Dynamic Reduct and Its Properties In the Object-Oriented Rough Set Models

M.Srivenkatesh
Dept.Of.Computer Science
GITAM University
Visakapatnam, India

P.V.G.D.Prasadreddy
Dept.CSSE
Andhra University
Visakapatnam, India

Y.Srinivas
Dept of IT
Vignan Engg.College
Visakapatnam, India

***Abstract*—This Paper deals with a new type of Reduct in the object-oriented rough set model which is called dynamic reduct. In the object–oriented rough set models, objects are treated as instances of classes, and illustrate structural hierarchies among objects based on is-a relationship and has-a relationship[6]. In this paper, we propose dynamic reduct and the notation of core according to the dynamic reduct in the object-oriented rough set models. It describes various formal definitions of core and discusses some properties about dynamic core in the object-oriented rough set models.**

Keywords — Rough Set, Dynamic Reduct, Feature Core, Indiscernibility Relations, Discernibility Matrices.

I. INTRODUCTION

Rough set theory [1, 2] deals with approximation and reasoning about data. In the aspect of approximation, the basic concepts are lower approximation, upper approximation by indiscernibilty relations which illustrate set-theoretic approximations of any given subset of data. we can find reducts and decision rules in traditional rough set theory. The object-oriented rough set model is an extension of the "traditional rough set theory" by introducing object-oriented paradigm[4] used in computer science and the Object-oriented rough set models[6] illustrates hierarchical structures between classes, names and objects based on is-a and has-a relationships. Kudo and Murai have extended the object–oriented rough set models to treat incomplete information[8].In the papers [6][8], formulation of the object–oriented rough set model was concentrated to the aspect of approximation, and in the paper[7], reasoning about objects in the object–oriented rough set model by introducing decision rules in the object–oriented rough set model, and revised discernibility matrix for finding reducts in the object-oriented rough set model was discussed. However dynamic reducts and properties in the object-oriented rough set model have not been discussed. In paper [5], deals with a comparison of dynamic and non-dynamic rough set methods for extracting laws from decision tables.

In this paper, we propose dynamic reduct and its properties in the object–oriented rough set models. The Reducts are stable in decision rules are called dynamic reducts in the object-oriented rough set models. Dynamic Reducts define set of classes or set of classes with names are called dynamic core in the object-oriented rough set models. This is the classes or classes with names included in all dynamic reducts in the object-oriented rough set models.

The rest of the paper is organized as follows: In section II, we review the object-oriented rough set model. In section III, we introduce dynamic reduct in the object-oriented rough set models. In section IV, we introduce dynamic core in the object-oriented rough set models. In section V, we introduce (F-λ)–dynamic core in the object-oriented rough set model. In section VI, we introduce generalized dynamic core in the object-oriented rough set model. In section VII, we draw conclusion.

II. THE OBJECT–ORIENTED ROUGH SET MODEL

We briefly review the object–oriented rough set model. First, we describe the concept of class, name and object. Next, we illustrate well-defined structures as a basic framework of the object–oriented rough set model. Moreover, we introduce equivalence relations based on "equivalence as instances". Note that the contents of this section are entirely based on the papers [6][8].

A. Class, Name, Object

Formally, a class structure $\mathbf{C}$, a name structure $\mathbf{N}$ and a object structure $\mathbf{O}$ are defined by the following triples, respectively:

$$\mathbf{C} = (C, \ni_C, \supseteq_C),\ \mathbf{N} = (N, \ni_N, \supseteq_N),\ \mathbf{O} = (O, \ni_O, \supseteq_O),$$

where C, N and O are finite and disjoint non-empty sets such that $|C| \leq N|$ ($|X|$ is the cardinality of X). Each element $c \in C$ is called a class. Similarly, each $n \in N$ is called a name, and each $o \in O$ is called an object. The relation $\ni_X \ (X \in \{C, N, O\})$ is an acyclic binary relation on X, and the relation $\supseteq_X$ is a reflexive, transitive, and asymmetric binary relation on X. Moreover, $\ni_X$ and $\supseteq_X$ satisfy the following property:

$$\forall\ x_i, x_j, x_k \in X,, x_i \supseteq_X x_j, x_j \ni_X x_k \Rightarrow$$

$$x_i \ni_X x_k \quad (1)$$

The class name and object structures have the following characteristics, respectively:

– The class structure illustrates abstract data forms and those hierarchical structures based on part / whole relationship (has-a relation) and specialized/ generalized relationship (is-a relation).

– The name structure introduces numerical constraint of objects and that identification, which provide concrete design of objects.

– The object structure illustrates actual combination of objects.

Two relations $\ni_X$ and $\supseteq_X$ on $X \in \{C, N, O\}$ illustrate hierarchical structures among elements in *X*. The relation $\ni_X$ is called a has-a relation, which illustrates part / whole relationship. $x_i \ni_x x_j$ means "x_i has-a x_j", or "x_j is a part of x_i". For example, $c_i \ni_C c_j$ means that "the class c_j has a class c_j", or "c_j is a part of c_j". On the other hand, the relation $\supseteq_X$ is called an is-a a relation, which illustrates specialized / generalized relationship. $x_i \supseteq_X x_j$ means that "x_i is-a x_j". For example, $\supseteq_C$ illustrates relationship between super classes and subclasses, and $c_i \supseteq_C c_j$ means that "c_i is a super class of c_j", or "c_j is a subclass of c_i".

B. Well–Defined Structures

Each object $o \in O$ is defined as an instance of some class $c \in C$, and the class of *o* is identified by the class identifier function. The class identifier id_C is a p-morphism between O and C [3], that is, the function $id_C : O \to C$ satisfies the following conditions:

1. $\forall o_i, o_j \in O,\ o_i \ni_O o_j \Rightarrow id_C(o_i) \ni_C id_C(o_j)$.
2. $\forall o_i \in O, \forall c_j \in C, id_C(o_i) \ni_C c_j \Rightarrow$
$$\exists\ o_j \in O\ s.t. o_i \ni_c o_j\ and\ id_C(o_j) = c_j, (2)$$

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

The object structure **O** and the class structure **C** are also connected through the name structure **N** by the naming function $nf : N \to C$ and the name assignment $na : O \to N$. The naming function provides names to each class, which enable us to use plural instances of the same class simultaneously. On the other hand, the name assignment provides names to every object, which enable us to identify objects by names.

Formally, the naming function $nf : N \to C$ is a surjective p-morphism between **N** and **C**, and satisfies the following name preservation constraint:

-For any $n_i, n_j \in N$, if $nf(n_i) = nf(n_j)$, then $H_N(c/n_i) = H_N(c/n_i) = H_N(c/n_j)$ is satisfied for all $c \in C$, where $H_N(c|n) = \{nj \in N \mid n \ni_N nj, f(nj) = c\}$ is the set of names of *c* that *n* has. The requirement that *nf* is a surjective p-morphism means that there is at least one name for each class, and structures between names reflect all structural characteristics between classes. The name preservation constraint requires that, for any class $c_i, c_j \in C$ such that $c_i \ni_C c_j$, and any name $n \in N$ with $nf(n) = c_i$, all names of the parts of *c* are uniquely determined. Thus, the number of names of c_j is fixed as $m = |H_N(c_j/n)|$, and we can simply say that "the class c_i has *m* objects of the class c_j".

On the other hand, the name assignment $na : O \to N$ is a p-morphism between **O** and **N**, and satisfies the following uniqueness condition:

-For any $x \in O$, if $H_O(x) \neq \phi$, the restriction of *na* into $H_O(x) : na / H_O(x) : H_O(x) \to N$ is injective, where $H_O(x) = \{y \in O / x \ni_O y\}$ is the set of objects that *x* has. $na(x) = n$ means that the name of the object *x* is *n*. The uniqueness condition requires that all distinct parts $y \in H_O(x)$ have different names.

We say that **C**, **N** *and* **O** *are* well-defined if and only if there exist a naming function $nf : N \to C$ and a name assignment $na : O \to N$ such that

$$id_C = nf \circ na, \quad (3)$$

that is, $id_C(x) = nf(na(x))$ for all $x \in O$.

We concentrate well-defined class, name and object structures. In well-defined structures, if a class c_i has *m* objects of a class c_j, then any instance o_i of the class c_i has exactly *m* instances $o_{j1}, \ldots, o_{jm}$ of the class c_j [2].This good property enables us the following description for clear

representation of objects. Suppose we have $o_1, o_2 \in O$, $n_1, n_2 \in N$, and $c_1, c2 \in C$, such that $o_i \ni_o o_2$, and $na(o_i) = n_i, nf(n_i) = c_i$ for $i \in \{1,2\}$. We denote $o_1.n_2$ instead of o_2 by means of "the instance of c_2 named n_2 as a part of o_1".

C. Indiscernibility Relations in the Object – Oriented Rough Set Model

All equivalence relations in object–oriented rough set models are based on the concept of equivalence as instances. In [6], to evaluate equivalence of instances, an equivalence relation ~ on *O* are recursively defined as follows:

$x \sim y \Leftrightarrow$ *x* and *y* satisfy the following two conditions:

1. $id_C(x) = id_C(y)$, and,

2. $\begin{cases} x.n \sim y.n, \forall n \in H_N\ (na(x)) & \text{if } H_N(na(x)) \neq \phi \\ Val(x) = Val(y) & \text{otherwise} \end{cases}$ (4)

Where $H_N(na(x))$ is the set of names that $na(x)$ has. *Val*(*x*) is the "value" of the "value object" *x*. Because *C* is a finite non-empty set and $\ni_C$ is acyclic, there is at least one class *c* such that *c* has no other class c', that is, $c \not\ni_C c'$ for any $c' \in C$. We call such class *c* an attribute, and denote the set of attributes by *AT*. 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* as an instance of the attribute *a* represents a "value" of the attribute.

D. Object-Oriented Rough Sets

$x \sim y$ means that the object *x* is equivalent to the object *y* as an instance of the class $id_C(x)$. Using the equivalence relation ~, an equivalence relation $\sim_B$ with respect to a given subset $B \subseteq N$ of names is defined as follows:

$x \sim_B y \Leftrightarrow$

x and *y* satisfy the following two conditions:

1. $B \cap H_N(na(x)) = B \cap H_N(na(y))$, and,

2. $\forall n[n \in B \cap H_N(na(x)) \Rightarrow x.n \sim y.n]$

(5)

$x \sim_B y$ means that *x* and *y* are equivalent as instances of the class $id_C(x)$ in the sense that, for all $n \in B \cap H_N(na(x))$, *x* and *y* have equivalent instances of the class $id_C(x.n)$. Equivalence classes $[x]_{\sim_B}$ by $\sim_B$ are usually defined. Note that, in the "traditional" rough set theory, all equivalence classes concern the same attributes. On the other hand, each equivalence class of the object–oriented rough set model may concern different classes. In particular, if $B \cap H_N(na(x)) = \phi$, the equivalence class $[x]_{\sim_B}$ is the set of objects that are not concerned any class $nf(n), n \in B$, at all.

Suppose *OORS* is Object-Oriented Rough Set Model, N is non empty subset of names , and $\sim_B$ be the equivalence relation defined by eq(5) .For any subset X⊆O of objects

$$\underline{\sim_B}(X) = \{x \in O \mid [x]_B \subseteq X\}$$

$$\overline{\sim_B}(X) = \{x \in O \mid [x]_B \cap X = \phi\}$$

(6)

are called *B*-lower approximation and *B*-upper approximation respectively. The *B*-lower approximation is also called the position region denoted by $POS_B(X)$.

Note that the contents of this section are entirely based on the paper [7].

III. Decision Rules and Discernibility Matrices in the Object–Oriented Rough Set Model

A. Decision Rule

Let OORS(**C**, **N**, **O**) be the object-oriented rough set model where $\mathbf{C} = (C, \ni_C, \supseteq_C)$, $\mathbf{N} = (N, \ni_N, \supseteq_N)$, $\mathbf{O} = (O, \ni_O, \supseteq_O)$ be the well defined class, name, object structures, respectively. Similar to the decision table in rough set theory, we divide the set of names *N* into the following two parts: the set of names that may appear in antecedents of decision rules (called condition names) N_{CON}, and the set of names that may appear in conclusions of decision rules (called decision names) N_{DEC}. Note that $N = N_{CON} \cup N_{DEC}$ and $N_{CON} \cap N_{DEC} = \phi$. The decision names provide decision classes as equivalence classes $[x]_{\sim N_{DEC}}$ based on the equivalence relation N_{DEC} by (5). Decision rules in the object–oriented rough set model are defined as follows.

Definition 1: A decision rule in the object-oriented rough set model has the following form:

$$c \wedge c.n1 \sim o.n1 \wedge ..... \wedge c.n_i \sim o.n_i \Rightarrow c.m1 \sim o.m1 \wedge ..... \wedge c.m_j \sim o.m_j \quad (7)$$

Where $c \in C, o \in O$ such that $id_C(o) = c, n_1 ..., n_i \in N_{CON} \cap H_N(na(o))(i \geq 0)$ and $m_1, ..., m_j \in N_{DEC} \cap H_N(na(o))(j \geq 1)$. We call this

rule a decision rule of the class c by the object o, and denote DR(C;O) .The decision rule *DR*(*c*; *o*) means that, for any object $o' \in O$, if o' is an instance of *c* and each part $o'.n_k$ is equivalent to $o.n_k$ ($k \le i$), then all parts $o'.m_l$ are also equivalent to $o.m_l$ ($l \le j$), respectively. Thus, *DR*(*c*; *o*) describes a certain property about combination of objects as an instance of the class *c*.

As a special case, we allow rules that have no condition names, that is, the case of *i* = 0 in (7) as follows:

$$c \Rightarrow c.m_1 \sim o.m_1 \wedge \cdots \wedge c.m_j \sim o.m_j .$$

This rule illustrates that all instances o' of the class *c* have some parts $o'.m_k (1 \le k \le j)$ that are equivalent to $o.m_k$, respectively. On the other hand, we require that there is at least one name $m \in N_{DEC}$ such that $m \in H_N(na(o))$ .This means that any object that has no decision name are not the target of decision rule generation.

B. Discernibility Matrix

Definition 2: A discernibility matrix of the object-oriented rough set model is a k×k matrix whose elements δ_{ij} at the i-the row and j-the column is defined as follows.

$$\delta_{ij} = \begin{cases} \{id_C(o_i)\} & \text{if } id_C(o_i) \neq id_C(o_j) \text{ and } \exists\, m \in N_{DEC} \cap H_N(na(o_i)) \text{ s.t } o_i.m \not\sim o_j.m \\ \left\{ id_C(o_i).n \middle| \begin{array}{l} n \in H_N(na(o_i)), \\ n \in N_{CON}, \\ o_i.n \not\sim o_j.n \end{array} \right\} & \text{if } id_C(o_i) = id_C(o_j) \text{ and } \exists\, m \in N_{DEC} \cap H_N(na(o_i)) \text{ s.t } o_i.m \not\sim o_j.m \\ \phi & \text{otherwise} \end{cases} \quad (8)$$

Where k is the number of objects, that is, $|o| = k.o_i \not\sim o_j.n$ means that $o_i.n$ is not equivalent to $o_j.n$

The element δ_{ij} is the set of classes that we should check to distinguish the object o_i and the object o_j . Thus, when we need to distinguish o_i and o_j , we check the class $id_C(o_i)$ and $id_C(o_j)$ firstly, and if these classes are not equal, we can distinguish these objects. Otherwise, we need to compare parts $o_i.n$ and $o_j.n$ such that *n* ∈ *NCON* $\cap H_N(na(o_i))$. Note that, different from the "traditional" discernibility matrix, we have generally $\delta_{ij} \neq \delta_{ji}$ in the revised discernibility matrix. Similar to the case of calculating reducts by the "traditional" discernibility matrix, we construct reducts of the object–oriented rough set model. First, for each element δ_{ij} in the revised discernibility matrix, we construct the following formula L(δ_{ij}):

$$L(\delta_{ij}) = \begin{cases} c & \text{if } \delta_{ij} = \{c\}, \\ c.n_1 \vee \ldots\ldots\ldots \vee c.n_l & \text{if } \delta_{ij} = \{c.n_1 \ldots\ldots\ldots, c.n_l\}, \\ \tau & \text{if } \delta_{ij} = \phi . \end{cases} \quad (9)$$

The intention of L(δ_{ij}) is that, for example the case of $L(\delta_{ij}) \equiv c.n_1 \vee .. \vee c.n_l$, we can distinguish o_i and o_j by checking at least one of $c.n_s$ ($1 \le s \le l$).

Next, connecting all formulas $L(\delta_{ij})$ by the logical product, we get a formula $\wedge_{i=1}^{k} \wedge_{j=1}^{k} L(\delta_{ij})$. This formula is the conjunctive normal form. Thus, finally, we transform this formula to the disjunctive normal form that is logically equivalent to $\wedge_{i=1}^{k} \wedge_{j=1}^{k} L(\delta_{ij})$ with no redundant expression as follows:

$$\bigwedge_{i=1}^{k} \bigwedge_{j=1}^{k} L(\delta_{ij}) = \bigvee_{s=1}^{m} \bigwedge_{t=1}^{st} c_{st} \quad , \quad (10)$$

where each conjunction $\vee_{t=1}^{st} C_{lt}$ describes a reduct of the object–oriented rough set model $R = \{c_{l1}, \cdots, c_{lst}\}$. This is because, for each element δ_{ij} of the revised discernibility matrix, *R* contains at least one expression *c* or *c.n* such that $c \in \delta_{ij}$ or $c.n \in \delta_{ij}$.

C. A Method of Generating Decision Rules

Let *R* be a reduct of the object–oriented rough set model. We consider decision rules from the reduct and each object in decision classes. However, for each object *o* in any decision class $[x]_{\sim NDEC}$, not all classes $c \in R$ and $c.n \in R$ are concerned with *o*. Thus, for each object *o* $\in [x]_{\sim NDEC}$ such that $id_C(o) = c$, we construct a decision rule *DR*(*c*; *o*) in the object–oriented rough set model as follows:

1. Select the class *c* such that $id_C(o)$ = *c* and all classes $c.n_s$ from the reduct *R*.

2. Construct an expression $c.n_s \sim o.n_s$ for each selected $c.n_s$ and the object *o*, and connect the class *c* and these expressions by $\wedge$ follows:

(Antecedents) $c \wedge c.n_1 \sim o.n_1 ..... \wedge c.n_l \sim o.n_l$

3. Construct an expression $c.m_t \sim o.m_t$ for each $c.m_t$ such that $m_t \in N_{DEC} \cap H_N(na(o))$, and connect the class *c* and these expressions by $\wedge$ as follows:

(Conclusions) $c \wedge c.m_1 \sim o.m_1 \wedge ... \wedge c.m_u \sim o.m_u$

4. Construct the decision rule *DR*(*c* : *o*) by connecting antecedents and conclusions by $\Rightarrow$ as follows:

$$c \wedge c.n_1 \sim o.n_1 \wedge ... \wedge c.n_l \sim o.n_l \Rightarrow c \wedge c.m_1 \sim o.m_1 \wedge ... \wedge c.m_u \sim o.m_u \quad (11)$$

IV. Dynamic Reduct in the Object-Oriented Rough Set Models.

Reducts generated from object-oriented rough set models are sensitive to changes in the models. This can be seen by removing a randomly chosen set of objects from the original object set. Those reducts frequently occurring in random sub-object-oriented models can be considered to be stable; it is this object-oriented reducts that are encompassed by dynamic reducts in the object-oriented rough set models.

The reducts stable in decision rules are called object-oriented dynamic reducts. Dynamic reducts define set of classes or set of classes with names are called dynamic core in the object-oriented rough set models. These are the classes or classes with names included in all object-oriented dynamic reducts.

The rules calculated by means of dynamic reducts are better pre-disposed to classify unseen cases, because these reducts in the object-oriented rough set models are in some sense the most stable reducts, and they are the most frequently appearing reducts in sub-object-oriented rough set models created by random samples of a given object-oriented rough set models.

Algorithm 1: Dynamic Reduct in the Object-Oriented Rough Set Model.

DynamicRedOORS(OORS, ε, nts)

OORS, the original object-oriented rough set model;

ε, the dynamic reduct threshold;

nts, the number of iterations.

$R \leftarrow \{\ \}$

$T \leftarrow$ calculateAllReducts(OORS)

for j=1..nts

$OORS' \leftarrow$ deleteRandomRows$(OORS)$

$R \leftarrow R \cup$ calculateAllReducts$(OORS')$

$\forall P \in T$

If $s_F(P,R) \geq \varepsilon$

Output P.

Firstly, all reducts are calculated for the given object-oriented rough set model, OORS. Then, new sub-object-oriented rough set model $OORS'$ by randomly deleting one or more rows form OORS. All reducts found for each sub-object-oriented rough set model, and the dynamic reducts are computed using $s_F(P,R)$ which denotes the significance of reduct P with all reducts found, R.

Definition 3: Let $OORS(\mathbf{C}, \mathbf{N}, \mathbf{O})$ be the object-oriented rough set model where $\mathbf{C} = (C, \ni_C, \supseteq_C)$, $\mathbf{N} = (N, \ni_N, \supseteq_N)$, $\mathbf{O} = (O, \ni_O, \supseteq_O)$ be the well defined class, name, object structures, respectively. Let $OORS'(\mathbf{C}', \mathbf{N}', \mathbf{O}')$ be the sub-object-oriented rough set model, where $\mathbf{C}' = (C', \ni_C, \supseteq_C)$, $\mathbf{N}' = (N', \ni_N, \supseteq_N)$, $\mathbf{O}' = (O', \ni_O, \supseteq_O)$ be the well defined class, name, object structures respectively. Let $\mathbf{C}' \subseteq \mathbf{C}$, $\mathbf{N}' \subseteq \mathbf{N}$, $\mathbf{O}' \subseteq \mathbf{O}$, $OORS' \subseteq OORS$, $OORS'$ is called sub-object-oriented rough set model of $OORS$. Let $\rho(OORS)$ be the set of all sub-object-oriented rough set model of $OORS$.

Definition 4: Let $OORS(\mathbf{C}, \mathbf{N}, \mathbf{O})$ be the object-oriented rough set model where $\mathbf{C} = (C, \ni_C, \supseteq_C)$, $\mathbf{N} = (N, \ni_N, \supseteq_N)$, $\mathbf{O} = (O, \ni_O, \supseteq_O)$ be the well defined class, name, object structures, respectively. Let $RED(OORS)$ denotes the set which contains all reducts of $OORS$ and $RED(OORS')$ denotes the set which includes all Reducts of $OORS'$. A object-oriented rough set model at least contains one reduct, which is just itself, so the set of reduct in the object-oriented rough set model is not empty.

In many cases, a given object–oriented rough model may exist several reducts. Each reduct can produce a rule set, and it is difficult to justify which the best rule is set. Therefore it

is a important to search the most stable reduct in the object-oriented rough set model, and hence reduct in the object-oriented rough set model is proposed in this case.

Definition 5: Let $OORS(\mathbf{C}, \mathbf{N}, \mathbf{O})$ be the object-oriented rough set model where $\mathbf{C} = (C, \ni_C, \supseteq_C)$, $\mathbf{N} = (N, \ni_N, \supseteq_N)$, $\mathbf{O} = (O, \ni_O, \supseteq_O)$ be the well defined class, name, object structures, respectively. Let $\rho(OORS)$ be the set of all sub-object-oriented rough set model of OORS, $F \subseteq \rho(OORS)$ is called a family of object-oriented rough set model OORS. Let $F \subseteq P(OORS)$ be the dynamic reduct of object-oriented rough set model OORS, denoted by $DR\ (OORS, F)$[5], and

$$DR\ (OORS, F) = RED(OORS) \cap \bigcap_{OORS' \in F} RED(OORS') \quad (12)$$

Any element of $DR(OORS, F)$ is called an F-dynamic reduct, which describes the most stable reducts in object-oriented rough set models. From the definition of dynamic reduct in the object-oriented rough set model, it follows that, it is also reduct of all sub-object-oriented rough set models from a given family F by random sampling.

The concept of dynamic core in the object-oriented rough set model is introduced here.

Dynamic Core in the Object-Oriented Rough Set Model.

Reduct finding is the basic problem in object-oriented rough set models, and the computation of feature core in the object-oriented rough set model is especially important for resolving this problem. All classes with names or classes in the feature core will be presence in any reduct, otherwise revised discernible relation in object-oriented rough set models can not be ensured. According to the feature core one can construct object-oriented reduct heuristically, and the efficiency of reduct can be improved greatly. For dynamic reduct in the object-oriented rough set model, the need for feature core is to be probed.

Definition 4: Let $OORS(\mathbf{C}, \mathbf{N}, \mathbf{O})$ be the object-oriented rough set model where $\mathbf{C} = (C, \ni_C, \supseteq_C)$, $\mathbf{N} = (N, \ni_N, \supseteq_N)$, $\mathbf{O} = (O, \ni_O, \supseteq_O)$ be the well defined class, name, object structures, respectively. Let $OORS'(\mathbf{C}', \mathbf{N}', \mathbf{O}')$ be the sub-object-oriented rough set model, where $\mathbf{C}' = (C', \ni_C, \supseteq_C)$, $\mathbf{N}' = (N', \ni_N, \supseteq_N)$, $\mathbf{O}' = (O', \ni_O, \supseteq_O)$ be the well defined class, name, object structures respectively. Let $\mathbf{C}' \subseteq \mathbf{C}$, $\mathbf{N}' \subseteq \mathbf{N}$, $\mathbf{O}' \subseteq \mathbf{O}$, $OORS' \subseteq OORS$, $OORS'$ is called sub-object-oriented rough set model of OORS. The feature core of object-oriented rough set models in static reduct is

$$CORE(OORS) = \bigcap_{i=1}^{p} C_i \quad \text{or} \quad \bigcap_{i=1}^{p} \bigcap_{j=1}^{q} C_i \,.\, n_j = \bigcap_{R \in RED(OORS)} R \quad (13)$$

Where $C_i\ (1 \le i \le p), C_i.n_j (1 \le i \le p), (1 \le j \le q)$ represents classes and classes with names belongs to reducts in the object-oriented rough set model.

Algorithm 2: Core of the object-oriented rough set model.

Input: object-oriented rough set model OORS.

Output: the core of the object-oriented rough set model.

T ← calculateAllReducts (OORS).

Core ← finding intersection of elements of T.

Output Core.

Definition 5: Let $OORS(\mathbf{C}, \mathbf{N}, \mathbf{O})$ be the object-oriented rough set model where $\mathbf{C} = (C, \ni_C, \supseteq_C)$, $\mathbf{N} = (N, \ni_N, \supseteq_N)$, $\mathbf{O} = (O, \ni_O, \supseteq_O)$ be the well defined class, name, object structures, respectively. Let $OORS'(\mathbf{C}', \mathbf{N}', \mathbf{O}')$ be the sub-object-oriented rough set model, where $\mathbf{C}' = (C', \ni_C, \supseteq_C)$, $\mathbf{N}' = (N', \ni_N, \supseteq_N)$, $\mathbf{O}' = (O', \ni_O, \supseteq_O)$ be the well defined class, name, object structures respectively. Let $\mathbf{C}' \subseteq \mathbf{C}$, $\mathbf{N}' \subseteq \mathbf{N}$, $\mathbf{O}' \subseteq \mathbf{O}$, $OORS' \subseteq OORS$, $OORS'$ is called sub-object-oriented rough set model of $OORS$. Let $\rho(OORS)$ be the set of all sub-object-oriented rough set model of $OORS$, the feature core of sub-object-oriented rough set models in static Reduct is

$$CORE\ (OORS') = \bigcap_{i=1}^{p} C_i \quad \text{or} \quad \bigcap_{i=1}^{p} \bigcap_{j=1}^{q} C_i.n_j = \bigcap_{R \in RED(OORS')} R$$

(14)

Where $C_i\,(1 \le i \le p), C_i.n_j\,(1 \le i \le p), (1 \le j \le q)$ represents classes and classes with names belongs to reducts in the sub-object-oriented rough set model.

Definition 6: Let $OORS(\mathbf{C}, \mathbf{N}, \mathbf{O})$ be the object-oriented rough set model where $\mathbf{C} = (C, \ni_C, \supseteq_C)$, $\mathbf{N} = (N, \ni_N, \supseteq_N)$, $\mathbf{O} = (O, \ni_O, \supseteq_O)$ be the well defined class, name, object structures, respectively. Let $OORS'(\mathbf{C}', \mathbf{N}', \mathbf{O}')$ be the sub-object-oriented rough set model, where $\mathbf{C}' = (C', \ni_C, \supseteq_C)$, $\mathbf{N}' = (N', \ni_N, \supseteq_N)$, $\mathbf{O}' = (O', \ni_O, \supseteq_O)$ be the well defined class, name, object structures respectively. Let $\mathbf{C}' \subseteq \mathbf{C}$, $\mathbf{N}' \subseteq \mathbf{N}$, $\mathbf{O}' \subseteq \mathbf{O}$, $OORS' \subseteq OORS$, $OORS'$ is called sub-object-oriented rough set model of $OORS$. Let $\rho(OORS)$ be the set of all sub-object-oriented rough set model of $OORS$, $F \subseteq \rho(OORS)$ be the Dynamic Core of $OORS$ on a family F is defined by

$$DCORE(OORS, F) = CORE(OORS) \cap \bigcap_{OORS' \in F} CORE(OORS')$$

(15)

$DCORE(OORS, F)$ is called F-Dynamic Core of Object-Oriented Rough Set Model of $OORS$.

V. (F-λ)–Dynamic Core in the Object-Oriented Rough Set Model

F–dynamic core can be sometimes too much restrictive so here applies a generalization of F–dynamic core.

Definition 7:. Let $OORS(\mathbf{C}, \mathbf{N}, \mathbf{O})$ be the object-oriented rough set model where $\mathbf{C} = (C, \ni_C, \supseteq_C)$, $\mathbf{N} = (N, \ni_N, \supseteq_N)$, $\mathbf{O} = (O, \ni_O, \supseteq_O)$ be the well defined class, name, object structures, respectively. Let $\rho(OORS)$ be the set of all sub-object-oriented rough set model of OORS, $F \subseteq \rho(OORS)$ is called a family of object-oriented rough set model OORS, $\lambda \in (0.5, 1]$, and the $(F-\lambda)$-dynamic core of $OORS$ based on Family F is defined by

$$DCORE_\lambda(OORS, F) = \{C \text{ or } C.n \in CORE(OORS) \mid \{ OORS' \in F : C \text{ or } C.n \in CORE(OORS')\}\mid \text{----------} \ge \lambda \Big| F \Big|$$

λ is precision coefficient and the value of x decides which class or class with name belongs to (F-λ)-Dynamic Core $DCORE_\lambda(OORS, F)$ in the object-oriented rough set model. Let c represents class and c.n represents class with name as λ approaches 1, $DCORE_\lambda(OORS, F)$ will be closed to $DCORE(OORS, F)$, while λ approaches 0.5, $DCORE_\lambda(OORS, F)$ is more rough compared with $CORE(OORS, F)$, but $DCORE(OORS, F)$ will comprise classes or more classes with names.

VI. Generalized Dynamic Core in the Object-Oriented Rough Set Model

According to the definition of dynamic core in the object-oriented rough set model, if some feature classes or classes with names of any sub-object-oriented rough set models in F family are comprised by dynamic core, then it is certainly a feature classes or classes with names of object-oriented rough set model. This notion can be sometimes not convenient because we are interested in useful sets of classes or classes with names which are not necessarily reducts of the object-oriented rough set model. Therefore we have to generalize the notion of a dynamic core in the object-oriented rough set model.

Definition 8: Let $OORS(\mathbf{C}, \mathbf{N}, \mathbf{O})$ be the object-oriented rough set model where $\mathbf{C} = (C, \ni_C, \supseteq_C)$, $\mathbf{N} = (N, \ni_N, \supseteq_N)$, $\mathbf{O} = (O, \ni_O, \supseteq_O)$ be the well defined class, name, object structures, respectively. Let $\rho(OORS)$ be the set of all sub-object-oriented rough set model of OORS, $F \subseteq \rho(OORS)$ is called a family of object-oriented rough set model, then

$$GDCORE(OORS, F) = \bigcap_{OORS' \in F} CORE(OORS')$$

(16)

$GDCORE(OORS, F)$ is called the Generalized Dynamic Core the of object-oriented rough set model $OORS$.

VII. CONCLUSION

We have considered object-oriented rough sets, decision rules and discernibility matrix to find reducts in the object-oriented rough set model. We proposed dynamic reducts in the object-oriented rough set model. We also proposed dynamic core and generalized core in the object-oriented rough set model. We have developed algorithms for dynamic reducts and core.

REFERENCES

[1] Pawlak, Z, "Rough sets." International Journal of Computer and Information Science **11,** 1982, pp.341–356.

[2] Pawlak, Z, "Rough Sets:" Theoretical Aspects of Reasoning about Data. Kluwer, Academic Publisher 1991.

[3] Popkorn, S, "First Steps in Model Logic." Cambridge University Press, 1994.

[4] Budd, T.A, "An Introduction of Object– Oriented Programming," 2nd Edition. Addison Wesley Longman, 1997.

[5] Bazan J, "A Comparison of Dynamic and Non-Dynamic Rough Set Methods for Extracting Laws from Decision Tables[C], In Polkowski and Skowron, (eds) Rough sets in Knowledge Discovery 1:Methodology and Applications, Physica-Verlag, Heidelberg,1998, pp. 321–365.

[6] Kudo, Y., Murai, T, "A theoretical formulation of object–oriented rough set models", Journal of Advanced Computational Intelligence and Intelligent Informatics 10(5),2006, pp.612-620.

[7] Kudo, Y., Murai, T, "A Method of Generating Decision Rules in Object–Oriented Rough Set Models." In:Greco, S.,Hata,Y,, Hirano,S., Inuiguchi,M., Miyamoto,S., Nguyen., Slowinski ,R,(eds) RSCTC 2006, LNCS (LNAI), Springer, Heidelberg, vol.4259,2006, pp.338-347.

[8] Kudo, Y., Murai, T, "A note on treatment of incomplete information in object–oriented rough sets." In: Proceedings of the Joint 3rd International Conference on Soft Computing and Intelligent Systems and 7th International Symposium on advanced Intelligent Systems. G.Wang et.al(Ede),RSKT ,LNAI 5009,2008, pp.115-123.