {Ox.f, Oy.f, d.f}

→ {Ox, Oy, d}.
Step 3: {P = INTERSECT(Ox,d), d, Ox}
→{P = [4, 0]}.
Step 4: {d, Oy, Q = INTERSECT(Oy,d)}
→ {Q = [0, 3]}.
Step 5: {P = [4, 0], Q = [0, 3]}
→ {P, Q}.
Step 6: {P, Q}
→ {MIDPOINT(P,Q) = [2, 3/2]}.
Step 7: {d, H = PROJECTION(O,d), O}
→{ H = [36/25, 48/25]}.
Step 8: {H = [36/25, 48/25]}
→ {H}.

Example 2: Given two points P(2, 5) and Q(5,1). Suppose d is a line that contains the point P, and the distance between Q and d is 3. Find the equation of line d.

Specification of the problem:
Objects = {[P, point], [Q, point], [d, line]}.
Hypothesis = {DISTANCE(Q, d) = 3, P = [2, 5],
Q = [5, 1], ["BELONG", P, d]}.
Goal = [d.f].
Solution found by the system:
Step 1: {P = [2, 5]}
→ {P}.
Step 2: {DISTANCE(Q, d) = 3}
→ {DISTANCE(Q, d)}.
Step 3: {d, P}
→ {2d[1]+5d[2]+d[3] = 0}.
Step 4: {DISTANCE(Q, d) = 3}
→ $\frac{|5d[1] + d[2] + d[3]|}{\sqrt{d[1]^2 + d[2]^2}} = 3$.
Step 5: {d[1] = 1, 2d[1] + 5d[2] + d[3] = 0,
$\frac{|5d[1] + d[2] + d[3]|}{\sqrt{d[1]^2 + d[2]^2}} = 3$ }
→ {d.f = ($x + \frac{24}{7} y - \frac{134}{7} = 0$),
d.f = (x – 2 = 0)}.
Step 6: {d.f = $x + \frac{24}{7} y - \frac{134}{7} = 0$, d.f = x - 2= 0}
→ {d.f}

Example 3: Given the parallelogram ABCD. Suppose M and N are two points of segment AC such that AM = CN. Prove that two triangles ABM and CDN are equal (see figure 2 in section II-B above).

Specification of the problem:
Objects = {[A, POINT], [B, POINT], [C, POINT],
[D, POINT], [M, POINT], [N, POINT],
[O1, PARALLELOGRAM[A, B, C, D],
[O2, TRIANGLE[A, B, M]],
[O3, TRIANGLE [C, D, N]]}.
Hypothesis = { [« BELONG », M, SEGMENT[A, C]],
[« BELONG », N, SEGMENT[A, C]],
SEGMENT[A, M] = SEGMENT[C, N] }.
Goal = { O2 = O3}.
Solution found by the system:
Step 1: Hypothesis
→ {O2.SEGMENT[A, M] = O3. SEGMENT[C, N],
O2.SEGMENT[A, B] = O1. SEGMENT[A, B],
O3.SEGMENT[C, D] = O1.SEGMENT[C, D]}.
Step 2: Produce new objects related to O2, O3, and O1
→ {[O4, TRIANGLE[A, B, C]],
[O5, TRIANGLE[C, D, A]]}.
Step 3: {[O1, PARALLELOGRAM[A, B, C, D]}
→{O4 = O5, SEGMENT[A, B] = SEGMENT[C, D]}.
Step 4: { O2.SEGMENT[A, B] = O1.SEGMENT[A, B],
O3.SEGMENT[C, D] = O1.SEGMENT[C, D],
SEGMENT[A, B] = SEGMENT[C, D]}
→{O2.SEGMENT[A, B] = O3.SEGMENT[C, D]}.
Step 5: {[« BELONG », M, SEGMENT[A, C]]}
→ {O4.angle_A = O2.angle_A}.
Step 6: {[« BELONG », N, SEGMENT[A, C]]}
→ { O5.angle_A = O3.angle_A }.
Step 7: {O4 = O5 }
→ {O4.angle_A = O5.angle_A}.
Step 8: { O4.angle_A = O2.angle_A ,
O5.angle_A = O3.angle_A ,
O4.angle_A = O5.angle_A }
→{ O2.angle_A = O3.angle_A}.
Step 9: { O2.SEGMENT[A, M] = O3. SEGMENT[C, N],
O2.SEGMENT[A, B] = O3.SEGMENT[C, D],
O2.angle_A = O3.angle_A }
→ {O2 = O3}.

6. Conclusions

The KBCO model is a knowledge representation tool that can be used to design and to implement intelligent systems for solving problems based on a knowledge base. It also has the specification language, the network of Com-Objects for modeling problems, algorithms for automated problem solving.

The models proposed provide a natural way for representing knowledge. By Object-Oriented approach the highly intuitive representation for knowledge has been established. These are the bases for designing the knowledge base of the system. The knowledge base is convenient for accessing and for using by the inference engine. The methods of modeling problems and algorithms for automated problem solving represent a normal way of thinking and writing of people.

KBCO model is a useful tool and method for designing practical knowledge bases, modeling complex problems and designing algorithms to solve automatically problems based on a knowledge base. It is also used for designing other components of knowledge base systems. The KBCO model was used to produce intelligent educational softwares for e-learning, and they were implemented by using C++, JAVA, and MAPLE. Besides applications were presented here, it is able to use in other domain of knowledge such as physics and chemistry. Moreover, it also has been used to develop applications for e-government.

References

[1] John F. Sowa, Knowledge Representation: Logical, Philosophical and Computational Foundations, Brooks/Cole, 2000.
[2] George F. Luger & William A Stubblefield, Artificial Intelligence, Addison Wesley Longman, Inc. 1998.
[3] L. Stojanovic, J. Schneider, A. Maedche, S. Libischer, R. Suder, T. Lumpp, A. Abecker, G. Breiter, J. Dinger, The Role of Ontologies in Autonomic Computing Systems, IBM Systems Journal, Vol 43, No 3, 2004.
[4] Stuart Russell & Peter Norvig, Artificial Intelligence – A modern approach (second edition), Prentice Hall, 2003.
[5] Nhon Do, "An ontology for knowledge representation And Applications". WASET, International Conference on Data, Information and Knowledge Management, Singapore, 2008.
[6] Michel Chein & Marie-Laure Mugnier, Graph-based Knowledge representation: Computational foundations of Conceptual Graphs, Springer-Verlag London Limited 2009.
[7] Gruber, T. R., "Toward Principles for the Design of Ontologies Used for Knowledge Sharing". International Journal Human-Computer Studies, 43(5-6):907-928, 1995.
[8] Do Van Nhon, "A Program for studying and Solving problemsin Plane Geometry", in proceedings of International Conference on Artificial Intelligence 2000, Las Vegas, USA, 2000, pp. 1441-1447.
[9] Do Van Nhon, "A system that supports studying knowledge and solving of analytic geometry problems", 16th World Computer Congress 2000, Proceedings of Conference on Education Uses of Information and Communication Technologies, Beijing, China, 2000, pp. 236-239.
[10] Frank van Harmelem, Vladimir, and Bruce, Handbook of Knowledge Representation, Elsevier, 2008.
[11] F. Lehmann, Semantic Networks in Artificial Intelligence, Elsevier Science Ltd, 2008.
[12] Amit Konar, Computational Intelligence : Principles, Techniques and Applications, Springer-Verlag Berlin Heidelberg, 2005.
[13] Leszek Rutkowski, Computational Intelligence: Methods and Techniques, Springer-Verlag Berlin Heidelberg, 2008.
[14] ToshinoriMunakata, Fundamentals of the New Artificial Intelligence: Neural, Evolutionary, Fuzzy and More, Springer-Verlag London Limited, 2008.
[15] M. Tim Jones, Artificial Intelligence : A System Approach, Infinity Science Press LLC, 2008.
[16] Berge, J.M., Levia, O., and Rouillard, J., Object-Oriented Modeling. Netherlands: Kluwer Academic Publishers, 1996.
[17] Joseph C. Giarratano, and Gary D. Riley, Expert Systems: Principles and programming, fourth edition, International Thomson Publishing (2004).
[18] Nhon Van Do & Tam Pham Huu, "The Extensive Computational Network and Applying in an Education Software", in proceedings of ICAIE 2009 – Wuhan, P.R China, August, 22-23, 2009, pp 720-727, ISBN 978-1-84626-010-0 (Volume 2).
[19] Asunción Gómez-Pérez, Mariano Férnandez-López, and Oscar Corcho, Ontological Engineering. Springer-Verlag, 2004.
[20] Chitta Baral, Knowledge Representation, Reasoning and Declarative Problem Solving, Cambridge University Press, 2003.
[21] Guarino, N., "Formal Ontology, Conceptual Analysis and Knowledge Representation", International Journal of Human-Computer Studies, 43(5-6):625–640, 1995.
[22] Wen-tsun Wu, Mechanical Theorem Proving in Geometries. Springer-Verlag, 1994.
[23] Chou, S.C., Gao, X.S., and Zhang, J.Z., Machine Proofs in Geometry, Singapore: Utopia Press, 1994.
[24] Pfalzgraf, J., and Wang D., Automated Practical Reasoning. NewYork: Springer-Verlag, 1995.
[25] Lakemeyer G., and Nebel B., Foundations of Knowledge representation and Reasoning. Berlin Heidelberg: Springer-Verlag, 1994.
[26] Nie Guihua, Jiang Xiangjie, Chen Donglin, Liang Yueling, Li Xiaofei, "The Research of Personalized Learning Based On Ontology",in proceedings of ICAIE 2009 – Wuhan, P.R China, August, 22-23, 2009, pp 22-26, ISBN 978-1-84626-010-0 (Volume 1).

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.