Solving problems on a knowledge model of operators and application

Hien D. Nguyen*

University of Information Technology,
VNU-HCM,
Quarter 6, Linh Trung Ward, Thu Duc District,
Ho Chi Minh City, Vietnam
Email: hiennd@uit.edu.vn
and
National Institute of Informatics,
2 Chome-1-2 Hitotsubashi, Chiyoda,
Tokyo 101-8430, Japan
*Corresponding author

Nhon V. Do and Vuong T. Pham

University of Information Technology,
VNU-HCM, Vietnam
Email: nhondv@uit.edu.vn
Email: vuongpt@uit.edu.vn

Katsumi Inoue

National Institute of Informatics,
2 Chome-1-2 Hitotsubashi, Chiyoda,
Tokyo 101-8430, Japan
Email: inoue@nii.ac.jp

Abstract: Knowledge of operators is useful to build the intelligent problem solver for knowledge domains about computing. In this paper, we present a mathematical approach for building a knowledge model of operators, called Ops-model. The foundation of this model includes: concepts, operators, and inference rules. Each concept of this model is a class of objects with the behaviours for solving problems on themselves. General problems on this model are also studied, such as: reducing an expression, prove an equality of expressions. The algorithms for solving these problems are also designed. Ops-model has been applied to specify a part of knowledge domain about vector algebra in high school. It is used to construct a program for solving some problems on this knowledge domain. The solutions of this program are step-by-step, readable and suitable with the learners' level. It is useful for supporting students to learn this subject.

Keywords: knowledge representation; intelligent problem solver; automated reasoning; knowledge engineering; intelligent system; expert system; knowledge-based system; intelligent computing; digital enterprise.

Reference to this paper should be made as follows: Nguyen, H.D., Do, N.V., Pham, V.T. and Inoue, K. (2018) 'Solving problems on a knowledge model of operators and application', *Int. J. Digital Enterprise Technology*, Vol. 1, Nos. 1/2, pp.37–59.

Biographical notes: Hien D. Nguyen is currently a Lecturer in Faculty of Computer Science at the University of Information Technology, VNU-HCM, Vietnam. His research interests include knowledge representation, automated reasoning, knowledge engineering, especially the intelligent systems in education, such as intelligent problem solver.

Nhon V. Do is currently an Associate Professor at the University of Information Technology, VNU-HCM, Vietnam. His research interests include artificial intelligence, computer science, and their practical applications, especially intelligent systems and knowledge base systems.

Vuong T. Pham is currently a Lecturer in Software Engineering Faculty at the University of Information Technology, VNU-HCM, Vietnam. His research interests include artificial intelligence, software engineering, and game development.

Katsumi Inoue is a Full Professor at the National Institute of Informatics (NII) and SOKENDAI in The Graduate University for Advanced Studies, and is a Specially Appointed Professor at the Tokyo Institute of Technology, Japan. His research interests include artificial intelligence, logic programming and computer science.

This paper is a revised and expanded version of a paper entitled 'Method for solving problems on a knowledge model of operators' presented at 2016 IEEE Conference on e-Learning, e-Management and e-Services (IC3e 2016), Langkawi, Malaysia, 10–12 October 2016.

1 Introduction

Knowledge base and inference engine are important components in intelligent systems. Among practical applications, a popular kind of the knowledge is computational knowledge domain. This knowledge domain contains the component about operators. It helps to improve the accuracy of representation the knowledge of operators between objects. There are many methods for knowledge representation, such as: first order logic, description logic, rule-based systems, conceptual graph and ontology (Harmelem et al. 2008; Kent, 2013). Some knowledge models of operators have been proposed and applied in the knowledge domains of operators.

Formal model of knowledge about operators is very necessary for knowledge representation. This formal model would be a framework for designing knowledge based system of computational knowledge. There are many studies for researching this, but most of them have limitations, thus they are very difficult to apply in practice.

In this paper, a mathematical approach for building the structure of a knowledge model of operators has been presented. This model, namely ops-model, represents knowledge of operators between objects. The foundation of this model includes concepts, operators, and inference rules of knowledge. Each concept is an abstract structure that

includes attributes, equation and deductive rules; and objects in concept have been equipped behaviour to solve some problems on object. This model also refers to unary and binary operators and their characteristics: commutation, association, identity. In addition, the definition abut transforming an expression, simpler relation between two expressions are presented. The classes of the general problems in the knowledge of operators and their specification are also studied in this model. Besides that, the algorithms for solving these problems are built and proved their effectiveness. Furthermore, ops-model is applied to specify a part of knowledge domain of vector algebra in high school and construct an intelligent problem solver for solving some problems in this knowledge domain. Its solutions are step-by-step, readable and suitable with the learners' level.

2 Related work

Knowledge of operator is a popular form of knowledge domains, especially computational knowledge domains. In linear algebra, the operators between matrixes, vectors, vector spaces, linear maps are foundation of this knowledge domain (Anton and Rorres, 2010). In vector algebra, the summary, inner products, cross product between vectors are basic tools for solving the problems in mathematics curriculum (Varberg et al., 2003; Vietnam Ministry of Education and Training, 2013). In knowledge domain about direct current electrical circuits, the parallel and series connections between circuits, capacitors are the operators between these objects (Kuphaldt, 2017). However, these current knowledge models are not effective for representation practical applications.

Yang and Cai (2008) built the mathematical structure of knowledge representation based on extension rules, which particularly aimed to solve contradiction problems with formal model. Nevertheless, the extension rules model was not effective for representing real knowledge domains and this model also did not include operators.

The authors represent a method to simplify trigonometric expressions by using combination rules (Fu et al., 2006). However, the knowledge base of this method is setup by rule-based system, thus it can only be used for trigonometric polynomials. This model still has many limitations. In particular, it has not yet considered to represent the knowledge of operators.

Grefenstette (2013) and Sakama et al. (2017) use the linear algebraic approach to represent the operators in logic. These operators and their relations are represented by the matrixes and tensors. Nonetheless, these methods cannot use to represent the transforming of a logical expression and the application of them to represent the real knowledge domain is difficult.

Besides, some methods for automated theorem proving have been studied, such as: using Groebner bases (Roanes-Lozano et al., 2009), algebraic structure based on temporal logics (Cordero, 2004), etc. However, these methods cannot represent knowledge domains precisely. In addition, they also cannot perform the way of human's thinking to solve problems.

The other research, Knyazhansky (2012) solved the problem about the information equivalence of knowledge by constructing the automorphic of knowledge-base. Nonetheless, the knowledge domain in this study is too simple to apply in practice.

Recently, the research in Wang (2015) has built an algebraic structure for concepts with relational operators and compositional operations between concepts. This paper develops a visualised knowledge representation tool for concept algebra. Besides that, the authors in (Valipour and Wang, 2016) improved the formal properties and rules of the relational, reproductive, compositional operations of formal concepts. These properties can be applied for cognitive machine learning. However, these results are theoretical foundation, they have not yet been applied in practical application.

In the other research, Tulceanu (2016) used concept algebra for designing a reasoning mechanism on images. However, this application does not consider the problems about computation and operators, thus it cannot apply for knowledge domain about computation.

The COKB model has been used to perform many kinds of practical knowledge domains and to construct intelligent systems (Do, 2015). Nevertheless, the operator's component was not sufficiently studied in COKB. Do and Nguyen (2015) have presented a reduce model of COKB. This model improved the operators' component in COKB, and it also solved problems such as: specification of operators and object determination. This model has been applied to represent the knowledge about direct current electrical circuits. Nonetheless, the components of this model have not yet been described in detail; some operator-related problems were not solved: Reducing an expression and Transforming an expression.

3 Knowledge model of operators

The knowledge of operators between objects plays an important role in real knowledge domains, especially computational knowledge. In this section, a knowledge model about operators is presented. This model includes concepts, operators, and inference rules of knowledge.

In this paper, some symbols are used:

$\mathbb{R}$: set of real numbers

var(u): set of variables in express u

The knowledge model about operators, called ops-model, consists of three components:

$$\mathfrak{K} = \mathbf{(C, Ops, Rules)}$$

In which, C is a set of concepts, each concepts in C is a class of objects with their behaviours to solve problems on themselves. **R** is a set of relations on the concepts. **Ops** is a set of operators; each operators is a unary or binary mapping, we consider the properties of it are: commutative, associative, identity. **Rules** is a set of inference rules. A rule in this model is one of two forms: deductive rule and equivalent rule.

3.1 The components in ops-model

3.1.1 Structure of components

The components of ops-model have been modelled as followed table.

Table 1 Structure of ops-model

Level	*C*	*Ops*	*Rules*
			$Rule_{deduce} \cup Rule_{equi}$
$C_{(0)}$	• Set of real number $\mathbb{R}$	• Operators between real numbers $\mathbb{R}$	• Form 1: r ∈ $Rule_{deduce}$, r is an deductive rule, r has the form
	• The basic concepts	• Operators between concepts in $C_{(0)}$	$u_{(t)} = \{f_1, f_2, \ldots, f_p\} \rightarrow \{q_1, q_2, \ldots, q_k\} = v_{(t)}$
	1 The basic concept c has been built based on a set of elements. This set is an instance set, denote I_c.	$O_{(0)} = O^1_{(0)} \cup O^2_{(0)}$	where f_i, q_i are facts of model, in which, kinds of fact has been classified as definition 2.3
	2 $Ic \neq \emptyset$	In which:	
	3 Each $o \in I_c$, o is an object of concept c	1 Set of unary operators $O^1_{(0)}$:	• Form 2: r ∈ $Rule_{equi}$, r is an equivalent rule, r has the form
		$O^1_{(0)} \subset \{\oplus : I_{ci} \rightarrow I_{ck} \mid ci, ck \in C_{(0)}\}$	g = h
		2 Set of binary operators $O^2_{(0)}$	where g, h are expressions of objects
		$O^2_{(0)} \subset \{\otimes I_{ci} \times I_{ci} \rightarrow I_{ck} \mid ci, cj, ck \in C_{(0)}\}$	Denote: left(r) = g
$C_{(1)}$	*(Attrs, EqObj, RulesObj)*	• Operators between concepts in $C_{(0)}$ and $C_{(1)}$:	right(r) = h
	1 *Attrs* is a set of attributes:	$O_{(1)} = O^1_{(1)} \cup O^2_{(1)}$	
	$\emptyset \neq Attrs \subset \{x_i, i = 1..n \mid x_i \in I_{ci}, ci \in C_{(0)}\}$	In which:	
	2 *EqObj* is a set of equations between of attributes:	1 Set of unary operators $O^1_{(1)}$	
	$EqObj \subset \{g = h \mid g, h$ are expressions, $var(g) \subseteq Attrs, var(h) \subseteq Attrs\}$	$O^1_{(1)} \subset \{\oplus : I_{ci} \rightarrow I_{ck} \mid ci, ck \in C_{(0)} \cup C_{(1)}\}$	
	3 *RulesObj* is a set of deductive rules of concept:	2 Set of binary operators O2(1):	
	$RulesObj \subset \{u \rightarrow v \mid u, v$ are sets of attributes, $var(u) \subseteq Attrs, var(v) \subseteq Attrs, u \sqcap v = \emptyset\}$	$O^2_{(1)} \subset \{\otimes : I_{ci} \times I_{ci} \rightarrow I_{ck} \mid ci, cj, ck \in C_{(0)} \cup C_{(1)}\}$	
		Each operator is checked its properties: commutation, association, identity.	

Table 1 Structure of ops-model (continued)

Level	*C*	*Ops*	*Rules*
			$Rule_{deduce} \cup Rule_{equi}$
$C_{(2)}$	*(Attrs, EqObj, RulesObj)*	• Operators between concepts in $C_{(0)}$, $C_{(1)}$, and $C_{(2)}$	• Form 1: $r \in Rule_{deduce}$, r is an deductive rule, r has the form
	1 $\emptyset \neq Attrs \subset \{x_i, i = 1 \ldots n \mid x_i \in I_{ci}, ci \in C_{(0)} \cup C_{(1)}\}$	$O_{(2)} = O^1_{(2)} \cup O^2_{(2)}$	$u_{(r)} = \{f_1, f_2, \ldots, f_p\} \rightarrow \{q_1, q_2, \ldots, q_k\} = v_{(r)}$
	2 $\exists x_o \in Attrs, \exists c_{xo} \in C_{(1)}, x_o \in I_{cxo}$	In which:	where f_i, q_i are facts of model, in which, kinds of fact has been classified as definition 2.3
	3 $EqObj \subset \{g = h \mid g, h$ are expressions, $var(g) \subseteq Attrs, var(h) \subseteq Attrs\}$	1 Set of unary operators $O^1_{(2)}$	
	4 $RulesObj \subset \{u \rightarrow v \mid u, v$ are sets of attributes, $var(u) \subseteq Attrs, var(v) \subseteq Attrs, u \sqcap v = \emptyset\}$	$O^1_{(2)} \subset \{\oplus : I_{ci} \rightarrow I_{ck} \mid ci, ck \in C_{(0)} \cup C_{(1)} \cup C_{(2)}\}$	• Form 2: $r \in Rule_{equi}$, r is an equivalent rule, r has the form
		2 Set of binary operators $O^2_{(2)}$	$g = h$
		$O^2_{(2)} \subset \{\otimes : I_{ci} \times I_{cj} \rightarrow I_{ck} \mid ci, cj, ck \in C_{(0)} \cup C_{(1)} \cup C_{(2)}\}$	where g, h are expressions of objects
		Each operator is checked its properties: commutation, association, identity.	Denote: left(r) = g right(r) = h

3.1.2 Length of expression

Definition 3.1: definition of expression

<expr>::= o | ⊕ <expr> | <expr>⊗<expr>

o: object

⊕: unary operator ⊗: binary operator

If ⊗ is associative then: (p, q, r are expressions)

$p \otimes q \otimes r = (p \otimes q) \otimes r = p \otimes (q \otimes r)$

Definition 3.2: length of an expression

Let g be an expression, length(g) – length of expression g – is computed like this:

a if g only has object x then:

length(g) = 1 if $x \in I_c$ and $c \in C_{(0)}$

length(g) = 2 if $x \in I_c$ and $c \in C_{(1)}$

length(g) = 3 if $x \in I_c$ and $c \in C_{(2)}$

b if g = ⊕ f, which f is expression, and operator ⊕ is an unary operator, then:

length(g) = length(f) + 1

if g = f ⊗ h, which f, h are expressions, and ⊗ is a binary operator, then:

length(g) = length(f) + length(h)

Definition 3.3: Let p is an expression, we define a tree *T*(*p*) to represent p inductively as follows:

a if p is an object, then T(p) is a single node labelled with p

b if p = ⊕q, where ⊕ is an unary operator and q is an expression, then T(p) is a tree with the root labeled with ⊕ whose immediate successor is T(q)

c if p = q Θ r, where Θ is a binary operator, q and r are expressions, then T(p) is a tree with the root labeled with Θ which has two immediate successors T(q) and T(r).

d if $p = q_1 \otimes q_2 \ldots \otimes q_k$, where ⊗ is a associative binary operator and q_j (j = 1 …k) are expressions, then T(p) is a tree with ⊗ is the root labelled which has k immediate successors $T(q_1)$, …, $T(q_k)$.

Definition 3.4: let two logical expressions p and q:

p is a sub-expression of q ⇔ T(p) is a sub-tree of T(q)

Definition 3.5: let two expressions p and q, a relation 'simpler than' (<<) is a binary relation such that:

$$p << q \text{ if and only if } \begin{cases} \text{Height}(T(p)) \le \text{Height}(T(q)) \\ \text{length}(p) \le \text{length}(q) \end{cases}$$

with Height(T(*expr*)) is the height of tree T(*expr*).

The relation ‘simpler than’ has properties are: reflexive and transitive.

3.2 Unification of facts

Definition 3.6

a Classify kinds of facts

Kind	*Meaning*	*Specification*	*Condition*
1	Information about object kind	x:c	$x \in S^*, c \in C$
2	Determination an object	x	$x \in I_c, c \in C$
3	Determination an object by a value or a constant expression	x = <const>	$x \in I_c, c \in C$ <const>: constant
4	Equality on objects	x = y	$x, y \in I_c, c \in C$
5	Dependence of an object by an expression.	x = <expr>	$x \in I_c, c \in C$ <expr>: expression
6	Equality of expressions	<expr1> = <expr2>	<expr1>: expression <expr2>: expression

b The unification of two facts

Give two facts f_1 and f_2, they are unified, ≅, when they satisfy the following conditions:

1 f_1 and f_2 have them same kind k, and

2 if k = 1, 2: $f_1 = f_2$

if k = 3:

$left(f_1) \cong left(f_2)$ *and* $compute(right(f_1)) = compute(right(f_2))$

if k = 4:

$(left(f_1) \cong left(f_2)$ and $right(f_1) \cong right(f_1))$

or $(left(f_1) \cong right(f_2)$ and $right(f_1) \cong left(f_2))$

if k = 5, 6:

$simplify(expand(left(f_1) - right(f_1) - left(f_2) + right(f_2))) = 0$ *or*

$simplify(expand(left(f_1) - right(f_1) + left(f_2) - right(f_2))) = 0$

which

- compute(expr): compute the value of the expression expr.
- simplify(expr): simplify the expression expr.
- expand(expr): expand the expression expr.

c Relations on set of facts

Let x be a fact, A and B are sets of facts, the relations between them have been defined as followed:

$$x\ \Theta\ A \Leftrightarrow \exists\ g \in A, x \cong g$$
$$A \sqsubseteq B \Leftrightarrow \forall x \in A, x\ \Theta\ B$$
$$A \cong B \Leftrightarrow A \sqsubseteq B \wedge B \sqsubseteq A$$

$$A \sqcap B = \{x |\ x\ \Theta\ A\ \Theta\ x\ \Theta\ B\}$$
$$A \sqcup B = \{x |\ x\ \Theta\ A\ \Theta\ x\ \Theta\ B\}$$
$$A \backslash B = \{x |\ x\ \Theta\ A\ \Theta\ not(x\ \Theta\ B)\}$$

4 Model of problem and algorithms

4.1 Model of problem and solution

Definition 4.1: model of problems

a Kind 1: model of problems consists of three sets below:

$O = \{O_1, O_2, \ldots, O_m\}$, the set of objects in the problem.

$F = \{f_1, f_2, \ldots, f_n\}$, the set of facts

G = {'KEYWORD': expr} with 'KEYWORD' is a keyword of the goal and expr is an expression, 'KEYWORD' may be the followings:

- 'determine': it means to determine an expression or an object
- 'compute': it means to determine the value of an expression
- 'reduce': it means to reduce the expression.

The problem will be denoted by **(O, F) → G**

b Kind 2: model of problems has the form:

(O, F), E → G

where

$E = \{expr_1, expr_2, \ldots, expr_p\}$ is the set of expressions between objects in O.

G = {'KEYWORD': expr} with 'KEYWORD' may be the followings:

- 'prove': it means to prove an equality of expressions
- 'transform': it means to transform an object into an expression between certain objects.

Problems in kind 1 with goals are: determine an object and compute values of an attribute were studied and solved in Do and Nguyen (2015). In this paper, we will study the methods for solving the other problems in kinds 1 and 2.

Definition 4.2: transform an expression.

1 Let expr, s, u be expressions.

Denote: subs(expr, s, u) is a new expression that is substituted u for s in the expression expr.

2 Let f be an expression, f has a sup-expression g and an equivalent rule r.

- f *can be transformed* by rule r if g is a side of r.
  a if g = left(r): Let r(f) = subs(f, g, right(r))

b if g = right(r): Let r(f) = subs(f, g, left(r))

- OR f *can be transformed* by rule r if ∃ a variable p_o in r and an expression e_o such that g is a side of subs(r, p_o, e_o)

  a if g = left(subs(r, p_o, e_o)):

  Let r(f) = subs(f, g, right(subs(r, p_o, e_o)))

  b if g = right(subs(r, po, eo)):

  Let r(f) = subs(f, g, left(subs(r, p_o, e_o)))

An object in ops-model has basic behaviours for solving problems on its attributes. Determining the closure of a set of object's facts is most important behaviour.

Definition 4.3: *the closure set of object's facts*:

Let Obj = (Attrs, EqObj, RulesObj) be an object of a concept in C, and A is a set of facts on object Obj

a if e ⊂ EqObj: e is an equation system between k variables $\{x_1, x_2, \ldots, x_k\} \subseteq$ Attr

e can be applied to A if from facts of kind 3, 4 and 5 in A, we have:

- e can be solved to compute the values of $\{x_1, x_2, \ldots, x_k\}$.

Let e(A) = A ⊔ $\{x_1, x_2, \ldots, x_k\}$

- OR e can be produced new relations as equation between $\{x_1, x_2, \ldots, x_k\}$

Let:

$$e(A) = A \bigsqcup_{\substack{f \in A \\ f \text{ is kind } 3}} \text{subs}(e, \text{left}(f), \text{right}(f))$$

b If g ∈ RulesObj: g is a deductive rule, g has form: u(g) → v(g)

g *can be applied* to A if u(g) ⊑ A. Let g(A) = A ⊔ v(g)

c Let $r_0(A) = A$ and s = $[r_1, r_2, \ldots, r_m]$ with $r_k \in$ RulesObj or $r_k \subseteq$ EqObj, s is called *object deduce* if satisfies three conditions:

1 ∀*k*, *m*, r_k can be applied to $r_{k-1}(A)$.

Let $r_k(A) = r_k(r_{k-1}(A))$

2 ∀r ∈ RulesObj\$\{r_1, r_2, \ldots, r_m\}$, r cannot be applied on $r_m(A)$.

3 ∀r ⊆ EqObj\$\{r_1, r_2, \ldots, r_m\}$, r cannot be applied on $r_m(A)$

Let $D_{Obj}(A) = \{s = [r_1, r_2, \ldots, r_m] \mid s \text{ is an object deduce}\}$

d Let: Obj.Closure(A): = $r_m(A)$

Obj.Deduce(A): = d_A, with $d_A \in D_{Obj}(A)$ and $card(d_A) = \min\{card(s) \mid s \in D_{Obj}(A)\}$

Definition 4.4: let knowledge domain ℜ = (C, Ops, Rules) as ops-model, Obj = (Attr, EqObj, RuleObj) is an obiect of concepts in C, rule r ∈ Rules and A is a set of facts.

a Let $A|_{Obj} = \{f \in A \mid var(f) \subseteq Obj.Attrs\}$

Obj(A) = Obj.Closure($A|_{Obj}$)

b if $r \in Rule_{deduce}$: r has the form: $u(r) \rightarrow v(r)$

r *can be applied* on A if $u(r) \sqsubseteq A$

Let $r(A) = A \sqcup v(r)$

c if $r \in Rule_{equi}$: r is an equation of k objects, r has form $g(x_1, x_2, \ldots, x_i) = h(x_i + 1, x_i + 2, \ldots, x_k)$ with x_i is an object. $(i = 1, \ldots, k)$

*r can be applied on A if:

1 $card(A \sqcap \{x_1, x_2, \ldots, x_k\}) = k - 1$

Let $r(A) = A \sqcap \{x_1, x_2, \ldots, x_k\}$

2 OR $\exists f \in A$, f can be transformed by r.

Let $r(A) = A \setminus \{f\} \sqcup \{r(f)\}$

Definition 4.5: let knowledge domain $\mathfrak{K} = (C, Ops, Rules)$ as Ops-model, give a problem S on Ops-model. Suppose $D = [d_1, d_2, \ldots, d_k]$ is a list of elements which $d_j \in$ Rules or $d_j \in O$. Denote: $F_0 = F$, $F_1 = d_1(F_0)$, $F_2 = d_2(F_1)$, …, $F_k = d_k(F_{k-1})$ and $D(F) = F_k$.

A problem S is called *solvable* if there is a list D such that $G \subseteq D(F)$. In this case:

$\forall j = 1, \ldots, k$:

- If $d_j \in$ Rules
  1 If d_j is a deduce rule: $step_j = [d_j, u(d_j), v(d_j)]$
  2 If d_j is an equivalent rule: $step_i = [d_j, f_j, d_j(f_j)]$ with $f_j \in F_{j-1}$ and f_j can be transformed by d_j.
- If $d_j \in O$: $step_j = [d_j, F_{j-1}, d_j(F_{j-1}) \setminus F_{j-1}]$

Let $Sol = [step_1, step_2, \ldots, step_k]$ and Sol is a *solution* of problem S.

4.2 Algorithms for solving problem

Algorithm 4.1: let Obj = (Attrs, EqObj, RulesObj) be an object as section II and A be a set of facts related to Obj. This algorithm is determine Obj.Closure(A).

Input Object Obj = (Attrs, Facts, RulObj), A is a set of facts related to Obj.

Output Obj.Closure(A).

Step 0 flag := true;
KnownFacts := A;

Step 1 Searching rules in Obj.RulObj can be applied based on KnownFacts

while (flag!=false) **do**

2.1 **if** (rule r can be found) **then**

for e in v(r) **do**

KnownFacts := KnownFacts ⊔ {e};

if (new facts can be determined from KnownFacts) **then**

```
                    Determining new facts from facts in KnownFacts by apply reasoning
                    rules;
                end if;
            end do;
        end if;
    2.2 if (rule r cannot be founded) then
            flag := false;
        end if;
end do;
Step 2  Searching rules in Obj.EqObj can be applied
    flag:=true;
    while (flag!=false) do
        flag:=false;
        for e ⊂ Obj.EqObj do
            if (e can be applied to KnownFacts) then
                flag:=true;
                KnownFacts := KnownFacts ⊔ e(KnownFacts);
            end if;
        end do;
    end do;
Step 3  Obj.Closure(A) := KnownFacts
```

Algorithm 4.2: *algorithm for proving an equality of expressions*

Let knowledge domain $\mathcal{K}$ = (C, Ops, Rules) as Ops-model, and the problem S = (O, F), E → G as definition 3.1b, this algorithm will prove an equality of expressions. The solution of problem S has been found though these steps:

Input (O, F), E → G with E = {f, g}

Output The solution of the proof: f = g

```
Step 0  KnownFacts := F;
        Sol := [ ];
        Solution_found := false;
        flag:=true;
Step 1  Use objects in O and facts of KnownFacts to determine the closure of a set of object's
        facts by using algorithm 4.1
    while not(Solution_found) do
        for Obj in O do
            if (object Obj produces new facts) then
                KnownFacts := KnownFacts ⊔ Obj.Closure(KnownFacts)
                Sol:= [op(Sol), Obj];
```

if (G $\sqsubseteq$ KnownFacts) **then**
Solution_found := true;
end if;
end if;
end do; #for
end do; #while

Step 2 Searching rule in Rules-set can be applied based on KnownFacts

while (flag!=false) and not(Solution_found) **do**
flag:=false;
f1 := f;

2.1 **if** (rule r can be found) and ($r \in Rule_{equi}$) **then**
flag:=true; Sol:=[op(Sol), r];
if f1 << g **then** f1 := Exp_{f1}; # Exp_{f1} is defined as definition 4.6
else f1 := $Redu_{f1}$; # $Redu_{f1}$ is defined as definition 4.6
end if;
end if;

2.2 **if** (rule r can be found) and ($r \in Rule_{deduce}$) **then**
flag := true;
Sol:=[op(Sol), r];
for e in v(r) **do**
KnownFacts := KnownFacts $\sqcup$ {e};
if (new facts can be determined from KnownFacts) **then**
Determining new facts from facts in KnownFacts by apply reasoning rules;
end if;
end do;
end if;

2.3 **if** (G $\sqsubseteq$ KnownFacts) **then**
Solution_found := true;
end if;

2.4 **if** (rule r cannot be founded) **then**
Flag := false;
end if;

end do;

Step 3 **if** (Solution_found) **then**
From list Sol, build the solution of this problem.;
else
There is no solution was found;
end if;

Definition 4.6: let an expression g:

- *Simple_Expand*(g) := [r_1, r_2, …, r_k] is a list of equivalent rules in $Rule_{equi}$ such that:

$$g_i << g_{i+1} \text{ with} \begin{cases} g_0 = g \\ g_i = r_i(g_{i-1}) \end{cases} \quad i = \overline{1,k}$$

  Denote: $Exp_g = g_k$

- *Simple_Reduce*(g): = [r_1, r_2, …, r_m] is a list of equivalent rules in $Rule_{equi}$ such that:

$$g_{i+1} << g_i \text{ with} \begin{cases} g_0 = g \\ g_i = r_i(g_{i-1}) \end{cases} \quad i = \overline{1,m}$$

  Denote: $Redu_g = g_m$

Algorithm 4.3: Algorithm for reducing an expression

Let knowledge domain K = (C, Ops, Rules) as Ops-model, give the problem S = **(O, F) → G**. The problem S has the goal is reducing the expression f, G = {Reduce: expr}, it means searching the expression g satisfies:

$$g \equiv \text{expr } and \ \forall h, h \equiv \text{expr} \Rightarrow g << h$$

Input (O, F) → G with G = {Reduce: expr}

Output g

The idea of this algorithm: this algorithm uses equivalent rules to simplify expression expr as simple as possible. We get the new expression, called *min*. After that, the expression *min* will be expanded and reduced the expansion expression, we get the expression *h*. If the new expression *h* is simpler than expression *min*, then record min by *h*. The processing to expand the expression is repeated at most β times (β is constant). The final result is expression g.

Constant β is chosen based on the sample space of common exercises in the knowledge domain. For instance, in the knowledge domain about vector algebra, the expressions can be simplified at most β times for the expansion. Thus, we choose β = 7.

Step 1 Setup initial values for variables.

//g is a result of simplification process of f

//min is an expression whose length is shortest in each step.

// D is a list of rules that deduce to the solution of the problem.

//Old_expr is a set of old expressions

```
g:= expr;
min:=g;
D:=[ ];
```

2.2 Simple reducing of temp_expr

```
h:= Redu_temp_expr;
Old_expr := Old_expr ⊔ {h};
if h << min then
   min := h;
   D:=D ∪ Simple_Reduce(temp_expr);
   count := 0;
else
   count := count +1;
end: # 2.2
g := h;
```

Old_expr := {expr};

Step 2

Use the heuristic rule to search the equivalent rules for transforming an expression.

Combine simple reducing and simple expansion until get shortest expression or count >β.

2.1 Simple expansion of g.

Old_expr := Old_expr ⊔ {g}

D:= D ∪ Simple_Expand(g);

if $Exp_g \notin$ Old_expr then

temp_expr := Exp_g;

else

goto Step 3.

end: # 2.1

Step 3

Go to step 2.

g := min;

From list D, build the solution of this problem.

return g;

4.3 Theorems

Lemma: given a knowledge domain ℜ = (C, Ops, Rules) as ops-model, and hypothesis (O, F) of a problem on this model. There exists an unique maximum set L(O, F) such that it contains all expression that can be deduced from (O, F).

Theorem 4.1: let a knowledge domain ℜ = (C, Ops, Rules) as model of operators, and a problem P = (O, F) → G on this model. The following statements are equivalent:

1 Problem P is solvable.

2 $G \subseteq L_{(O, F)}$

3 There exists a list D such that G ⊆ D(F)

This theorem shows that forward chaining reasoning will deduce to goals of problems. Besides, algorithms 4.1 and 4.2 were designed based on forward chaining reasoning, so this theorem is ensure the effectiveness of these algorithms.

Theorem 4.2: give the problem P = **(O, F) → G** in knowledge model of operators, with G = 'reduce: expr'. The complexity of algorithm 4.3 to solve this problem is:

$$O\left(n.(l+d)^2\right)$$

In which

l = length(expr)

n = number of equivalent rules in rules-set.

d = max{abs(length(left(r)) – length(right(r))) | r is an equivalent rule}

5 Applications

5.1 Design knowledge base of vector algebra

Based on knowledge about vector algebra in high school has been mentioned in textbook of (Vietnam Ministry of Education and Training, 2013), a part of this knowledge domain can be represented by Ops-model. This knowledge base has also been used to build the program for solving the problems about vector expressions automatically.

1 *C–set of concepts*: the set C consists of concepts such as 'point', 'vector', 'segment', 'triangle' and 'quadrangle'.

Eg. 4.1: $C_{(0)}$ = {ℜ, POINT, LINE}

$C_{(1)}$ = {SEGMENT, VECTOR}

Concept VECTOR ∈ $C_{(1)}$ has structure:

Attrs = {_A, _B, module}, which:

_A, _B: POINT

module: ℝ ;

EqObj = {module = Segment(_A,_B)}

RulesObj = { }

$C_{(2)}$ = {ANGLE, TRIANGLE and types of it, QUADRANGLE and types of it, CIRCLE, …}

Concept PARALLELOGRAM ∈ $C_{(2)}$ consists of:

Attrs = {A, B, C, D, a, b, c, d, S, p...}

A, B: POINT

a, b, c, d: SEGMENT

S, p: ℜ

EqObj = {Angle(A)+ Angle(B)+ Angle(C)+ Angle(D) = 360,

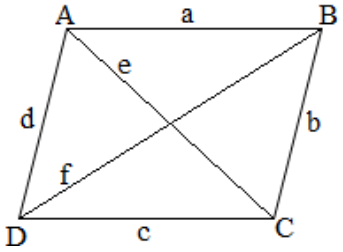
$\overrightarrow{AB} = \overrightarrow{DC}$, $\overrightarrow{AD} = \overrightarrow{BC}$,

$\overrightarrow{AC} = \overrightarrow{AB} + \overrightarrow{AD}$, $\overrightarrow{BD} = \overrightarrow{BA} + \overrightarrow{BC}$…}

RulesObj = {{a = b}→ {ABCD: RHOMBUS},

{e = f} → {ABCD: RECTANGLE}}

Figure 1 Parallelogram



2 *Ops–set of operators between concepts*

Ops-set includes these operators as followed:

Table 2 Operators in vector algebra

Operator	*Meaning*	*Arguments*	*Return*	*Properties*
+	Sum of vectors	Vector × Vector	Vector	Commutative associative identity
*	Product between a real number and a vector	ℝ × Vector	Vector	
.	Inner product of vectors	Vector × Vector	ℝ	Commutative
o	Cross product of vectors	Vector × Vector	Vector	

3 *Rules-set*: rules in this model are classified of two forms: deductive rules and equivalent rules.

a Some deductive rules in rules-set

R1: {AB: segment, M: point, M midpoint AB}

$$\rightarrow \left\{ \overrightarrow{MA} + \overrightarrow{MB} = \vec{0},\ \overrightarrow{AM} = \overrightarrow{MB} = \frac{1}{2}\overrightarrow{AB} \right\}$$

R2: {u, v: vector, u $\perp$ v} → {u.v = 0}

R3: {a,b,c: vector, c = a o b} → {c $\perp$ a, c $\perp$ b}

R4: {ABC: triangle, G: Point, G center of ABC} → $\{\overrightarrow{GA} + \overrightarrow{GB} + \overrightarrow{GC} = \vec{0}\}$

R5: {ABC: triangle, M: Point, N: Point, M is midpoint AB, N is midpoint AC}

$$\rightarrow \left\{ \overrightarrow{MN} = \frac{1}{2}\overrightarrow{BC} \right\}$$

b Some equivalent rules in rules-set

R6: A, B: Point, $\overrightarrow{AB} = -\overrightarrow{BA}$

R7: A, B, C: Point, $\overrightarrow{AB} + \overrightarrow{BC} = \overrightarrow{AC}$

R8: u: vector, $u^2 = u.u = (u.module)^2$

R9: u, v: vector, u.v = u.module * v.module * cos(u, v)

R10: u, v: vector, uov = –vou

5.2 Design the inference engine of system

Model of problem in this knowledge base is defined as definition 4.1. This inference engine can solve the practical problems with the solutions are naturally alike those of human. Besides the above algorithms in Section 4.2, the program has been integrated the heuristic rules for searching the solutions of the problems.

5.2.1 Heuristic rule about using sample problems

When dealing with a practical problem, a convenient way to proceed is considering whether we have met a similar or related problem before or not. If so, then the solution for the practical problem can be obtained effectively by using the results of the related problem. The related problems are called sample problems (Do et al., 2013).

In the knowledge domain about vector algebra, the transforming on common expressions has been uses as the sample problems in the processing of the expression. Via this transforming, the algorithms can speed up to search the solution of the problems. Some transforming for common expressions is as followed:

Eg. 4.2: some of sample problems have been used in this heuristic rule:

a (SP1): A, B, M: point

$$\overrightarrow{AB}^2 = \overrightarrow{AB}^2$$
$$= (\overrightarrow{AM} + \overrightarrow{MB})^2$$
$$= \overrightarrow{AM}^2 + \overrightarrow{MB}^2 + 2.\overrightarrow{AM}.\overrightarrow{MB}$$

b (SP2): u,v, t: vector, v + t = $\vec{0}$

$$u.v + u.t = u.(v + t)$$
$$= u.\cdot\vec{0}$$
$$= \vec{0}$$

5.2.2 Arrange the order of rules in priority

When dealing with a practical problem, this heuristic rule arranges the order of inference rules in knowledge base to apply. This arrangement will prioritise to apply the rules which related with the facts in the practical problem. By using this heuristic rule, the

processing of the program can omit the rules that do not need for solving the current problem.

For designing inference engine of this system, determination the list of rule can be done by these steps:

Input FactSet: set of currently facts in searching processing

Output ListRules: list of rules may be applied

Stage 1	*Choose the rules in rules-set may be applied*
	• Determine structure of rule r in rules-set
	• Based on FactSet and hypothesis of rule r, make facts database to store information about facts in hypothesis.
	• Using Sort Merge Joint to make a joint between facts database by the same columns.
	• If result of joint is not NULL then r may be applied, so record rule r into ListRules.
Stage 2	*Arrange the order of ListRules*
	• Sorting the rules in ListRules by:
	• Number of the facts in rule that related with the facts in FactSet.
	• If the rules are equivalent rules, sorting by the simpler (<<) relation
	• If the rules are deductive rules, the rules about midpoint, perpendicular or parallel are priority.

5.3 Testing and experiments

5.3.1 Result of testing

The program for solving problems about Vector Algebra has been tested to solve the exercises in the curriculum of the high-school mathematics in Vietnam (Vietnam Ministry of Education and Training, 2013). Via the structure of this knowledge base, the problems have been solved by using the transforming steps, replacement and deductive rules. The solutions are step-by-step and readable. The reasoning uses the knowledge of the student about this course. This program can solve some basic and advanced exercises. It is useful for the studying of the students in Vector Algebra at the high school.

Eg. 4.3: *(Prob. S1)* let triangle ABC and point G be a centre of this triangle. Let point M.
Prove: $MA^2 + MB^2 + MC^2 = GA^2 + GB^2 + GC^2 + 3MG^2$

1 Specification of problem:

- O:= {ABC: triangle, G: point, M: point}
- F:= {G centre of ABC}
- E:= $\{MA^2 + MB^2 + MC^2, GA^2 + GB^2 + GC^2 + 3MG^2\}$
- G:= Prove: $MA^2 + MB^2 + MC^2 = GA^2 + GB^2 + GC^2 + 3MG^2$

2 Solution of program:

$MA^2 + MB^2 + MC^2$

1	$\overrightarrow{MA}^2 + \overrightarrow{MB}^2 + \overrightarrow{MC}^2$	apply rule R8
2	$\left(\overrightarrow{MG} + \overrightarrow{GA}^2\right)^2 + \left(\overrightarrow{MG} + \overrightarrow{GB}\right)^2 + \left(\overrightarrow{MG} + \overrightarrow{GC}\right)^2$	apply rule R7
3	$3\overrightarrow{MG}^2 + \overrightarrow{GA}^2 + \overrightarrow{GB}^2 + \overrightarrow{GC}^2 + 2.\overrightarrow{MG}.\overrightarrow{GA} + 2.\overrightarrow{MG}.\overrightarrow{GB} + 2.\overrightarrow{MG}.\overrightarrow{GC}$	
4	$3\overrightarrow{MG}^2 + \overrightarrow{GA}^2 + \overrightarrow{GB}^2 + \overrightarrow{GC}^2 + 2.\overrightarrow{MG}.\left(\overrightarrow{GA} + \overrightarrow{GB} + \overrightarrow{GC}\right)$	
5	$3\overrightarrow{MG}^2 + \overrightarrow{GA}^2 + \overrightarrow{GB}^2 + \overrightarrow{GC}^2 + 2.\overrightarrow{MG}.\vec{0}$	apply rule R4
6	$3MG^2 + GA^2 + GB^2 + GC^2$	apply rule R8

In this example, steps 1–3 are applied the sample problem SP1 to expand the expression. After that, steps 4–6 are applied the sample problem SP2 to simplify the expression in step 3. Thus, this example can be solved fast.

Eg. 4.4: *(Prob. S2)* let parallelogram ABCD. Denote I is a midpoint of segment CD.

Transform: $\overrightarrow{BI}$ into an expression of $\overrightarrow{AB}$ and $\overrightarrow{AD}$

1 Specification of problem:

- O:= {ABCD: parallelogram, I: point}
- F:= {I midpoint CD}
- E := {}
- G:= Transform: $\overrightarrow{BI}$ by $\overrightarrow{AB}$, $\overrightarrow{AD}$

2 Solution of program:

1	$\overrightarrow{BI} = \overrightarrow{BA} + \overrightarrow{AD} + \overrightarrow{DI}$	apply rule R7
2	$\overrightarrow{BA} = -\overrightarrow{AB}$	apply rule R6
3	{DC:segment, I: point, I midpoint CD} $\rightarrow \left\{\overrightarrow{DI} = \frac{1}{2}\overrightarrow{DC}\right\}$	apply rule R1
4	$\overrightarrow{BI} + \overrightarrow{AB} = \overrightarrow{AD} + \frac{1}{2}\overrightarrow{DC}$	
5	{ABCD: parallelogram} $\rightarrow \{\overrightarrow{DC} = \overrightarrow{AB}\}$	properties of the parallelogram
6	$\overrightarrow{BI} = -\overrightarrow{AB} + \overrightarrow{AD} + \frac{1}{2}\overrightarrow{AB}$	
7	$\overrightarrow{BI} = -\frac{1}{2}\overrightarrow{AB} + \overrightarrow{AD}$	

5.3.2 Experiments

The exercises are collected from the workbook of Ministry of Education and Training (2013). They are classified to these kinds.

- Kind 1: reduce an expression of vectors.
- Kind 2: prove the equation between two expressions.
- Kind 3: compute the value of an expression.
- Kind 4: transform a vector into an expression between certain vectors.

With the other programs for solving vector problems, they only solve the problems about vector calculator. Symbolab (2017) can solve many kinds of problems in mathematics, but it only solves the vector problems about computing the value of a simple expression. (Woflfram|Alpha, 2017) can illustrate the results of vector calculator however they are not the solution of the problems for the students.

The results for solving these kinds of problems are as this followed table.

Table 3 The ability for solving problems of the systems

Problem		*Kind 1*	*Kind 2*	*Kind 3*	*Kind 4*	*Total*
Number of problems		*20*	*21*	*15*	*7*	*63*
Number of problems can be solved	Program for solving problems about vector algebra	17	15	13	6	51
	Symbolab	5	3	7	0	15
	Wolfram|Alpha	6	3	7	0	16

This program has been also examined by 161 students of three high-schools: two schools in Ho Chi Minh City and one school in Binh Duong province, Vietnam. This survey is also interested in four criteria: user-friendly interface, sufficient knowledge base, the ability to solve problems and usefulness.

Firstly, each student chooses four exercises in the exercises which can be solved by the program (51 problems), each kinds is one exercise. They receive the solutions of them from the program. Secondly, they are requested to input other three exercises by specification language, and the program shows the solutions of them. Based on these results, the students evaluate this program by four criteria with level from 1–5, respectively very bad–very good. The result of this survey is as followed:

Table 4 Result of the survey

Criterion	*Level (Very bad → very good)*				
	1	*2*	*3*	*4*	*5*
User-friendly interface		19%		81%	
Sufficient Knowledge		21%		79%	
Ability to solve problem		20%		80%	
Usefulness		19%		81%	

As the result of this survey, our program meets the requirements of an intelligent problem solver in education (VanLehn, 2006). It is useful for supporting high-school students to learn about vector algebra. The program has a sufficient knowledge base to solve the common problems in this knowledge domain. Its solutions like the solving method of

human. They include reasoning steps which are suitable with the knowledge's level of the students. Hence, the students can use this program for their studying.

6 Conclusions

In this paper, a mathematical structure of knowledge model of operators is presented, namely Ops-model. The foundation of this model has three components: concept, operators between concepts and rules. Each concept is an abstract structure that includes attributes, equation and deductive rules and objects in concept have been equipped behaviours to solve some problems on object. Ops-model is effective in formally representing the knowledge of operators. The model problems in kinds 1 and 2 have been studied. The algorithms are also designed to solve problems: problems of reducing an expression, proving an equality of expressions and transforming an object into a expression of certain objects. These algorithms are also proved their effectiveness.

Ops-model is useful tool for designing practical knowledge bases, especially knowledge domains of computing. It has been applied to specify a part of the knowledge domain about vector algebra in high school. The program for automatic solving some problems on this knowledge domain has been built, such as: reduce a vector expression, prove the equation between two vector expressions, compute the value of an expression and transform a vector into an expression between certain vectors. The system provides readable and human-alike solutions. These solutions can be understood by the high school students and their reasoning is suitable with the learner's level.

In the future, we will continue to complete the knowledge model of operators and solve some problems of operators such as: solving equations systems between objects. Besides, the combined model of ops-model and the knowledge model of relations (Nguyen et al., 2015) will be more suitable for real applications. It will become the foundation of general knowledge model.

Acknowledgements

This research is funded by Vietnam National University HoChiMinh City (VNU-HCM) under grant number C2016-26-06.

References

Anton, H. and Rorres, C. (2010) *Elementary Linear Algebra*, 10th ed., John Wiley & Sons.

Cordero, P., Gutiérrez, G., Martínez, J. and Guzmán, I.P. (2004) 'A new algebraic tool for automatic theorem provers', *Annals of Mathematics and Artificial Intelligence*, Vol. 42, No. 4, pp.369–398.

Do, N.V. (2015) 'Ontology COKB for knowledge representation and reasoning in designing knowledge-based systems', *Communications in Computer and Information Science (CCIS)*, Springer, Vol. 513, pp.101–118.

Do, N.V. and Nguyen, H.D. (2015) 'Reducing model of COKB about operators knowledge and solving problems about operators', in Camacho, D. et al. (Eds.): *New Trends in Computational Collective Intelligence*, Studies in Computational Intelligence 572, Springer, pp.39–49.

Do, N.V., Nguyen, H.D. and Mai, T.T. (2013) 'Designing an intelligent problems solving system based on knowledge about sample problems', *Proceeding of 5th Asian Conference on Intelligent Information and Database Systems (ACIIDS 2013)*, March 2013, pp.465–475, LNAI 7802, Springer, Kuala Lumpur, Malaysia.

Fu, H., Zhong, X. and Zeng, Z. (2006) 'Automated and readable simplification of trigonometric expressions', *Mathematical and Computer Modeling*, Vol. 44, Nos. 11–12, pp.1169–1177.

Grefenstette, E. (2013) 'Towards a formal distributional semantics: simulating logical calculi with tensors', *Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 1: Proceedings of the Main Conferene and the Shared Task*, pages 1–10, 13–14 June 2013, Atlanta, Georgia.

Harmelem, F., Lifschitz, V. and Porter, B. (2008) *Handbook of Knowledge Representation*, Elsevier, Amsterdam.

Kent, R.E. (2013) 'The first-order logical environment', Pfeiffer, H.D. et al. (Eds.): *Conceptual Structures in Research and Education, LNCS*, Vol. 7735, pp.210–230, Springer-Verlag Berlin Heidelberg.

Knyazhansky, M. (2012) 'Knowledge bases over algebraic models: some notes about information equivalence', *International Journal of Knowledge Management*, Vol. 8, No. 1, pp.22–39, ISSN: 1548-0666.

Kuphaldt, T.R. (2017) *Direct Current*, Vol. 1 [online] https://www.allaboutcircuits.com/textbook/ (accessed 31 October 2017).

Nguyen, H.D, Pham, V.T., Le, T.T. and Tran, D.H (2015) 'A mathematical approach for representation knowledge about relations and its application', in *KSE 2015: Proceeding of 2015 IEEE International Conference on Knowledge and Systems Engineering*, pp.324-327, Ho Chi Minh, Vietnam, ISBN: 978-1-4673-8013-3.

Roanes-Lozano, E., Laita, L.M., Hernando, A. and Roanes-Macias, E. (2009) 'A Groebner bases-based approach to backward reasoning in rule based expert systems', *Annals of Mathematics and Artificial Intelligence*, Vol. 56, Nos. 3–4, pp.297–311, Springer.

Sakama, C., Inoue, K. and Sato, T. (2017) 'Linear algebraic characterization of logic programs', *Proceeding of 10th International Conference on Knowledge Science, Engineering and Management (KSEM 2017)* , August 2017, LNAI 10412, Springer, Melbourne, Australia.

Symbolab (2017) *https://www.symbolab.com/solver/vector-calculator* (accessed 28 August 2017).

Tulceanu, V. (2016) 'Consideration regarding an algebraic model for inference and decision on heterogeneous sensory input', *Soft Computing Applications, Advances in Intelligent Systems and Computing 356*, pp.539–548, Springer.

Valipour, M. and Wang, Y. (2016), 'Formal properties and mathematical rules of concept algebra for cognitive machine learning', *Journal of Advanced Mathematics and Application*, Vol. 5, No. 1, pp.69–86, American Scientific Publishers.

VanLehn, K. (2006) 'The behavior of tutoring system', *International Journal of Artificial Intelligence in Education*, Vol. 16, No. 3, pp.227–265.

Varberg, D., Purcell, E. and Rigdon, S. (2003) *Calculus: Chapter 13 – Vector Algebra*, 9th ed., Prentice-Hall.

Vietnam Ministry of Education and Training (2013) *Textbook and Workbook of High school Mathematics*, Publisher of Education, Vietnam.

Wang, Y. (2015) 'Concept algebra: a denotational mathematics for formal knowledge representation and cognitive robot learning', *Journal of Advanced Mathematics and Application*, Vol. 4, No. 4, pp.61–86, American Scientific Publishers.

Wolfram|Alpha (2017) *https://www.wolframalpha.com/examples/Vectors.html* (accessed 28 August 2017).

Yang, C. and Cai W. (2008) 'Knowledge representations based on extension rules', *Proceedings of the 7th World Congress on Intelligent Control and Automation*, Chongqing, China.