International Journal of Information, Control and Computer Sciences
ISSN: 2517-9942
Vol:2, No:6, 2008

An Ontology for Knowledge Representation and Applications

Nhon Do

***Abstract*—**Ontology is a terminology which is used in artificial intelligence with different meanings. Ontology researching has an important role in computer science and practical applications, especially distributed knowledge systems. In this paper we present an ontology which is called Computational Object Knowledge Base Ontology. It has been used in designing some knowledge base systems 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 knowledge system in linear algebra**.**

***Keywords*—**Artificial intelligence, knowledge representation, knowledge base system, ontology.

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 [2], [3], [4] and [9] we have found popular methods for knowledge representation in designing knowledge base systems (KBS). Ontology is a terminology which is used in artificial intelligence with different meanings, and some of its definition can be found in [1], [3], [5], [8] and [10]. The followings are some definitions of the terminology ontology:

- "An Ontology is an explicit specification of a conceptualization" [5].
- "The subject *ontology* is the study of *categories* of things that exist or may exist in some domain. The product of such a study, called *an ontology*, is a catalog of the types of things that are assumed to exist in a domain of interest D from the perspective of a person who uses a language L for the purpose of talking about D" [3].
- For Artificial Intelligence researchers "an ontology describes a formal, shared conceptualization of a particular domain of interest. Thus, ontologies provide a way of capturing a shared understanding of a domain that can be used both by humans and systems to aid in information exchange and integration" [1].

Ontologies give us a modern approach for designing knowledge components of KBS. However, practical applications of intelligent systems expect more powerful and useful models for knowledge representation. In this paper an ontology, which is called "*Computational Object Knowledge Base Ontology*" (COKB-ONT), will be presented. It includes models, specification language, problems and deductive methods. The COKB-ONT was used to produce applications in education and training such as a program for studying and solving problems in plane geometry presented in [6], a system that supports studying knowledge and solving of analytic geometry problems presented in [7], and a knowledge base system in linear algebra. These intelligent programs must have suitable knowledge base and they not only give human readable solutions but also present solutions as the way teachers and students usually write them. The practical methods in [11], [12] and [13] are difficult to use for designing the above programs. Our 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.

The COKB-ONT has been shown that it is convenient for studying of users and for using by inference engine. Besides, problems are also modeled easily so that we can design algorithms for solving problems automatically and propose a simple language for specifying them.

II. Model of Computational Object Knowledge Base

The traditional methods for knowledge representation such as those presented in [2], [4], [9] and [14] 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 of computational object knowledge base 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 [15]. 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.

A. Components of the Model

The model of computational object knowledge base (COKB model) consists of 6 components:

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

The meanings of the components are as follows:

- **C** is a set of concepts of computational objects (C-Object).
- **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.

Each concept in C is a class of C-objects. The structure C-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;

There are relations represent specializations between concepts in the set *C*; *H* represent 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 Fig. 1 below represents special relations on the classes of triangles.

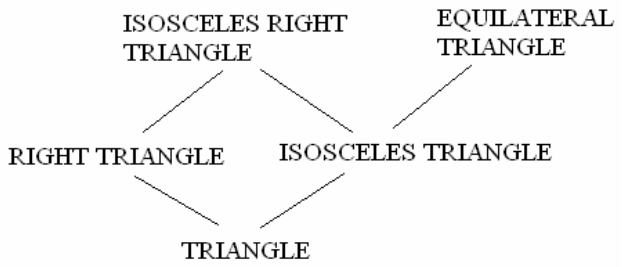

Fig. 1 Specialization relations on the classes of triangles

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 COKB 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 C-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 COKB 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 COKB 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.

B. Kinds of Facts in COKB Model

In the COKB 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 ∈ (P), F ∈ (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 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 Fig. 2).

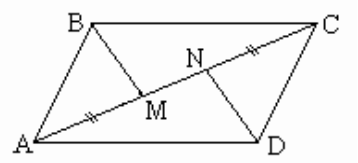

Fig. 2 Problem: prove that Δ ABM = Δ CDN

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 COKB 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. Discussion about these applications will be presented in section IV.

III. Specification Language

The language for the COKB model is constructed to represent the knowledge of the form COKB 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 COKB 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 C-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;
```

IV. Applications

Some practical intelligent systems were produced with designing based on COKB 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.

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

Stage 1: Collecting and classifying real knowledge based on COKB model, to analyze requirements.

Stage 2: Establishing knowledge base organization for the system based on COKB model and specification language. Knowledge base can be organized by structured text files. They include the files below.

- The file OBJECT_KINDS.txt stores names of concepts.
- The files <name of concept>.txt store the specifications of structures of C-Objects.
- The file HIERARCHY.txt stores the Hasse diagram representing for the component H of COKB model.
- The files RELATIONS.txt and RELATIONS_DEF.txt store the specification of relations (the component R of COKB model).
- The files OPERATORS.txt and OPERATORS_DEF.txt store the specification of operators (the component Ops of COKB model).
- The files FUNCTIONS.txt and FUNCTIONS_DEF.txt store the specification of functions (the component Funcs of COKB model).
- The file FACT_KINDS.txt stores the definition of kinds of facts.
- The file RULES.txt stores deductive rules.
- The file SOMEOBJECTS.txt stores certain objects.

In the appendix, a part of the files SEGMENT.TXT and TRIANGLE.TXT shows the specification of the concept “segment” and the concept “triangle” in the knowledge base of plane geometry; the definition of function MIDPOINT in the file FUNCTIONS_DEF.TXT will be also listed. The specification for some rules in the file RULES.TXT will be shown too. We also present a part of the file FUNCTIONS_DEF.TXT that specifies some functions in the knowledge base of analytic geometry.

Stage 3: Modeling problems and designing algorithms.

Problems are represented using a model that is called *networks of C-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 C-objects, F is the set of facts given on the objects, and Goal consists of goals. A goal of a problem may be the followings:

- Determine an object.
- Determine an attribute (or some attributes) of an object.
- Consider a relation between objects.
- Find a relation between objects.
- Find an expression relative to some objects.
- Compute a parameter (or some parameters).
- Compute a value of a function relative to objects.

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 COKB 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.

The following general algorithm represents one strategy for

solving problems: forward chaining reasoning with heuristics in which objects attend the reasoning process as active agents.

Step 1: Record the elements in hypothesis part and goal part.
Step 2: Check goal *G*. If *G* is obtained then goto step 7.
Step 3: Using heuristic rules to select a rule for producing new facts or new objects.
Step 4: If selection in step 3 fails then search for any rule, which can be used to deduce new facts or new objects.
Step 5: If there is a rule found in step 3 or in step 4 then record the information about the rule, new facts in Solution, and new situation (previous objects and facts together with new facts and new objects), and goto step 2.
Step 6: Else {search for a rule fails} Conclusion: Solution not found, and stop.
Step 7: Reduce the solution found by excluding redundant rules and information in the solution.

Some heuristic rules were used for reasoning are listed below.

1. Priority use of rules for determining objects and attributes.
2. Transform objects to objects at a higher level in the hierarchical graph if there are enough facts. For example, a triangle will become isosceles triangle if it has two equal edges.
3. Use rules for producing new objects that contains elements, which are not in existing objects.
4. Use rules for producing new objects that have relationship with existing objects, especially the goal, in necessary situations.
5. Try to use deduction rules to get new facts, especially facts that have relationship with the goal.
6. If we could not produce new facts or new objects then we should use parameters and equations.
7. There are always new facts (relations or expressions) when we produce new objects.

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, 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}.

V. Conclusion

The COKB-ONT is an ontology that can be used to design and to implement intelligent systems for solving problems based on a knowledge base. It consists of the COKB model, the specification language for COKB, the network of C-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.

COKB-ONT 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 COKB-ONT 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, COKB-ONT 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.

Appendix

A. Part of the File SEGMENT.TXT that Defines the Concept "Segment" in the Knowledge of Plane Geometry

```
begin_object: SEGMENT[_A,_B];
 _A,_B: POINT;

 begin_othername
end_othername

begin_variables
  a: REAL;  # length
end_variables

begin_constraints
  a > 0;
end_constraints

begin_construction_properties
   SEGMENT[_A,_B] = SEGMENT [_B,_A];
end_construction_properties

 begin_properties
  ["BELONG",_A,"Object"]
  ["BELONG",_B,"Object"]
end_properties

begin_computation_relations
end_computation_relations

begin_facts
end_facts

begin_rules


begin_rule
   kind_rule = "object_determined";
   a:REAL
   hypothesis_part:
     {a}
   end_hypothesis_part
   goal_part:
     {"Object"}
   end_goal_part
end_rule

 end_rules

end_object
```

B. Part of the File TRIANGLE.TXT that Defines the Concept "Triangle" in the Knowledge of Plane Geometry

```
begin_object: TRIANGLE[_A,_B,_C];

 _A,_B,_C: POINT

 begin_othername
```

```
end_othername

begin_variables
  angle_A : ANGLE[_C,_A,_B];
  angle_B : ANGLE [_A,_B,_C];
  angle_C : ANGLE [_B,_C,_A];
  a : SEGMENT[_B,_C];
  b : SEGMENT [_A,_C];
  c : SEGMENT [_A,_B];
  ha,hb,hc,ma,mb,mc,pa,pb,pc : SEGMENT;
  S,p,R, CV: REAL;
end_variables

begin_constraints
 S > 0;   p > 0;   R > 0;
end_constraints

begin_computation_relations

 begin_relation  0
  flag = 1
  Mf   ={angle_A.a, angle_B.a, angle_C.a}
  rf   =1
  vf   ={}
  expf =`  angle_A.a + angle_B.a + angle_C.a = Pi`
 end_relation

begin_relation  1
  flag = 1
  Mf={CV, SEGMENT[_B,_C].a, SEGMENT [_A,_C].a,
       SEGMENT [_A,_B].a}
  rf   =1
  vf   ={}
  expf = `CV = SEGMENT [_B,_C].a + SEGMENT
       [_A,_C].a + SEGMENT [_A,_B].a`
 end_relation

end_computation_relations


begin_rules

  begin_rule

  kind_rule = "theorem";
  _A,_B,_C : POINT
  hypothesis_part:
    {(SEGMENT[_B,_C].a)^2 =
(SEGMENT [_A,_B].a)^2 +( SEGMENT [_A,_C].a)^2}
  end_hypothesis_part
  goal_part:
    {ANGLE[_B,_A,_C].a= Pi/2}
  end_goal_part
 end_rule

begin_rule
  kind_rule = "theorem";
  _A,_B,_C : POINT
  hypothesis_part:
    {ANGLE[_B,_A,_C].a= Pi/2}
  end_hypothesis_part
  goal_part:
    {(SEGMENT[_B,_C].a)^2 =
(SEGMENT[_A,_B].a)^2+(SEGMENT[_A,_C].a)^2}
      end_goal_part
    end_rule

     begin_rule
      kind_rule = "";
      _A,_B,_C : POINT
      hypothesis_part:
        {ANGLE[_A,_B,_C].a= ANGLE[_A,_C,_B].a}
      end_hypothesis_part
      goal_part:
     {[TRIANGLE[_A,_B,_C],
"ISOSCELES_TRIANGLE"]}
      end_goal_part
    end_rule

   end_rules

 end_object
```

C. Definition of Function MIDPOINT

```
begin_function: MIDPOINT(_A,_B)
   _A,_B: POINT
   result _W: POINT

   begin_proc
   end_proc
   properties
     SEGMENT[_A,_W].a = SEGMENT [_B,_W].a
     ["BELONG", _W, SEGMENT[_A,_B]]
     SEGMENT [_A,_W].a = SEGMENT [_A,_B].a/2
     SEGMENT [_B,_W].a = SEGMENT [_A,_B].a/2
     SEGMENT [_A,_W].a + SEGMENT [_B,_W].a
        = DOAN[_A,_B].a
   end_properties
  end_function
```

D. Specification for a Rule in the File RULES.TXT

```
begin_rule
  kind_rule = "equality of triangles, edge-edge-edge"
  _A,_B,_C,_M,_N,_P: POINT
  hypothesis_part:
    {SEGMENT[_A,_B].a = SEGMENT[_M,_N].a,
  SEGMENT[_B,_C].a = SEGMENT[_N,_P].a,
  SEGMENT[_A,_C].a = SEGMENT[_M,_P].a}
  end_hypothesis_part
  goal_part:
    {TRIANGLE[_A,_B,_C] = TRIANGLE[_M,_N,_P]}
  end_goal_part
end_rule
begin_rule
  kind_rule = "equality of triangles, edge-angle-edge"
  _A,_B,_C,_M,_N,_P: POINT
hypothesis_part:
  {SEGMENT[_A,_B].a = SEGMENT[_M,_N].a,
    ANGLE[_A,_B,_C].a = ANGLE[_M,_N,_P].a,
    SEGMENT[_B,_C].a = SEGMENT[_N,_P].a}
end_hypothesis_part
goal_part:
```

```
    {TRIANGLE[_A,_B,_C] = TRIANGLE[_M,_N,_P]}
  end_goal_part
  end_rule
```

E. Part of the File FUNCTIONS_DEF.TXT that Specifies some Functions in the Knowledge Base of Analytic Geometry

```
  begin_function: MIDPOINT(A,B)
    A, B: POINT
    return I: POINT
    begin_proc
     local I;
     I.x := (A.x+B.x)/2;
     I.y := (A.y+B.y)/2;
     return I;
    end_proc
    properties
    end_properties
  end_function

  begin_function: PROJECTION(A, f)
    A: POINT
    f: LINE
    return B: POINT
    begin_proc
     local B, g;
     g :=  LineEquation(A, f,1);
     B := INTERSECT(f,g);
     return B;
    end_proc
    properties
    end_properties
  end_function
```

REFERENCES

[1] 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.
[2] Stuart Russell & Peter Norvig, Artificial Intelligence – A modern approach (second edition), Prentice Hall, 2003.
[3] John F. Sowa. Knowledge Representation: Logical, Philosophical and Computational Foundations, Brooks/Cole, 2000.
[4] George F. Luger & William A Stubblefield, Artificial Intelligence, Addison Wesley Longman, Inc. 1998.
[5] 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.
[6] Do Van Nhon, A Program for studying and Solving problems in Plane Geometry, Proceedings of International Conference on Artificial Intelligence 2000, Las Vegas, USA, 2000, pp. 1441-1447.
[7] Do Van Nhon, A system that supports studying knowledge and solving of analytic geometry problems, 16 th World Computer Congress 2000, Proceedings of Conference on Education Uses of Information and Communication Technologies, Beijing, China, 2000, pp. 236-239.
[8] Asunción Gómez-Pérez & Mariano Férnandez-López & Oscar Corcho, Ontological Engineering. Springer-Verlag, 2004.
[9] Chitta Baral, Knowledge Representation, Reasoning and Declarative Problem Solving, Cambridge University Press, 2003.
[10] Guarino, N. Formal Ontology, Conceptual Analysis and Knowledge Representation, International Journal of Human-Computer Studies, 43(5-6):625–640, 1995.
[11] Wen-tsun Wu, Mechanical Theorem Proving in Geometries. Springer-Verlag, 1994.
[12] Chou, S.C. & Gao, X.S. & Zhang, J.Z. Machine Proofs in Geometry. Singapore: Utopia Press, 1994.
[13] Pfalzgraf, J. & Wang, D. Automated Practical Reasoning. NewYork: Springer-Verlag, 1995.
[14] Lakemeyer, G. & Nebel, B. Foundations of Knowledge representation and Reasoning. Berlin Heidelberg: Springer-Verlag, 1994.
[15] Berge, J.M. & Levia, O. & Rouillard, J. Object-Oriented Modeling. Netherlands: Kluwer Academic Publishers, 1996.

Nhon 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.