Abstract

We introduce object-oriented information system into rough set models. To accomplish this, we construct the class, object, and name structures using rough set theory. Next, combining class, name and object structures, we propose object-oriented information systems. Moreover, we introduce indiscernibility relations on the set of objects, lower and upper approximations, and object-oriented rough sets in the object-oriented information systems and we introduce the concept of object oriented reducts, object oriented reducts using indiscernibility matrix and algorithms for object oriented reducts.

Keywords: rough set, object-orientation, discernibility matrix, reducts

1. Introduction

Rough set theory ([1,2,3]), proposed by Z. Pawlak in 1982, offered an effective mathematical method to deal with uncertainty knowledge. Recently, rough set theory and its application have been developed rapidly, which are mainly concentrated on the generalization of rough set model, the research on uncertainty theory in rough set, rough set operations and their connections with other uncertainty operations, rough set and its contacts with other mathematical theories and so on. In this paper, we introduce the object-oriented paradigm[4] to the rough set theory, and propose object-oriented rough set models.

The rest of this paper is organized as follows. In Section 2, we briefly review rough set theory. In Section 3, we constructed object oriented rough set models with the help of object-oriented information systems. In Section 4, we define indiscernibility relations of objects that reflect hierarchical structures between objects. Moreover, we propose lower and upper approximations, and
object-oriented rough sets based on object-oriented information systems. In section 5 we developed algorithms for object oriented reducts and section 6 summarizes contributions of this paper.

2. Rough Sets and Information Systems

An information system is a pair \( S = (U,A) \) where \( U \) and \( A \) are finite and nonempty sets. \( U \) is called the universe, and each element \( x \in U \) is called an object, respectively. On the other hand, each element \( a \in A \) is called an attribute, which is identified with a function \( a: U \rightarrow v_a \) that assigns a value to each object \( x \in U \), where \( v_a \) is the set of values of the function \( a \).

For any subset \( B \subseteq A \) of attributes, we construct an indiscernibility relation \( R_B \) on \( U \) as follows:

\[
x R_B y \iff a(x) = a(y), \quad \forall \ a \in B
\]

where \( a(x) \) means the value of the object \( x \in U \) at the attribute \( a \). \( x R_B y \) means that we can not discern \( x \) and \( y \) by any combination of attributes in \( B \). It is clear that the indiscernibility relation \( R_B \) is an equivalence relation. We denote the equivalence class by \( R_B \) that contain \( x \) as \( [x]_{R_B} \). The class of all equivalence classes by \( R_B \) provides a partition \( U/R_B \) of \( U \).

For a given information system \( S=(U,A) \), a given subset \( B \subseteq A \) of attributes, and any subset \( X \subseteq U \), we construct a lower approximation \( R_B(X) \) and an upper approximation \( \overline{R_B}(X) \) of \( X \) as follows, respectively:

\[
R_B(X) = \{x \in U / [x]_B \subseteq X\}
\]

\[
\overline{R_B}(X) = \{x \in U / [x]_B \cap X \neq \emptyset\}
\]

The lower approximation of \( X \) is the set of objects \( x \) that the equivalence class \( [x]_{R_B} \) of \( x \) is included to \( X \), and the upper approximation of \( X \) is the set of objects \( x \) that \( [x]_{R_B} \) has a non-empty intersection with \( X \). Note that we have the following set-inclusion relation: \( R_B(X) \subseteq X \subseteq \overline{R_B}(X) \).

A rough set of \( X \) is a pair \( R(X) = (R_B(X), \overline{R_B}(X)) \) of the lower approximation and the upper approximation of \( X \). The rough set \( R(X) \) provides an approximation of the set \( X \) in the information system \( S \) based on attributes in \( B \). If we have \( R_B(X) = X = \overline{R_B}(X) \), \( X \) is called \( R_B \)-definable. On the other hand, if we have \( R_B(X) \subseteq X \subseteq \overline{R_B}(X) \), \( X \) is called \( R_B \)-rough.

Quality of approximation of \( X \) by the rough set \( R_B(X) \) is numerically evaluated as follows:

\[
\frac{|R_B(X)|}{|\overline{R_B}(X)|}
\]
Reducts in object oriented information system

where, for any set $S$, $|S|$ means the cardinality of $S$. It is clear that the quality of approximation is equal to 1 if and only if $X$ is $R_B$-definable.

Let $P$ and $Q$ be equivalence relations over $U$, then the positive, negative and boundary regions are defined as:

$$\text{POS}_P(Q) = \bigcup_{x \in U/Q} \sim_P(X)$$

$$\text{NEG}_P(Q) = U - \bigcup_{x \in U/Q} \sim_P(X)$$

$$\text{BND}_P(Q) = \bigcup_{x \in U/Q} \sim_P(X) - \bigcup_{x \in U/Q} \sim_P(X)$$

The positive region comprises all objects of $U$ that can be classified to classes of $U/Q$ using the information contained within attributes $P$. The boundary region $\text{BND}_P(Q)$, is the set of objects that can possibly, but not certainly, be classified in this way. The negative region, $\text{NEG}_P(Q)$, is the set of objects that can not be classified to classes of $U/Q$.

Let $S = (U,C,D)$ be information system where $C$ is set of condition attributes and $D$ is set of decision attributes. The set of attributes $R \subseteq C$ is called a reduct of $C$, if $S' = (U,R,D)$ is independent and $\text{POS}_R(D) = \text{POS}_C(D)$.

The set of all the condition attributes indispensable in $T$ is denoted by $\text{CORE}(C) = \bigcap \text{RED}(C)$ where $\text{RED}(C)$ is the set of all reducts of $C$.

3. Object-Oriented Rough Set Models

In this section, we propose object-oriented information systems that illustrate hierarchical structures of object oriented concepts. First, we propose class structures that represent abstract data forms and hierarchical structures between classes. Next, we define object structures that illustrate many kinds of objects and actual dependence among objects by has-a relationship and offers-a relationship.

Moreover, we define name structures that introduce strict constraint to guarantee consistency of structures. Name structures provide concrete design of objects, and connect the class structure and the object structure consistently. Finally, combining these structures, we provide object oriented information systems as generalization of “traditional “information systems of rough set theory.

3.1. Class

Definition 1: A class structure $C$ is the following triple:

$$(C, R_c, S_c) --- (6)$$

where $C$ is finite non-empty set, $R_c$ is acyclic binary relation on $C$ that is $R_c$ satisfies the following property:

$c_1, c_2, \ldots, c_n \in C$ such that $c_1 R_c c_2, c_2 R_c c_3, \ldots, c_{n-1} R_c c_n, c_n R_c c_1$

and $S_c$ is a reflexive, transitive, and asymmetric binary relation on $C$. Moreover, $C_R$ and $C_S$ satisfying the following property:
∀ c_i, c_j, c_k ∈ C , c_i S c c_j , c_j R c c_k ⇒ c_j R c c_k .... (7)

Each c ∈ C is called a class that represents an abstract data form. Note that each class corresponds to a sort in many-sorted logic [5] and order-sorted logic [6].

Two relations R_c and S_c illustrate hierarchical structures among classes. The relation R_c is called a offers-a relation, which illustrates part / whole relationship between classes. c_i R_c c_j means “c_i offers a c_j”. The relation S_c is called a has -a relation, and c_i S_c c_j means that “c_i has a c_j” or c_i is a part of c_j.

Because C is a finite non-empty set, and R_c is acyclic, there is at least one class c such that c has no other class c’, that is, c R_c c’ for any c’ ∈ C. We call such class c an attribute, and denote the set of attributes by AT . Formally, AT is defined as follows:

\[ AT = \{ c ∈ C / c R_c c’, ∀ c’ ∈ C \} \] \hspace{2cm} (8)

Example 1: Let \( \mathcal{C} = (C, R_C, S_C) \) be class structure with C={University,College,Department,Faculty,Student,Course,Ncollege,Ndept,Nstudents} and have the following relations.

has-a relation: University S_c College,
College S_c Department,
University S_c Department,

Offers- a relation: College R_c Department
Department R_c Courses.

Suppose moreover that Ncollege, Ndept and Nstudents are attributes.

By the Property(6) these relations illustrate connection between classes, for example, “University has a College” and “College offers Department “ imply “University offers department”

(or)

“University has a Department ” and “Department offers Course” imply “University offers Course ”

3.2. Object

We define an object structure that illustrates hierarchical structures among objects.

Definition 2: An object structure O is the following triple:

\[ (O, R_o, S_o) \] \hspace{2cm} (9)

where O is a finite non-empty set, R_o is an acyclic binary relation on O, and S_o is a reflexive, transitive, and asymmetric binary relation on O. Moreover, similar to the definition of class, R_c, and S_c satisfy the following property:

\[ ∀ o_i, o_j, o_k ∈ O, o_i S_o o_j , o_j R_o o_k ⇒ o_i R_o o_k \] \hspace{2cm} (10)

We intend that every object o ∈ O is an instance of some class c ∈ C. To represent this intention, we define a class identifier function id_C as follows.

Definition 3: Let \( \mathcal{C} = (C, R_C, S_C) \) be the class structure and \( \mathcal{O} = (O, R_o, S_o) \) be the object structure. A function id_C:O→C is called class identifier iff id_c a p-morphism between \( \mathcal{O} \) and \( \mathcal{C} \)(cf.[8],p142) that is, the function id_C satisfies the following conditions:
1. $\forall o_i, o_j \in O, o_i R_o o_j \Rightarrow id_C(o_i) R_c id_c(o_j) \quad \ldots \quad (11)$

2. $\forall o_i \in O, \exists c \in C, id_C(o_i) R_c c \Rightarrow \exists o_j \in O \text{ s.t. } o_i R_o o_j \text{ and } id_c(o_j) = c \ldots \quad (12)$

and the same conditions are also satisfied for $S_o$ and $S_c$. $id_C(o) = c$ means that the object $o$ is an instance of the class $c$.

For any object $x$, if $id_c(x) = a$ and $a \in AT$, we call such object $x$ a value object of the attribute $a$. The value object $x$ is an instance of the attribute $a$ represents a “value” of the attribute. Thus, if $y$ is another value object of $a$, it is natural to enable us to compare the “value” of $x$ and $y$. We introduce the concept of “value” of value objects.

**Definition 4:** For any object $x$, if $id_c(x) = a$ and $a \in AT$, we call such object $x$ a value object of the attribute $a$. We denote the “value” of the value object $x$ by $Val(x)$.

### 3.3. Name

We introduce a name structure to provide concrete design of objects, and connect the class structure and the object structure consistently. The class structure provides abstract data forms of objects, however, does not provide constraints about the number of parts and their identification. Suppose we have $c_i R_c c_j$ and we intend that any instance $o_i$ of the class $c_i$ has $m$ objects of $c_j$ as parts of $o_i$ and each object of $c_j$ should be strictly identified. Direct connection between objects and classes by the class identifier $id_C$.

**Definition 5:** Let $C = (C, R_c, S_c)$ be the class structure. A name structure $N$ for $C$ is the following triple:

$$(N, R_N, S_N) \quad \ldots \quad (13)$$

where $N$ is a finite non-empty set such that $|C| \leq |N|$, $R_N$ is an acyclic binary relation on $N$, and $S_N$ is a reflexive, transitive, and asymmetric binary relation on $N$. Moreover, similar to the definition of class, $R_N$ and $S_N$ satisfy the following property:

$\forall n_i, n_j, n_k \in N, n_i S_N n_j, n_j R_N n_k \Rightarrow n_i R_N n_k \quad \ldots \quad (14)$

We call each $n \in N$ a name.

We intend that a naming function $f_n : N \rightarrow C$ provides names to each class. To introduce the naming function precisely, we define the following notations.

**Definition 6:** Let $C = (C, R_c, S_c)$ be the class structure, $N = (N, R_N, S_N)$ be the name structure, and $f : N \rightarrow C$ be a function. For any name $n \in N$, we denote the set of names that $n$ has by:

$H_N(n) = \{ n_j \in N / n R_N n_j \}$

Moreover, using the function $f$, we denote the set of names of a class $c \in C$ that $n$ has by

$H_n^f(c/n) = \{ n_j \in N / n R_N n_j, f(n_j) = c \}$

**Definition 7:** Let $C = (C, R_c, S_c)$ be the class structure, $N = (N, R_N, S_N)$ be the name structure. A function $f_n : N \rightarrow C$ is called a naming function if and only if $f_n$ is a surjective $p$-morphism between $N$ and $C$ and satisfies the following name preservation constraint:
For any \( n_i, n_j \in \mathbb{N} \), if \( f_n(n_i) = f_n(n_j) \) then
\[
H^*_n(c/n_i) = H^*_n(c/n_j)
\]
is satisfied for all \( c \in C \).

**Example 2:** This example is continuation of Example 1. Let \( C = (C, R_C, S_C) \) be the class structure in Example 1, \( N = (N, R_N, S_N) \) is a name structure with \( N = \{\text{university, college, department, faculty, student, college2, course, ncollege, ndepartment, nstudents}\} \) and the following relationships:

- Has-a relation: 
  - university \( S_N \) college,
  - college \( S_N \) department,
  - university \( S_N \) Department,
  - ………

- Offers-relation:
  - College \( R_N \) Department
  - Department \( R_N \) Courses.

Moreover, suppose we have a naming function \( f_n : N \rightarrow C \) such that
\[
\begin{align*}
 f_n(\text{university}) &= \text{University}, \\
f_n(\text{college}) &= f_n(\text{college2}) = \text{College}, \\
f_n(\text{department}) &= \text{Department}, \\
f_n(\text{faculty}) &= \text{Faculty}, \\
f_n(\text{student}) &= \text{Student}, \\
f_n(\text{course}) &= \text{Course}, \\
f_n(\text{ncollege}) &= \text{Ncollege}, \\
f_n(\text{ndepartment}) &= \text{Ndepartment}, \\
f_n(\text{nstudent}) &= \text{Nstudent}.
\end{align*}
\]

Note that we have \( H_N(\text{College/}\text{university}) = \{\text{college, college2}\} \), and
\( H_N(\text{Ndepartment/ college}) = H_N(\text{Ndepartment/ college2}) = \{\text{ndepartment}\} \).

Here, to illustrate connection between the classes and names, we use class diagrams of UML[8] authorized by OMG[9] as in Fig 1. For example, the class diagram “University” illustrates that University class has two objects of the College class, called “college” and “college2”, respectively, one object “student” of the Student class, and one object “faculty” of the Faculty class.
Definition 8: Let \( O = (O, R_O, S_o) \) be the object structure and \( N = (N, R_N, S_N) \) be the name structure. A function \( a_n : O \to N \) is called a name assignment if and only if \( a_n \) is a \( p \)-morphism between \( O \) and \( N \) satisfies the following uniqueness condition:

For any \( x \in O \), if \( H_O(x) \neq \phi \), the restriction of \( a_n \) into \( H_O(x) \):

\[ a_n/_{H_O(x)} = H_O(x) \to N \]

is injective, where \( H_O(x) = \{ y \in O / x \overset{R_O}{\Rightarrow} y \} \) is the set of objects that \( x \) has \( a_n(x) = n \) means that the name of the object \( x \) is \( n \).

Definition 9: Let \( C = (C, R_C, S_C) \) be the class structure, \( N = (N, R_N, S_N) \) be the name structure, \( O = (O, R_O, S_o) \) be the object structure. Moreover, let \( \text{id}_C : O \to C \) be the class identifier. We say that \( C, N \) and \( O \) are well defined if and only if there exists a naming function \( f_n : N \to C \) and a name assignment \( a_n : O \to N \) such that

\[ \text{id}_C = f_n \circ a_n \]

that is, \( \text{id}_C (x) = f_n (a_n(x)) \) for all \( x \in O \).

Definition 10: Let \( C, N \) and \( O \) be well defined structures. Suppose we have \( o_1, o_2, \ldots, o_k \in O, n_1, n_2, \ldots, n_k \in N \) and \( c_1, c_2, \ldots, c_k \in C \) such that \( o_i \overset{R_O}{\Rightarrow} o_{i+1} \) for \( 1 \leq i \leq k-1 \), and \( a_n(o_i) = n_i, f_n(n_i) = c_i \) for \( 1 \leq i \leq k \). We denote \( o_1, n_2, \ldots, n_i \) instead of \( o_i \) for \( 2 \leq i \leq k \) by means of "the instance of \( c_i \) named \( n_i \) as a part of the instance of \( c_i \) as a part of \( o_1 \)."

Example 3: This example is continuous of Example 2. Let \( C = (C, R_C, S_C) \) and \( N = (N, R_N, S_N) \) are the same class structure and name structure in Example 2, respectively. Moreover, let \( O = (O, R_O, S_o) \) be an object structure with the offers -- a relationship illustrated in Fig 2 and the following has-a relationship.
\( x \subseteq O \), \( \forall x \in O \) and 

university3 \( \subseteq O \), university1 \( \subseteq O \), university3 \( \subseteq O \), university2.

Moreover, let \( \alpha_n : O \to N \) be the following name assignment:

\[
\begin{align*}
\alpha_n(\text{university1}) &= \alpha_n(\text{university2}) = \text{college}, \\
\alpha_n(\text{university3}) &= \text{college2}, \\
\alpha_n(\text{c1}) &= \text{college}, \\
\alpha_n(\text{c2}) &= \text{college}, \\
\alpha_n(\text{c3}) &= \text{college}, \\
\alpha_n(\text{c4}) &= \text{college2}, \\
\alpha_n(\text{s1}) &= \text{student}, \\
\alpha_n(\text{f1}) &= \text{student}, \\
\alpha_n(\text{f2}) &= \text{faculty}, \\
\alpha_n(\text{f3}) &= \text{faculty}, \\
\alpha_n(\text{s2}) &= \text{student}, \\
\alpha_n(\text{f4}) &= \text{faculty}, \\
\alpha_n(\text{f5}) &= \text{faculty}, \\
\alpha_n(\text{n1}) &= \text{ncollege}, \\
\alpha_n(\text{n2}) &= \text{nstudent}.
\end{align*}
\]

We define the class identifier \( \text{id}_C : O \to C \) by Eq.(19) using \( \alpha_n \) and \( f_n \) used in example 2. It is not hard to check that \( C, N \) and \( O \) are well defined.

![Fig 2. Offers- a relation on objects in example3](image-url)
Object-Oriented Information System

Using well defined class, name structure and object structures, we introduce an object oriented information system that corresponds to the information in “traditional” roughset theory.

Definition 11: Let \( C = (C, R_C, S_C) \) and \( N = (N, R_N, S_N) \), \( O = (O, R_o, S_o) \) be well defined class, name, object structures respectively. An object oriented information system \( OOIS(O, C, N) \) is the following structure:

\[
OOIS(O, C, N) = (O, C, N, o, id_c) \tag{20}
\]

where \( id_c = f_n \circ a_o \)

The object oriented information system can be illustrate “traditional” information system as special case. In particular, for any information system \( IS(U, A) \), we can construct an object oriented information system \( OOIS(O_{IS}, C_{IS}, N_{IS}) \) that corresponds to \( IS \): First, using the information system \( IS = (U, A) \), we construct a name structure.

\[
N_{IS} = (N_{IS}, R_{N_{IS}}, S_{N_{IS}})
\]

as follows:

\[
N_{IS} = A \cup \{IS\}
\]

\[
R_{N_{IS}} = \{(s, a) / a \in A\}
\]

\[
S_{N_{IS}} = \{(n, n) / n \in N_{IS}\}
\]

Where \( s \) is a symbol that doesn’t appear in \( A \). We also construct and object structure

\[
O_{IS} = (O_{IS}, R_{O_{IS}}, S_{O_{IS}})
\]

as follows:

\[
O_{IS} = U \cup (\bigcup_{a \in A} \{v^a_x \exists a, \exists x, v \in v, a(x) = v\})
\]

\[
R_{O_{IS}} = \{(x, v^a_x) / x \in U\}
\]

\[
S_{O_{IS}} = \{(o, o) / o \in O_{IS}\}
\]

Where \( v^a_x \) is a new symbol that corresponds to the value of the object of the attribute \( a \) as a part of the object \( x \), and \( v(v^a_x) = v \).

We set a class structure \( C_{IS} = (C_{IS}, R_{C_{IS}}, S_{C_{IS}}) \) as \( C_{IS} = N_{IS} \), \( R_{C_{IS}} = R_{N_{IS}} \) and \( S_{C_{IS}} = S_{N_{IS}} \).

Finally, we construct a name assignment \( (a_n)_{N_{IS}} \), a naming function \( (f_n)_{C_{IS}} \), and a class identifier \( id_{C_{IS}} \), respectively. Suppose a function \( (a_n)_{N_{IS}} : O_{IS} \rightarrow N_{IS} \) by

\[
(a_n)_{N_{IS}}(o) = \begin{cases} 
  s & \text{if } o \in U \\
  a & \text{if } o \in v^a_x, \exists x \in U
\end{cases}
\]

The function \( (a_n)_{N_{IS}} \) becomes a name assignment: if \( o \in U \), then we have

\[
H_O(o) = \{v^a_o / a \in A\},
\]

that is, the set of value objects about \( o \), and by the construction of value objects \( v^a_o \), each \( v^a_o \) and \( a = (a_n)_{N_{IS}} v^a_o \in N_{IS} \) corresponds
one to one. Otherwise, we have \( o = v^x \), and therefore \( H_0(o) = \phi \). We define the naming function \( (f_n)_{C_\alpha} : N_{IS} \rightarrow C_{IS} \) by \( (f_n)_{C_\alpha}(n) = n \in C_{IS} \) for all \( n \in N_{IS} \).

Using \( (a_n)_{N_{IS}} \), we get \( id_{C_\alpha} = (f_n)_{C_\alpha} o (a_n)_{N_{IS}} \).

OOIS \((O_{IS}, C_{IS}, N_{IS})\) satisfies the following property: \( a(x) = v \iff \text{val}(x,a) = v \), \( \forall x \in U \), \( \forall a \in A \).

4. Indiscernability Relations and object orient Rough sets

4.1 Equivalence as instance

We provide indiscernability relation on \( O \) based on structural aspects of objects. At first, we introduce an equivalence relation that illustrates “equivalence as instances of a class”. To evaluate equivalence of instances, we use names of parts of instances as follows.

**Definition 12:** Let \( OOIS (O,C,N) \) be the object oriented information system. We define a binary relation \( \sim \) on \( O \) recursively as follows:

\[
\begin{align*}
x \sim y & \iff x \text{ and } y \text{ satisfy the following two conditions:} \\
1. & \quad \text{id}_C(x) = \text{id}_C(y) \quad \text{and} \\
2. & \quad \text{val}(x) = \text{val}(y)
\end{align*}
\]

where \( H_N(a_n(x)) \) is the set of names that \( a_n(x) \) has, and defined by Eq.(15).

**Example 4:** This example is continuation of Example3. Let \( OOIS (O,C,N) \) be the object-oriented information system based on the class structure \( C \), the object structure \( O \) and the name structure \( N \) used in example 3. We construct the equivalence relation \( \sim \) on \( O \) by Eq.(20). Equivalence classes by \( \sim \) are as follows:

\[
\begin{align*}
\text{[university1]} & \sim \{\text{university 1}\}, \\
\text{[university2]} & \sim \{\text{university 2}\}, \\
\text{[university3]} & \sim \{\text{university 3}\}, \\
\text{[s1]} & \sim \{\text{s1,s2,s4}\}, \\
\text{[s2]} & \sim \{\text{s2}\}, \\
\text{[d1]} & \sim \{\text{d1}\}, \\
\text{[d2]} & \sim \{\text{d1,d2}\}, \\
\text{[c1]} & \sim \{\text{c1,c2}\}, \\
\text{[c3]} & \sim \{\text{c3}\}, \\
\text{[24]} & \sim \{\text{university1.college.dept,university3.college.dept,university3.college2.dept}\}, \\
\text{[16]} & \sim \{\text{university2.college2.dept}\}.
\end{align*}
\]

Note that \( a_n(c1) = a_n(c3) = a_n(c4) = \text{college} \), \( a_n(c4) = \text{college2} \) and \( f_n(\text{college}) = f_n(\text{college2}) = \text{college} \). By the name preservation constraint, we have \( H_N(N\text{department/college}) = H_N(N\text{department/college2}) = \{N\text{department}\} \) as illustrated in example 2. Therefore, for example, we can evaluate the equivalence of \( c1 = \text{university.college} \) and \( c4 = \text{university3.college2} \) by evaluating the equivalence of \( \text{university1.college.department} \) and \( \text{university3.college2.department} \).
4.2. Object-Oriented Rough Sets

Definition 13: Let OOIS (\(O, C, N\)) be the object oriented information system, and \(\sim\) be the equivalence relation defined by Eq.(20). For any non-empty subset \(B \subseteq N\), we define a binary equivalence relation \(\sim_B\) on \(O\) as follows:

\[
\begin{align*}
1. & \quad B \cap H_N(a_N(x)) = B \cap H_N(a_N(y)) \quad \text{and}, \\
2. & \quad \forall n [ n \in B \cap H_N(a_N(x)) \Rightarrow x.n \sim y.n ]
\end{align*}
\]

(22)

\[
\text{Definition 14: Let OOIS (O, C, N) be an object oriented information system, B} \\
\text{\(\subseteq N\) is a non-empty subset of names, and \(\sim_B\) be the equivalence relation defined} \\
\text{by Eq.(22). For any subset } X \subseteq O \text{ of objects, the lower approximation} \\
\text{\(\overline{\sim}_B(X)\) and upper approximation} \\
\text{\(\overline{\sim}_B(X)\) of } X \text{ by } \sim_B \text{ are defined as follows, respectively:} \\
\text{\(\overline{\sim}_B(X) = \{ x \in O / [x]_{-B} \subseteq X \}\) ...} \\
\text{\(\overline{\sim}_B(X) = \{ x \in O / [x]_{-B} \cap X \neq \emptyset \}\) ...}
\]

Moreover, the object-oriented rough set \(\sim_B(X)\) of \(X\) by \(\sim_B\) is the following pair:

\[ (\overline{\sim}_B(X), \underleftarrow{\sim}_B(X)) \]

(25)

- \(X\) is roughly \(\sim_B\)-definable, if and only if \(\overline{\sim}_B(X) \neq \emptyset\) and \(\underleftarrow{\sim}_B(X) \neq \emptyset\).
- \(X\) is internally \(\sim_B\) undefinable, if and only if \(\overline{\sim}_B(X) = \emptyset\) and \(\underleftarrow{\sim}_B(X) \neq \emptyset\).
- \(X\) is externally \(\sim_B\) – undefinable, if and only if \(\overline{\sim}_B(X) = \emptyset\) and \(\underleftarrow{\sim}_B(X) = U\).
- \(X\) is totally \(\sim_B\) – undefinable, if and only if \(\overline{\sim}_B(X) = \emptyset\) and \(\underleftarrow{\sim}_B(X) = U\).

The quality of approximation of \(X\) by the rough set \(\sim_B\) is defined by

\[
\frac{|\overline{\sim}_B(X)|}{|\overline{\sim}_B(X)|}
\]

and some properties of approximations are as follows.

1. \(\overline{\sim}_B(X) \subseteq X \subseteq \overline{\sim}_B(X)\)
2. \(\overline{\sim}_B(\emptyset) = \underleftarrow{\sim}_B(\emptyset) = \emptyset\), \(\overline{\sim}_B(O) = \underleftarrow{\sim}_B(O) = O\).
3. \(\overline{\sim}_B(X \cup Y) = \overline{\sim}_B(X) \cup \overline{\sim}_B(Y)\)
4. \(\overline{\sim}_B(X \cap Y) = \overline{\sim}_B(X) \cap \overline{\sim}_B(Y)\).
5. \(X \subseteq Y\) implies \(\overline{\sim}_B(X) \subseteq \overline{\sim}_B(Y)\) and \(\underleftarrow{\sim}_B(X) \subseteq \underleftarrow{\sim}_B(Y)\)
6. \(\overline{\sim}_B(X \cup Y) \supseteq \overline{\sim}_B(X) \cup \overline{\sim}_B(Y)\)
7. \(\overline{\sim}_B(X \cap Y) \subseteq \overline{\sim}_B(X) \cap \overline{\sim}_B(Y)\)
8. \(\overline{\sim}_B(\sim X) = \sim \overline{\sim}_B(X)\)
9. \(\overline{\sim}_B(\sim X) = \sim \overline{\sim}_B(X)\)
10. $\sim_B(\sim_B(X)) = \sim_B(\sim_B(X)) = \sim_B(X)$

11. $\sim_B(\sim_B(X)) = \sim_B(\sim_B(X)) = \sim_B(X)$

where $-X$ denotes $O-X$.

**Definition 15**: Let $O = (O, R_O, S_O)$ be the object structure and $D = \{d_1, d_2, \ldots, d_n\}$ be the set of decision attribute values and $d \notin AT$ where $AT$ is set of all condition attributes. A function $g_n : O \rightarrow D$ is called decision function if and only if $g_n$ is a surjective $p$-morphism and satisfies the following constraint:

$$\land c_i \Rightarrow d_j \quad \text{where} \quad c_i \in AT \quad \text{and} \quad d_j \in D \quad (i=1, 2, \ldots, |AT|) \quad \ldots \quad (26)$$

**Example 5**: This example is continuation of Example 4. Suppose $B = \{\text{college}\}$, using the equivalence relation $\sim$ constructed in example 4, we construct the equivalence relation $\sim_B$ and equivalence classes by $\sim_B$ are as follows:

- $[\text{university1}]_{\sim_B} = \{\text{university1, university3}\}$
- $[\text{university2}]_{\sim_B} = \{\text{university2}\}$
- $[c1]_{\sim_B} = O - \{\text{university1, university2, university3}\}$

The equivalence classes $[\text{university1}]_{\sim_B}$ correspond to the set of objects with the “24 college” and $[\text{university2}]_{\sim_B}$ is the set of objects with “16 colleges”. On the other $[c1]_{\sim_B}$ represents the set of objects that has no instance of the College class. Note that university1 and university3 are in the same university class $[\text{university1}]_{\sim_B}$ even though $id_C(\text{university1}) \neq id_C(\text{university3})$.

Let $X = \{\text{university2, university3}\}$. The lower approximation $\sim_B(X)$ and upper approximation $\sim_B(X)$ of $X$ by $\sim_B$ are constructed as follows, respectively.

$$\sim_B(X) = \{x \in O : [x]_{\sim_B} \subseteq X\} = \{\text{university2}\},$$

$$\sim_B(X) = \{x \in O : [x]_{\sim_B} \cap X \neq \emptyset\} = \{\text{university1, university2, university3}\},$$

Thus object oriented rough set $\sim_B(X)$ is:

$$\sim_B(X) = (\sim_B(X), \sim_B(X)) = \{\text{university2}\}, \{\text{university1, university2, university3}\}$$

and the quality of approximation of $X = \{\text{university2, university3}\}$ by the roughest $\sim_B(X)$ is $\frac{1}{3}$.

**5. Object oriented reducts**

We provide definition of reducts in the object-oriented rough set models and propose algorithms for calculating reducts.
Consider the attributes that preserve the indiscernibility relation and, consequently, set approximation. There are usually several subsets of attributes and those which are minimal are called reducts.

**Positive, Negative and Boundary Regions:**

Let \( P \) and \( Q \) be equivalence relations over \( O \), then the positive, negative and boundary regions are defined as:

\[
\text{POS}_P(Q) = \bigcup_{x \in O/Q} \sim_p(X)
\]

... \( (27) \)

\[
\text{NEG}_P(Q) = O - \bigcup_{x \in O/Q} \sim_p(X)
\]

... \( (28) \)

\[
\text{BND}_P(Q) = \bigcup_{x \in O/Q} \sim_p(X) - \bigcup_{x \in O/Q} \sim_p(X)
\]

... \( (29) \)

The positive region comprises all objects of \( O \) that can be classified to classes of \( O/Q \) using the information contained within attributes \( P \). The boundary region \( \text{BND}_P(Q) \), is the set of objects that can possibly, but not certainly, be classified in this way. The negative region, \( \text{NEG}_P(Q) \), is the set of objects that can not be classified to classes of \( O/Q \).

**Dispensable & Indispensable attributes**

Let \( c \in AT \). Attribute \( c \) is dispensable in OOIS if \( \text{POS}_{AT} (D) = \text{POS}_{AT\setminus c} (D) \), otherwise attribute \( c \) is indispensable in OOIS.

The AT-positive region of \( D \):

\[
\text{POS}_{AT} (D) = \bigcup_{x \in O/D} \sim_{AT}(X)
\]

... \( (30) \)

**Independent Object oriented Information system:**

OOIS \( (O,C,N) \) is independent if all \( c \in AT \) are indispensable in OOIS.

**Reduct and Core:**

- The set of attributes \( R \subseteq AT \) is called a reduct of \( AT \), if \( \text{OOIS}'(O,R,N) \) is independent and \( \text{POS}_R (D) = \text{POS}_{AT} (D) \).
- The set of all the condition attributes indispensable in \( T \) is denoted by \( \text{CORE } (AT) = \bigcap \text{RED } (AT) \) where \( \text{RED } (AT) \) is the set of all reducts of \( AT \).

**Feature Dependency and significance:**

An important issue in data analysis is discovering dependencies between attributes. Intuitively, a set of attributes \( Q \) depends totally on a set of attributes \( P \), denoted \( P \Rightarrow Q \), if all attribute values from \( Q \) are uniquely determined by values of attributes from \( P \). If there exists a functional dependency between values of \( Q \) and \( P \), then \( Q \) depends totally on \( P \). In object oriented rough set theory, dependency is defined in the following way:

For \( P,Q \subset AT \), it is said that \( Q \) depends on \( P \) in a degree \( k \) (0 ≤ \( k \) ≤1), denoted \( P \Rightarrow_k Q \), if

\[
k = \gamma_p(Q) = \frac{|\text{POS}_P(Q)|}{|O|}.
\]

... \( (31) \)

By calculating the change in dependency when a feature is removed form the set of considered possible features, an estimate of the significance of that feature can be obtained. The higher the change in dependency, the more significant the
feature is. If the significance is 0, then the feature is dispensible. More formally, given \( P, Q \) and a feature \( x \in P \), the significance of feature \( x \) upon \( Q \) is defined by
\[
\sigma_p(Q, a) = \gamma_p(Q) - \gamma_{p-\{a\}}(Q). \tag{32}
\]

**Reducts**

For many application problems, it is necessary to maintain a concise form of the object oriented information system. One way to complement this is to search for a minimal representation of the original data set. For this, the concept of a reduct is introduced and defined as a minimal subset \( R \) of the initial attribute set \( A_T \) such that for a given set of attributes \( D \),
\[
\gamma_R(D) = \gamma_{A_T}(D). \tag{33}
\]

From the literature, \( R \) is a minimal subset if \( \gamma_{R-\{a\}}(D) \neq \gamma_R(D) \) for all \( a \in R \). This means that no attributes can be removed from the subset affecting the dependency degree. Hence, a minimal subset by this definition may not be the global minimum(a reduct of smallest cardinality). A given dataset may have many reduct sets, and the collection of all reducts is denoted by
\[
R_{all} = \{X / X \subseteq A_T, \gamma_X(D) = \gamma_{A_T}(D); \gamma_{X-\{a\}}(D) \neq \gamma_X(D), \forall a \in X\}. \tag{33}
\]

The intersection of all the sets in \( R_{all} \) is called the core, the elements of which are those attributes that cannot be eliminated without introducing more contradictions to the representation of the dataset.

For many tasks, a reduct of minimal cardinality is ideally searched for. That is, an attempt is to be made to locate a single element of the reduct set \( R_{min} \subseteq R_{all} \):
\[
R_{min} = \{X / X \in R_{all}, \forall Y \in R_{all}, |X| \leq |Y|\} \tag{34}
\]

**Quick Reduct** \((A_T, D)\) :

\( A_T \), the set of all conditional attributes; \( D \), the set of decision attributes

(1) \( R \leftarrow \{\} \)
(2) do
(3) \( T \leftarrow R \)
(4) \( \forall x \in (A_T - R) \)
(5) if \( \gamma_{R-\{x\}}(D) > \gamma_T(D) \)
(6) \( T \leftarrow R \cup \{x\} \)
(7) \( R \leftarrow T \)
(8) until \( \gamma_R(D) = \gamma_{C}(D) \)
(9) return \( R \)

**Discernibility Matrix** :

Many applications of rough sets make use of discernibility matrix for finding rules or reducts. A discernibility matrix of a decision table \((O, A_T \cup D)\) is a symmetric \(|O| \times |O|\) matrix with entries defined by:
\[
c_{ij} = \{a \in A_T / a(x_i) \neq a(x_j)\} i, j = 1, ..., |O| \tag{35}
\]

Each \( c_{ij} \) contains those attributes that differ objects \( i \) and \( j \).
A discernibility function $f_{OOIS}$ for an object oriented information system is a boolean function of $m$ Boolean variables $a_1^*,...,a_m^*$ corresponding to the membership of attributes $a_1,...,a_m$ to a given entry of the discernibility matrix defined as below:

$$f_{OOIS}(a_1^*,...,a_m^*) = \bigwedge \{ \forall c_{ij}^* / 1 \leq j \leq l \ O \ | c_{ij}^* \neq \phi \} \quad \text{...} \quad (36)$$

where $c_{ij}^* = \{a^* / a \in c_{ij}^* \}$. By finding the set of all prime implicants of the discernibility function, all the minimal reducts of a system may be determined.

**Example 6:** Consider the following table which shows an example of object oriented data set.

<table>
<thead>
<tr>
<th>$x \in O$</th>
<th>Ncollege</th>
<th>Ndepartment</th>
<th>Nstudent</th>
<th>$\Rightarrow$</th>
<th>Granting Research Fund(GRF)</th>
</tr>
</thead>
<tbody>
<tr>
<td>university1</td>
<td>5</td>
<td>10</td>
<td>100</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>university2</td>
<td>6</td>
<td>12</td>
<td>110</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>university3</td>
<td>8</td>
<td>14</td>
<td>100</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>university4</td>
<td>5</td>
<td>10</td>
<td>100</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>university5</td>
<td>6</td>
<td>12</td>
<td>110</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>university6</td>
<td>7</td>
<td>15</td>
<td>100</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>university7</td>
<td>8</td>
<td>14</td>
<td>100</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>university8</td>
<td>9</td>
<td>15</td>
<td>140</td>
<td>No</td>
<td></td>
</tr>
</tbody>
</table>

The indiscernibility classes defined by $R= \{\text{Ncollege, Ndepartment, Nstudent}\}$ are $\{\text{university1, university4}\}, \{\text{university2, university5}\}, \{\text{university3, university7}\}, \{\text{university6}\}, \{\text{university8}\}$.

Let $G = \{x / \text{GRF}(x) = \text{Yes}\}$.

$G1 = \{o / G(o) = \text{Yes} \} = \{ \text{university1, university3, university5, university6} \}$

$\overline{RG1} = \{ \text{university6} \}$

$\overline{RG1} = \{\text{university1, university4, university2, university5, university3, university7, university6}\}$

$G2 = \{o / G(o) = \text{Yes} \} = \{ \text{university2, university4, university7, university8} \}$
\( RG2 = \{\text{university}8\} \)

\( \overline{RG2} = \{\text{university}1, \text{university}4, \text{university}2, \text{university}5, \text{university}3, \text{university}7, \text{university}8\} \)

Let \( P = \{\text{Ncollege}, \text{Ndepartment}, \text{Nstudent}\}, Q = \{\text{GRF}\}. \)

\[
P^*_P(Q) = \bigcup_{x \in U} P(x)
\]

\( = RG1 \cup RG2 \)

\( = \{\text{university}6, \text{university}8\} \)

\[
NEG_P(Q) = O - \bigcup_{x \in U} \overline{P}(x)
\]

\( = O - \{RG1 \cup \overline{RG2}\} \)

\( = O - O = \emptyset \)

\[
BND_P(Q) = \bigcup_{x \in U} \overline{P}(x) - \bigcup_{x \in U} P(x)
\]

\( = \{\text{university}1, \text{university}3, \text{university}4, \text{university}5, \text{university}7, \text{university}8\} \)

\( \text{REDUCT}1 = \{\text{Ndepartment}, \text{Ncollege}\} \)

\( \text{REDUCT}2 = \{\text{Ncollege}, \text{Nstudent}\} \)

\( \text{CORE} = \text{REDUCT}1 \cap \text{REDUCT}2 \)

\( = \{\text{Ncollege}\} \)

Let \( C = \{\text{Ncollege, NDepartment, Nstudent}\} \) and \( D = \{\text{GRF}\} \) be set of condition and decision attributes respectively. Discernibility Matrix is

### Table 2 – The Decision –Relative Discernibility Matrix

<table>
<thead>
<tr>
<th>x ∈ O</th>
<th>U1</th>
<th>U2</th>
<th>U3</th>
<th>U4</th>
<th>U5</th>
<th>U6</th>
<th>U7</th>
<th>U8</th>
</tr>
</thead>
<tbody>
<tr>
<td>U1</td>
<td></td>
<td>[NC, ND, NS]</td>
<td>---</td>
<td></td>
<td></td>
<td></td>
<td>[NC, ND]</td>
<td>[NC, ND, NS]</td>
</tr>
<tr>
<td>U2</td>
<td>[NC, ND, NS]</td>
<td>---</td>
<td>[NC, ND]</td>
<td>---</td>
<td></td>
<td></td>
<td>[NC, ND, NS]</td>
<td>---</td>
</tr>
<tr>
<td>U3</td>
<td>---</td>
<td>[NC, ND, NS]</td>
<td>---</td>
<td>[NC, ND]</td>
<td>---</td>
<td></td>
<td>[NC, ND, NS]</td>
<td>---</td>
</tr>
<tr>
<td>U4</td>
<td></td>
<td></td>
<td>[NC, ND]</td>
<td></td>
<td>[NC, ND, NS]</td>
<td>---</td>
<td>[NC, ND]</td>
<td>---</td>
</tr>
<tr>
<td>U5</td>
<td></td>
<td></td>
<td></td>
<td>[NC, ND, NS]</td>
<td>---</td>
<td>[NC, ND]</td>
<td></td>
<td>[NC, ND, NS]</td>
</tr>
<tr>
<td>U6</td>
<td></td>
<td></td>
<td>[NC, ND]</td>
<td></td>
<td></td>
<td>[NC, ND]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>U7</td>
<td>[NC, ND]</td>
<td>---</td>
<td></td>
<td>[NC, ND, NS]</td>
<td>---</td>
<td>[NC, ND]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>U8</td>
<td>[NC, ND, NS]</td>
<td>---</td>
<td>[NC, ND]</td>
<td></td>
<td>[NC, ND, NS]</td>
<td>---</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

where NC, ND, NS represents Ncollege, NDepartment, Nstudent respectively.

From Table 2, the decision relative discernibility function is (with duplicates removed):

\[
f_D(\text{Ncollege, Ndepartment, Nstudent}) = (\text{Ncollege} \lor \text{Ndepartment} \lor \text{Nstudent}) \land
\]

\[
(\text{Ncollege} \lor \text{Ndepartment}) \land (\text{Ncollege} \lor \text{Nstudent})
\]

Therefore, the minimal reducts are \{Ncollege, Ndepartment\} and \{Ncollege, Nstudent\}.

### Conclusion

We have introduced object-oriented paradigm into the frame work of “traditional” rough set theory. First we have constructed the class, object, and
name structures using rough set theory. Next, combining class, name and object structures, we have proposed object-oriented information systems. Moreover, we have introduced indiscernibility relations on the set of objects, lower and upper approximations, and object-oriented rough sets in the object-oriented information systems and we have introduced the concept of object oriented reducts, object oriented reducts using indiscernibility matrix for object oriented reducts. We can develop object oriented dynamic reducts using rough set theory and genetic algorithm for object oriented reducts using rough set theory.

References


Received: March, 2010