The Concept of Reducts in Pawlak Three-Step Rough Set Analysis

Yiyu Yao and Rong Fu

Department of Computer Science, University of Regina
Regina, Saskatchewan, Canada S4S 0A2
{yyao,fu207}@cs.uregina.ca

Abstract. Rough set approaches to data analysis involve removing redundant attributes, redundant attribute-value pairs, and redundant rules in order to obtain a minimal set of simple and general rules. Pawlak arranges these tasks into a three-step sequential process based on a central notion of reducts. However, reducts used in different steps are defined and formulated differently. Such an inconsistency in formulation may unnecessarily affect the elegancy of the approach. Therefore, this paper introduces a generic definition of reducts of a set, uniformly defines various reducts used in rough set analysis, and examines several mathematically equivalent, but differently formulated, definitions of reducts. Each definition captures a different aspect of a reduct and their integration provides new insights.

Keywords: Pawlak three-step analysis, reducts, rough set analysis.

1 Introduction

In his seminal book, *Rough Sets: Theoretical Aspects of Reasoning About Data*, Pawlak [13] provided a simple and elegant method for analyzing data represented in a tabular form. The method can be applied to decision table simplification and rule learning. In our previous paper [21], with a slightly different formulation, we reviewed Pawlak approach and examined several of its variations. More specifically, we introduced a generic notion of a reduct of a set and an explicit expression of a concept by a pair of intension and extension of the concept. We formulated Pawlak approach as a three-step method for analyzing an information table. Our objective was to show that the three steps use three types of reducts, namely, attribute reducts of the table with respect to decision attributes, attribute reducts of an object with respect to decision attributes, and rule reducts. However, due to space limitation, we were only able to provide an outline of our argument. The objective of this paper is to expand our outline into a more complete and thorough investigation.

This paper is different from and complementary to many other studies. It is not intent on proposing a new method nor comparing different methods. The main contribution is to provide a new interpretation of Pawlak approach to data analysis. Our formulation starts with a generic notion of reducts of a set and an

J.F. Peters et al. (Eds.): Transactions on Rough Sets XVI, LNCS 7736, pp. 53–72, 2013.

explicit expression of a concept as a pair of a logic formula (i.e., intension of the concept) and a set of objects (i.e., the extension of the concept) [18]. We hope that a reformulation and reinterpretation of Pawlak three-step approach will offer several new insights. The generic notion of reducts unifies the three steps and demonstrates the simplicity and reflexibility of Pawlak approach. Instead of using several forms and definitions of reducts, we use only one general form and one definition. The explicit expression of concepts, in terms of logic formulas as intensions and subsets of objects as extensions, offers new understanding of reducts and rules. In summary, our reformulation, based on a single notion and a uniform exposition, aims at showing the simplicity, elegancy, and flexibility of Pawlak three-step approach at a conceptual level, rather than demonstrating its efficiency at an implementation level. This allows us to focus on the definition of reducts, instead of designing methods for constructing reducts.

For simplicity and clarity, we restrict our discussion to the basic notion of reducts in Pawlak's book. As future work, it will be interesting and worthwhile to investigate if the same argument, with some modifications, can be applied to various generalized notions of reducts, including, dynamic reducts [1], association reducts [16], approximate reducts [11], decision bireducts [17], and many others [5,8,9,10,19,24]. The restriction allows us to concentrate on the basic issues without being distracted by minute details of various generalizations. The results of this paper can be used to relate Pawlak approach to other standard rule learning algorithms, such as partition-based decision-tree methods [15,25] and covering-based sequential covering methods [2,3,4,6,7,26]. While other methods focus mainly on rule learning algorithms, Pawlak approach emphasizes on a study of intrinsic properties of rules independent of a particular rule learning algorithm [14].

The rest of this paper is organized as follows. Section 2 presents an overview of rough set analysis and explicitly expresses such an analysis into a sequential three-step process. Section 3 introduces a general definition of a reduct of a set, examines a simpler definition of a reduct when a monotonic evaluation is used, and investigates an $\cap$-reduct and an $\cup$-reduct of a family of subsets of a set. Section 4 is a critical analysis of Pawlak three-step approach. Based on the generic definition of a reduct of a set introduced in Section 3, we study about twenty different definitions of reducts used in rough set analysis. Each definition interprets a reduct from an unique angle and, pooling together, all interpretations provide new insights.

2 An Overview of Pawlak Rough Set Analysis

Rough set analysis (RSA) deals with a finite set of objects called the universe, in which each object is described by values of a finite set of attributes. In his book, Pawlak first used a subset of objects to represent a concept and a partition of the universe to represent a classification at an abstract level. More specifically, he called subsets categories, a partition or equivalently an equivalence relation (classification) knowledge and a family of equivalence relations a knowledge base.

Those notions were later explained by using an information table. Although such a formulation, from abstract notions to concrete examples, provides a more general framework, the meanings of various notions are not entirely clear when they are introduced. For this reason, we start our formulation by directly referring to an information table.

Definition 1. *An information table is the following tuple:*

$$S = (U, At, \{V_a \mid a \in At\}, \{I_a \mid a \in At\}),$$

where U is a finite nonempty set of objects called the universe, At is a finite nonempty set of attributes, V_a is a nonempty set of values for $a \in At$, and $I_a : U \longrightarrow V_a$ is a complete information function that maps an object of U to exactly one value in V_a.

For an object $x \in U$, $I_a(x)$ denotes the value of x on attribute $a \in At$. For notational simplicity, for a subset of attributes $A \subseteq At$, $I_A(x)$ denotes the vector value of x on A.

Definition 2. *A classification table or a decision table is an information table $S = (U, At = C \cup D, \{V_a \mid a \in At\}, \{I_a \mid a \in At\})$, where C is a set of condition attributes and D is a set of classification or decision attributes. If for all objects $x, y \in U$, $I_C(x) = I_C(y)$ implies that $I_D(x) = I_D(y)$, the table is called a consistent classification table, and is called an inconsistent table otherwise.*

An information table provides all available information about a set of objects. We analyze attributes and objects based on the information functions in the table. Pawlak investigated three main tasks of rough set analysis and presented them in a sequential three steps [13], as shown in Figure 1. We use a naming system that is slightly different from the one used in Pawlak's book. More specifically, we use "attribute reduction" and "attribute reduct" instead of "knowledge reduction" and "reduct of knowledge," respectively, and use "attribute-value-pair reduction" and "attribute-value-pair reduct" instead of "reduction of categories" and "value reducts," respectively.

The first step analyzes attribute dependencies with an objective to simplifying a table. The main tasks involving identifying superfluous (i.e., dispensable) attributes and finding a minimal subset of attributes that preserves the same information as the entire set of attributes for the purpose of classification. Such a minimal set of attributes is called an attribute reduct of the table or a relative attribute reduct of a classification table. There may exist more than one reduct for each table. With respect to a reduced table with a minimal set of attributes in a decision table, we can construct a set of decision rules. The left-hand-side of each decision rule is a conjunction of a set of attribute-value pairs.

The second step analyzes dependencies of attribute values with an objective to simplifying a decision rule. Similar to the notion of superfluous attribute in a table, there may exist superfluous attribute-value pairs in the left-hand-side of a decision rule. The main tasks of the second step are to identify superfluous

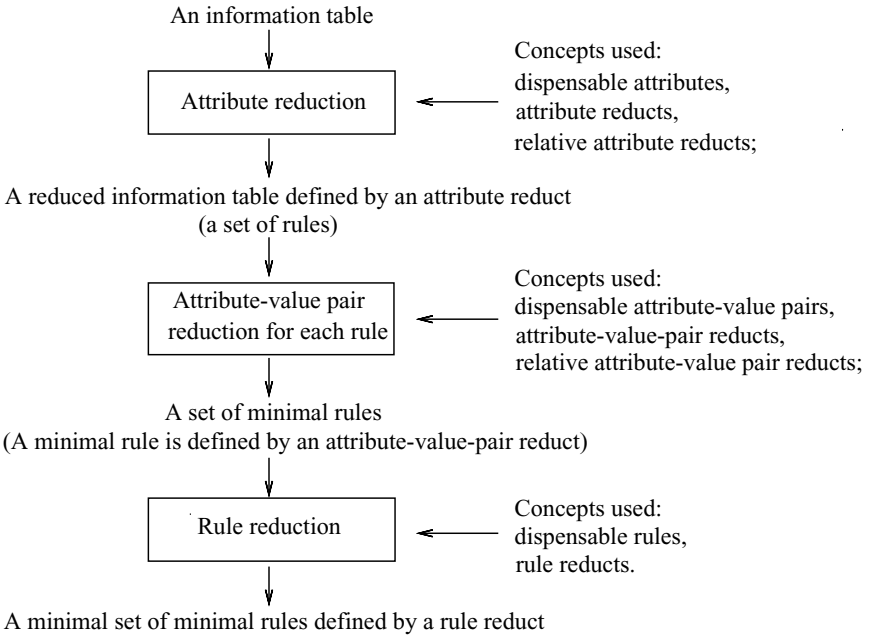

Fig. 1. Pawlak three-step rough set analysis

attribute-value pairs and to derive a minimal set of attribute-value pairs for each decision rule. A minimal set of attribute-value pairs is called a relative attribute-value-pair reduct. Again, there may exist more than one reduct. The result of the second step is a set of minimal decision rules.

The third step analyzes dependencies of decision rules with an objective to simplifying a set of decision rules. There may exist superfluous (i.e., dispensable) rules in the set of decision rules obtained in the second step. By removing superfluous rules, one can obtain a minimal set of rules called a rule reduct.

In Pawlak's book, the three steps are clearly separated. As pointed out by a reviewer of this paper, the steps of attribute reduction, attribute-value reduction and rule reduction do not need to follow each other. In applications they may occur optionally or independently. For example, attribute-value reduction may be treated as a special case of attribute reduction, which leads toward merging the first two above-mentioned steps together. In some cases, rule reduction may be avoided; one may simply use techniques based on voting to deal with redundant or conflicting rules.

Although each of the three steps involves different entities or subjects, they share high-level similarities. All analyze relationships between entities with an objective to make simplification by removing superfluous entities. More importantly, the result of simplification is a reduct, namely, an attribute reduct of a table, an attribute-value-pair reduct of a rule, and a rule reduct of a set of

rules. This observation suggests that one may unify the three steps. However, in rough set literature, different forms and definitions are given for different types of reducts. In rest of this paper, we present a generic definition of reducts and show that Pawlak three-step analysis may be uniformly formulated based on a generic definition of reducts.

3 A General Definition of Reducts

We introduce a general definition of reducts of a set and examine basic properties of reducts.

3.1 Reducts of a Set

Reducts are a fundamental notion of rough set analysis. As showed in the last section, different types of reducts have been proposed and studied. Consider an attribute reduct of a table, intuitively speaking, an attribute reduct is a subset of attributes that preserve the same information or property as the entire set of attributes (i.e., the sufficiency condition) and at the same time contains no superfluous attributes (i.e., non-redundancy condition). This interpretation of an attribute reduct can be generalized into a generic definition of a reduct of any set. First, we specify a property such that the entire set has the property. Then, we state the sufficiency and non-redundancy conditions on a subset of the set for it to be a reduct. The sufficiency condition suggests that a reduct has the same property as the entire set. The non-redundancy condition requires that a reduct must be a minimal subset having the property.

Definition 3. *Suppose S is a finite set and 2^S is the power set of S. Let $\mathbb{P}$ denote a unary predicate on subsets of S, that is, for $X \subseteq S, \mathbb{P}(X)$ stands for the statement that "subset X has the property $\mathbb{P}$." An evaluation e of $\mathbb{P}$ is understood as a truth assignment for every subset of S: $\mathbb{P}_e(X)$ is true if X has property $\mathbb{P}$, otherwise, it is false.*

An evaluation typically depends on a particular data set. For example, an evaluation of subsets of attributes is determined by a particular information table. A reduct of S is therefore defined with respect to a given evaluation. We use a subscript e to explicitly denote the evaluation.

Definition 4. *Given an evaluation e of $\mathbb{P}$, A subset $R \subseteq S$ is called a reduct of S if it satisfies the following conditions:*

$$\begin{aligned} &\text{(w)} \quad \mathbb{P}_e(S), \\ &\text{(s)} \quad \mathbb{P}_e(R), \\ &\text{(n)} \quad \forall B \subset R, (\neg\mathbb{P}_e(B)). \end{aligned}$$

Condition (w) requires that the whole set S must have the property $\mathbb{P}$. In many studies, this condition is typically implicitly assumed or embedded in $\mathbb{P}$. It ensures that a reduct of S exists. Condition (s) is a sufficiency condition, stating

that a reduct R of S is sufficient for preserving property $\mathbb{P}$ of S. Condition (n) is a non-redundancy condition, indicating that none of the proper subsets of R has the property.

According to Definition 4, it is necessary to check all proper subsets of R in order to verify if R is a reduct. This imposes an unpractical computational constraint. In many situations, one can study a special class of property $\mathbb{P}$ that satisfies the monotonicity with respect to set inclusion.

Definition 5. *A predicate $\mathbb{P}$ is said to be monotonic with respect to set inclusion if it satisfies the following property:*

$$\forall A, B \subseteq S, (A \subseteq B \Longrightarrow (\mathbb{P}(A) \Longrightarrow \mathbb{P}(B))). \tag{1}$$

The monotonicity states that if a subset has a property, then a superset of it also has the property. It is important to point out that, unlike the definition of a reduct, the monotonicity of $\mathbb{P}$ is defined based on all possible evaluations. That is, the monotonicity must hold for all possible evaluations. In terms of information tables, each table determines an evaluation and all possible tables determine all possible evaluations. The monotonicity of a predicate must hold for all possible information tables. The monotonicity can be equivalently re-expressed as

$$\forall A, B \subseteq S, (A \subseteq B \Longrightarrow (\neg\mathbb{P}(B) \Longrightarrow \neg\mathbb{P}(A))). \tag{2}$$

That is, if a set does not have the property, then none of its subsets has the property. Thus, once we know that a set does not have the property, we do not need to check its subsets. This leads to a simplified definition of reducts.

Definition 6. *Suppose $\mathbb{P}$ satisfies monotonicity. Given an evaluation e of $\mathbb{P}$, a subset $R \subseteq S$ is called a reduct of S if it satisfies the following properties:*

$$\begin{array}{ll} \text{(w)} & \mathbb{P}_e(S) \\ \text{(s)} & \mathbb{P}_e(R) \\ \text{(n)} & \forall a \in R, (\neg\mathbb{P}_e(R - \{a\})) \end{array}$$

Condition (n) shows that each element $a \in R$ is necessary. That is, elements of R are individually necessary. With the monotonicity, a verification of a reduct becomes easier, one only needs to check individual elements from S based on condition (n) instead of all subsets of R. A reduct is always defined with respect to a particular evaluation. In the rest of this paper, for notational simplicity we sometimes omit the subscript e by simply writing $\mathbb{P}_e(X)$ as $\mathbb{P}(X)$ for subset $X \subseteq S$. It may be commented that many definitions of reducts in rough set theory obey the monotonicity.

In the study of reducts, there are two additional important notions. The first one is superfluous or redundant elements and the second one is core elements. The concept of superfluous element is only applicable when considering monotonic evaluations. One can also define generic notions of redundant elements and core elements.

Definition 7. *Suppose $\mathbb{P}$ satisfies monotonicity. Given an evaluation e of $\mathbb{P}$, an element a is called a redundant element if it satisfies the following properties:*

$$\begin{aligned} &(\text{r1}) \quad \mathbb{P}_e(S) \\ &(\text{r2}) \quad \mathbb{P}_e(S - \{a\}) \end{aligned}$$

Condition (r2) states that when removing an element a from the set S, the rest elements of the set still satisfy the property $\mathbb{P}$. That is to say, the element a is unnecessary and dispensable for preserving the property $\mathbb{P}$ and one can have a same result without considering the element a in S. Therefore, we say a is redundant in S.

Definition 8. *Suppose $\mathbb{P}$ satisfies monotonicity. Given an evaluation e of $\mathbb{P}$, an element a is called a core element if it satisfies the following properties:*

$$\begin{aligned} &(\text{c1}) \quad \mathbb{P}_e(S) \\ &(\text{c2}) \quad \neg\mathbb{P}_e(S - \{a\}) \end{aligned}$$

That is, for a set S satisfying property $\mathbb{P}$, if we remove element a from S, the rest elements can no longer preserve property $\mathbb{P}$. Therefore, the element a is necessary and indispensable for keeping property $\mathbb{P}$.

Definition 9. *Given a set S, let $RED(S)$ denote the family of all reducts of S, the set of core elements of S can be defined as follows:*

$$CORE(S) = \bigcap RED(S). \tag{3}$$

The $CORE$ is the intersection of all reducts, in other words, elements in $CORE$ are included in every reduct. Therefore, the $CORE$ is the most important subset that none of its elements can be eliminated for preserving a specific property.

3.2 Reducts of a Family of Subsets of a Set

The proposed definition of reducts is flexible. As an example, we consider a set S whose elements are subsets of a set. Given a set W, suppose $S \subseteq 2^W$ is a family of subsets of W. According to definition of reducts in Definition 6, we introduce $\cap$-reducts and $\cup$-reducts of S.

Definition 10. [13] *Suppose W is a finite set and $S \subseteq 2^W$. A set $R \subseteq S$ is called an $\cap$-reduct of S if it satisfies the following conditions:*

$$\begin{aligned} &(\text{w}) \quad \cap S = \cap S, \\ &(\text{s}) \quad \cap R = \cap S, \\ &(\text{n}) \quad \forall a \in R, (\neg(\cap(R - \{a\}) = \cap S)). \end{aligned}$$

A set $Q \subseteq S$ is called an $\cup$-reduct of S if it satisfies the following conditions:

$$\begin{aligned} &(\text{w}') \quad \cup S = \cup S, \\ &(\text{s}') \quad \cup Q = \cup S, \\ &(\text{n}') \quad \forall a \in Q, (\neg(\cup(Q - \{a\}) = \cup S)). \end{aligned}$$

Conditions (w) and (w′) simply state that S has the property. We explicit list them to show the connection to Definition 6.

In some situations, we want to use a family of subset S to represent other subsets of W. This leads to a definition of relative reducts.

Definition 11. [13] *Suppose $S \subseteq 2^W$ is a family of subsets of a finite set W and $T \subseteq W$ is a subset of W. An $\cap$-reduct of S relative to T, or simply an $\cap$-relative-reduct is defined by the following conditions,*

$$\begin{aligned} &(\text{w}) \quad \cap S \subseteq T, \\ &(\text{s}) \quad \cap R \subseteq T, \\ &(\text{n}) \quad \forall a \in R, (\neg(\cap(R - \{a\}) \subseteq T)). \end{aligned}$$

Condition (w) states that the family S has the property of $\cap S \subseteq T$. We will show later that those reducts form a basis of rough set analysis.

4 A Critical Analysis of Pawlak Three-Step Approach

In this section, we provide a critical analysis of Pawlak three-step approach based on the notion of reducts introduced in the last section.

4.1 Rough Set Approximations

Rough set theory analyzes an information table based on equivalence relations (i.e., reflexive, symmetric and transitive relations) induced by subsets of attributes [12,13].

Definition 12. [13] *Given an information table, a subset of attributes $A \subseteq At$ defines an equivalence relation on U as follows:*

$$\begin{aligned} xE_Ay &\Longleftrightarrow \forall a \in A, (I_a(x) = I_a(y)) \\ &\Longleftrightarrow I_A(x) = I_A(y). \end{aligned} \tag{4}$$

That is, x and y are equivalent if and only if they have the same values on all attributes in A. The equivalence relation E_A induces a partition of the universe and is denoted by $U/E_A = \{[x]_{E_A} \mid x \in U\}$, where $[x]_{E_A} = \{y \mid xE_Ay\}$ is the equivalence class containing x.

There is a one-to-one correspondence between all equivalence relations on U and all partitions of U. Therefore, we use equivalence relations and partitions interchangeably. An equivalence relation E is a set of pairs, that is, $E \subseteq U \times U$, where $U \times U$ is the cartesian product of U and U. One can apply set-theoretic operations and relations on equivalence relations. If E_1 and E_2 are two equivalence relations, then $E_1 \cap E_2$ is also an equivalence relation.

The standard set inclusion of equivalence relations defines a partial order on partitions as follows: for two equivalence relations E_1 and E_2,

$$U/E_1 \preceq U/E_2 \Longleftrightarrow E_1 \subseteq E_2. \tag{5}$$

If $U/E_1 \preceq U/E_2$, each block of U/E_2 must be a union of some blocks of U/E_1, and the partition U/E_1 is called a refinement of U/E_2 and U/E_2 a coarsening of U/E_1.

With respect to different subsets of attributes, we can establish the following relationships, for $A, B \subseteq At, x \in U$,

$$\begin{aligned}
(1) \qquad & E_A = \bigcap_{a \in A} E_{\{a\}}, \\
& E_{A \cup B} = E_A \cap E_B, \\
(2) \qquad & [x]_{E_A} = \bigcap_{a \in A} [x]_{E_{\{a\}}}, \\
& [x]_{E_{A \cup B}} = [x]_{E_A} \cap [x]_{E_B}, \\
(3) \qquad & A \subseteq B \Rightarrow E_B \subseteq E_A, \\
& A \subseteq B \Rightarrow U/E_B \preceq U/E_A. \qquad (6)
\end{aligned}$$

Properties (1) and (2) show that the equivalence relation, or the corresponding partition, defined by a subset of attributes can be constructed from the individual equivalence relations or partitions defined by singleton subsets of attributes. Property (3) states that the refinement-coarsening relation $\preceq$ is monotonic with respect to set inclusion of sets of attributes.

Consider the equivalence relation E_A defined by a subset of attributes $A \subseteq At$. For a subset $X \subseteq U$, its lower and upper approximations are defined by [12,13]:

$$\begin{aligned}
\underline{apr}(X) &= \bigcup \{[x]_{E_A} \in U/E_A \mid [x]_{E_A} \subseteq X\}; \\
\overline{apr}(X) &= \bigcup \{[x]_{E_A} \in U/E_A \mid [x]_{E_A} \cap X \neq \emptyset\}. \qquad (7)
\end{aligned}$$

That is, the lower approximation $\underline{apr}(X)$ is the union of those equivalence classes that are subsets of X, and the upper approximation $\overline{apr}(X)$ is the union of those equivalence classes that have nonempty intersection with X. By the lower and upper approximation, one can divide the universe U into three pair-wise disjoint regions [12], namely, the positive region $\mathrm{POS}(X)$, the boundary region $\mathrm{BND}(X)$, and the negative region $\mathrm{NEG}(X)$:

$$\begin{aligned}
\mathrm{POS}(X) &= \underline{apr}(X), \\
\mathrm{BND}(X) &= \overline{apr}(X) - \underline{apr}(X), \\
\mathrm{NEG}(X) &= U - \overline{apr}(X) = (\overline{apr}(X))^c, \qquad (8)
\end{aligned}$$

where $(\cdot)^c$ denotes the set complement. Some of these regions may be empty. The pair of lower and upper approximations and the three regions uniquely define each other. One can formulate the theory of rough sets by using any one of them.

Based on these notions, we are ready to review Pawlak three-step approach to data analysis.

4.2 Step 1: Analysis of Attribute Dependencies

Pawlak refers to partitions, or equivalently equivalence relations, defined by subsets of attributes as classification knowledge or simply classification. Analysis of

attribute dependencies is performed through equivalence relations defined by subsets of attributes.

Reducts of an Information Table. Consider first the notion of reducts of an information table. Pawlak introduces the notion of a reduct of a family of partitions or equivalence relations. Since each attribute defines an equivalence relation, we can use a Pawlak reduct of a family of equivalence relations to define an attribute reduct of an information table.

Definition 13. *Given an information table, consider the family of equivalence relations defined by singleton subsets of attributes* $S = \{E_{\{a\}} \mid a \in At\}$. *A reduct of* S *is defined as a subset* $R \subseteq S$ *satisfying the following conditions:*

$$\begin{aligned} &\text{(s1)} \quad \cap R = \cap S, \\ &\text{(n1)} \quad \forall E \in R, (\neg(\cap(R - \{E\}) = \cap S)). \end{aligned} \tag{9}$$

In this definition, the condition $\cap S = \cap S$ is not explicitly given. Recall that an equivalence relation is a set of pairs. It follows that $S \subseteq 2^{U \times U}$. Therefore, according to Definition 10, a reduct as defined by Definition 13 is in fact an $\cap$-reduct of S.

There is only a small problem when characterizing an information table by the family of equivalence relations $\{E_{\{a\}} \mid a \in At\}$. Two different attributes $a, b \in At$ may define the same equivalence relation, that is, $E_{\{a\}} = E_{\{b\}}$. To resolve the problem, Pawlak treats all those attributes that define the same equivalence relation as one attribute. According to Definition 6, the following definition resolves this problem by directly referring to the set of attributes At.

Definition 14. *In an information table, an attribute reduct is a subset of attributes* $R \subseteq At$ *satisfying each of the following equivalent pairs of conditions:*

$$\begin{aligned} &\text{equivalence relation based conditions :} \\ &\quad \text{(s2)} \quad E_R = E_{At}, \\ &\quad \text{(n2)} \quad \forall a \in R, (\neg(E_{R-\{a\}} = E_{At})); \\ &\text{partition based conditions :} \\ &\quad \text{(s3)} \quad U/E_R = U/E_{At}, \\ &\quad \text{(n3)} \quad \forall a \in R, (\neg(U/E_{R-\{a\}} = U/E_{At})); \\ &\text{equivalence class based conditions :} \\ &\quad \text{(s4)} \quad \forall x \in U, ([x]_{R_R} = [x]_{E_{At}}), \\ &\quad \text{(n4)} \quad \forall a \in R \exists x \in U, (\neg([x]_{E_{R-\{a\}}} = [x]_{E_{At}})). \end{aligned} \tag{10}$$

The definition contains both commonly used conditions based on equivalence relations or partitions and new conditions based on equivalence classes. Each pair of conditions provides a different characterization and understanding of a reduct. That is, a reduct is a minimal set of attributes that defines the same equivalence relation as E_{At}. The last pair of conditions is particularly interesting

and is closely related to the conditions for defining attribute-value-pair reducts. Again, the first condition of a general reduct is not explicitly stated. For example, we omit the condition $E_{At} = E_{At}$. By Definition 6, an attribute reduct is an exapmle of a reduct of the set of attributes At.

Relative Reducts of a Consistent Classification Table. For analyzing a classification table, Pawlak introduces the notion of a relative reduct. Based on an equivalence relations defined by subsets of attributes, a classification table with $At = C \cup D$ is consistent if

$$E_C \subseteq E_D, \textit{or equivalently } U/E_C \preceq U/E_D. \tag{11}$$

For a consistent classification table, similar to Definition 13, a relative reudct can be defined according to Definition 11.

Definition 15. *Consider the set of all equivalence relations defined by singleton subsets of condition attributes $S = \{E_{\{a\}} \mid a \in C\}$. A reduct of S relative to E_D is a subset $R \subseteq S$ satisfying the following properties:*

$$\begin{aligned} &\text{(w)} \quad \cap S \subseteq E_D, \\ &\text{(s)} \quad \cap R \subseteq E_D, \\ &\text{(n)} \quad \forall E \in R, (\neg(\cap(R - \{E\}) \subseteq E_D)). \end{aligned} \tag{12}$$

Similar to Definition 14, a relative attribute reduct can also be equivalently defined by using equivalence relations, partitions, and equivalence classes, respectively.

Definition 16. *Given a consistent classification table $S = (U, At = C \cup D, \{V_a \mid a \in At\}, \{I_a \mid a \in At\})$, a subset $R \subseteq C$ is called a reduct of C relative to D, or simply a relative reduct, if R satisfies one of the following equivalent pairs of conditions:*

$$\begin{aligned} &\text{equivalence relation based conditions :} \\ &\quad \text{(s5)} \quad E_R \subseteq E_D, \\ &\quad \text{(n5)} \quad \forall a \in R, (\neg(E_{R-\{a\}} \subseteq E_D)); \\ &\text{partition based conditions :} \\ &\quad \text{(s6)} \quad U/E_R \preceq U/E_D;, \\ &\quad \text{(n6)} \quad \forall a \in R, (\neg(U/E_{R-\{a\}} \preceq U/E_D)); \\ &\text{equivalence class based conditions :} \\ &\quad \text{(s7)} \quad \forall x \in U, ([x]_{E_R} \subseteq [x]_{E_D}), \\ &\quad \text{(n7)} \quad \forall a \in R \exists x \in U, (\neg([x]_{E_{R-\{a\}}} \subseteq [x]_{E_D})). \end{aligned} \tag{13}$$

Conditions in the definition suggest that a relative reduct R is a minimal set of attributes whose partition U/E_R is the same or finer than E_D. For example, condition (s7) states that the equivalence class of E_R containing x is a subset

of the equivalence class of E_D containing x. That means that one can infer the equivalence class $[x]_{E_D}$ from the equivalence class $[x]_{E_R}$ so that it preserves the descriptive ability for classification. Condition (n7) states that any attribute in R is necessary for inferring $[x]_{E_D}$.

Relative Reducts of an Inconsistent Classification Table. For an inconsistent classification table, Pawlak defines a relative reduct by using the positive region of the classification U/E_D induced by E_C:

$$\begin{aligned}\mathrm{POS}_{E_C}(U/E_D) &= \bigcup\{\mathrm{POS}_{E_C}(K) \mid K \in U/E_D\} \\ &= \bigcup\{\underline{apr}_{E_C}(K) \mid K \in U/E_D\}. \end{aligned} \tag{14}$$

More specifically, a relative reduct of an inconsistent classification table is a minimal set of attributes that preserves the positive region of U/E_D; there is not consideration of objects not in the positive region.

Definition 17. [13] *Given a consistent classification table $S = (U, At = C \cup D, \{V_a \mid a \in At\}, \{I_a \mid a \in At\})$, a subset $R \subseteq C$ is called a reduct of C relative to D if R satisfies the two conditions:*

$$\begin{aligned} &\text{(s8)} \quad \mathrm{POS}_{E_R}(U/E_D) = \mathrm{POS}_{E_C}(U/E_D), \\ &\text{(n8)} \quad \forall a \in R, (\neg(\mathrm{POS}_{E_{R-\{a\}}}(U/E_D) = \mathrm{POS}_{E_C}(U/E_D))). \end{aligned}$$

For a consistent classification table, we have $\mathrm{POS}_{E_C}(U/E_D) = U$. Pawlak's definition is therefore applicable to both consistent and inconsistent classification tables.

Although Pawlak's definition is an example of a relative reduct of the set of condition attribute C, it is very different in form from the definition of a relative reduct of a consistent classification table as given by Definitions 15 and 16. By insisting on having the same positive region, a relative reduct does not care about objects in the boundary region. This observation provides a hint: one can transform an inconsistent table into a consistent table by focusing on individual positive regions of equivalence classes of E_D so that the definition of a relative reduct of a consistent table can be used. According to the positive regions of equivalence classes in U/E_D, we can form the following partition:

$$\{\mathrm{POS}_{E_C}(K) \neq \emptyset \mid K \in U/E_D\} \cup (\{\cup\{\mathrm{BND}_{E_C}(K) \mid K \in U/E_D\}\} - \{\emptyset\}). \tag{15}$$

Suppose $E_{D'}$ is the equivalence relation corresponding to this partition, and the partition can be denoted by $U/E_{D'}$. According to the partition $U/E_{D'}$, a relative reduct of an inconsistent table can be defined based on Definition 16.

Definition 18. *Given a consistent classification table $S = (U, At = C \cup D, \{V_a \mid a \in At\}, \{I_a \mid a \in At\})$, a subset $R \subseteq C$ is called a reduct of C relative to D if R satisfies any of the following equivalent pairs of conditions:*

equivalence relation based conditions :

(s9) $E_R \subseteq E_{D'}$,

(n9) $\forall a \in R, (\neg(E_{R-\{a\}} \subseteq E_{D'}))$;

partition based conditions :

(s10) $U/E_R \preceq U/E_{D'}$,

(n10) $\forall a \in R, (\neg(U/E_{R-\{a\}} \preceq U/E_{D'}))$;

equivalence class based conditions :

(s11) $\forall x \in U, ([x]_{E_R} \subseteq [x]_{E_{D'}})$,

(n11) $\forall a \in R \exists x \in U, (\neg([x]_{R-\{a\}} \subseteq [x]_{E_{D'}}))$. (16)

A consistent table is a special case of inconsistent tables. For a consistent table, $U/E_{D'} = U/E_D$ and, hence, the definition is also valid for a consistent table. An advantage of Definition 18 is that it is in a uniform format as relative of a consistent table. However, we need to construct a partition $U/E_{D'}$ originally not in the table. If we examine the Pawlak definition again, we find that keeping the positive region is equivalent to saying that $\forall x \in U, ([x]_{E_C} \subseteq [x]_{E_D} \Longrightarrow [x]_{E_R} \subseteq [x]_{E_D})$. Based on this observation, we can have another definition of a relative reduct of an inconsistent table.

Definition 19. *Given an inconsistent classification table* $S = (U, At = C \cup D, \{V_a \mid a \in At\}, \{I_a \mid a \in At\})$*, a subset* $R \subseteq C$ *is called a reduct of* C *relative to* D *if* R *satisfies the two conditions:*

(s12) $\forall x \in U, ([x]_{E_C} \subseteq [x]_{E_D} \Longrightarrow [x]_{E_R} \subseteq [x]_{E_D})$,

(n12) $\forall a \in R \exists x \in U, (\neg([x]_{E_C} \subseteq [x]_{E_D} \Longrightarrow [x]_{E_{R-\{a\}}} \subseteq [x]_{E_D}))$. (17)

For a consistent table, $[x]_{E_C} \subseteq [x]_{E_D}$ is true for all $x \in U$. In this case, conditions (s12) and (n12) are equivalent to conditions (s7) and (n7). Therefore, Definition 19 is also valid for a consistent table. An advantage of this definition is that we do not need to introduce any extra structures not given in the table.

Relative Reducts and Attribute Dependencies. The relationship between the set of condition attributes C and the set of decision attributes D can be easily extended to a study of dependency of any two sets of attributes in an information table.

Consider two arbitrary subsets of attributes $A, B \subseteq At$ in an information table. The two subsets may have an nonempty intersection. If $E_A \subseteq E_B$ holds, we say that B depends on A. In this paper, we only consider two sets of attributes with a full dependency. This is similar to a consistent classification table with $E_C \subseteq E_D$. By applying the results of a relative reduct of a consistent classification table, it is straightforward to define a reduct of A relative to B for a simplified attribute dependency.

Definition 20. *For a pairs of subsets of attributes* $A, B \subseteq At$ *in an information table with* $E_A \subseteq E_B$*, a subset* $R \subseteq A$ *is called a reduct of* A *relative to* B *if it satisfies the following conditions:*

$$\begin{aligned} &\text{(s13)} \quad E_R \subseteq E_B, \\ &\text{(n13)} \quad \forall a \in R, (\neg(E_{R-\{a\}} \subseteq E_B)). \end{aligned} \tag{18}$$

With this definition, we can interpret a reduct $R \subseteq At$ of an information table as a relative reduct with respect to the entire set of attributes At. Relative reducts of a consistent classification table are also special cases.

Attribute dependencies can be formally studied through attribute-level rules in a table [23]. In this way, we can unify notions of attribute reducts and attribute-value-pair reducts in a common framework of rules. For such a purpose, we need to introduce a decision logic language $\mathcal{L}_{\mathcal{A}}$ similar to the one used in Pawlak's book.

Definition 21. *In an information table, a decision logic language $\mathcal{L}_A$ is recursively defined as follows: an atomic formula is given by $=_a$, where $a \in At$. If p and q are formulas, then $p \wedge q$ is a formula.*

By using language $\mathcal{L}_A$, we can express attribute dependency $E_A \subseteq E_B$ as,

$$\bigwedge_{a \in A} =_a \quad \rightarrow \quad \bigwedge_{b \in B} =_b, \tag{19}$$

or simply,

$$=_A \quad \rightarrow \quad =_B, \tag{20}$$

where both the left-hand-side and right-hand-side of $\rightarrow$ are formulas of $\mathcal{L}_A$. Consequently, finding a relative reduct can be viewed as searching for a minimal set of atomic formulas on the left-hand-side of an attribute-dependency rule.

The meaning of formulas of $\mathcal{L}_A$ are given by pairs of objects. More specifically, a pair of objects (x, y) is said to satisfy an atomic formula $=_a$ if and only if $I_a(x) = I_a(y)$. In general, the meanings of formulas can be recursively defined.

Definition 22. *The meanings of formulas of $\mathcal{L}_A$ are recursively computed as follows:*

$$\begin{aligned} m(=_a) &= \{(x, y) \in U \times U \mid I_a(x) = I_a(y)\}, \\ m(p \wedge q) &= m(p) \cap m(q). \end{aligned} \tag{21}$$

A formula may be interpreted as the intension of a concept and the meanings set is the extension of the concept. In this way, we express a concept jointly by a pair of a formula and a set. A concept in the context of attribute-level rules is an equivalence relation. By definition, it follows that,

$$\begin{aligned} m(=_a) &= E_{\{a\}}, \\ m(\bigwedge_{a \in A} =_a) &= E_A. \end{aligned} \tag{22}$$

With respect to the left-hand-side of an attribute dependency rule given by equation (20), we can define a set of atomic formulas and a set of the meaning sets of atomic formulas:

$$S = \{=_a |\ a \in A\},$$
$$m(S) = \{m(=_a) \mid a \in A\}. \tag{23}$$

In this way, an attribute reduct of A relative to B can be interpreted as a) a reduct of the set of atomic formulas S relative to the formula $=_B$, and b) a reduct of the family of equivalence relations $m(S)$ relative to the equivalence relation E_B. According to Definition 6 and Definition 10, we have two more definitions of a ruduct of an attribute dependency rule.

Definition 23. *For an attribute dependency rule $=_A \rightarrow =_B$, a subset $R \subseteq S$ is reduct of the set of atomic formulas S relative to $=_B$ if R satisfies the following conditions:*

$$\begin{aligned} &\text{(s14)} \quad \cap \{m(p) \mid p \in R\} \subseteq m(=_B), \\ &\text{(n14)} \quad \forall q \in R, (\neg(\cap\{m(p) \mid p \in (R - \{q\})\} \subseteq m(=_B))). \end{aligned} \tag{24}$$

Definition 24. *For an attribute dependency rule $=_A \rightarrow =_B$, a subset $R' \subseteq m(S)$ is reduct of the set of equivalence relations $m(S)$ relative to the equivalence relation $m(=_B)$ if R' satisfies the following conditions:*

$$\begin{aligned} &\text{(s15)} \quad \cap R' \subseteq m(=_B), \\ &\text{(n15)} \quad \forall E \in R, (\neg(\cap(R' - \{E\}) \subseteq m(=_B))). \end{aligned} \tag{25}$$

Recall that different attributes may define the same equivalence relation, like Definition 13, Definition 24 is not a very accurate characterization of a reduct of an attribute dependency rule.

4.3 Step 2: Analysis of Attribute-Value Dependencies

For a classification table with $At = C \cup D$, the result of Step 1 analysis is an attribute reduct $R \subseteq C$. For an equivalence class $[x]_{E_R}$ satisfying the condition $[x]_{E_R} \subseteq [x]_{E_D}$, Pawlak constructs a classification rule showing a dependency between values of x on attributes R and D, respectively. To represent formally such classification rules, we consider a sub-language of the decision logic language used Pawlak [13].

Definition 25. *In an information table, a decision logic language $\mathcal{L}_\mathcal{V}$ is recursively defined as follows: an atomic formula is given by $a = v$, where $a \in At$ and $v \in V_a$. If ϕ and ψ are formulas, then $\phi \wedge \psi$ is a formula.*

An atomic formula $a = v$ is commonly known as an attribute-value pair, written (a, v), or a descriptor. By restricting to the logic connective $\wedge$, we only consider a formula that is the conjunction of a family of atomic formulas. The meaning of a formula is defined by the set of objects satisfying the formula.

Definition 26. *The meanings of formulas of $\mathcal{L}_V$ are recursively computed as follows:*

$$m(a = v) = \{x \in U \mid I_a(x) = v\},$$
$$m(\phi \wedge \psi) = m(\phi) \cap m(\psi). \tag{26}$$

With the introduced logic language $\mathcal{L}_V$, a classification rule can be defined as:

$$\bigwedge_{a \in R} a = I_a(x) \rightarrow \bigwedge_{d \in D} d = I_d(x), \tag{27}$$

or simply,

$$R = I_R(x) \rightarrow D = I_D(x), \tag{28}$$

The left-hand-side of the rule can be understood as a set of attribute value pairs, namely, atomic formulas. A classification rule is therefore called an attribute-value-level rule. Like an attribute-level rule, there may exist superfluous attribute-value pairs on the left-hand-side of the rule. Pawlak calls $m(a = v)$ a category and introduces the notion of a reduct of categories to simplify a classification rule.

By using the same argument for defining a reduct of an attribute dependency rule, we can define a reduct of an attribute-value dependency rule. For an object $x \in U$, we have:

$$\begin{aligned} m(a = I_a(x)) &= [x]_{E_{\{a\}}}, \\ m(\bigwedge_{a \in A} a = I_a(x)) &= [x]_{E_A}. \end{aligned} \tag{29}$$

Based on these results, for rule $R = I_R(x) \rightarrow D = I_D(x)$, we introduce two definitions of an attribute-value-pair reduct relative to $[x]_{E_D}$.

Definition 27. *For a classification rule $R = I_R(x) \rightarrow D = I_D(x)$, a subset of attributes $R(x) \subseteq R$ is called an attribute reduct of x relative to D if $R(x)$ satisfies the two conditions:*

$$\begin{aligned} &\text{(s16)} \quad [x]_{E_{R(x)}} \subseteq [x]_{E_D}; \\ &\text{(n16)} \quad \forall a \in R(x), (\neg([x]_{E_{R(x)-\{a\}}} \subseteq [x]_{E_D})). \end{aligned}$$

Note that (s16) and (n16) are related to (s7) and (n7) of Definition 16. By comparison, an attribute reduct of an information table must be defined with respect to all objects in the table and an attribute reduct of a classification rule is defined with respect to only objects equivalent to x.

Given a classification rule $R = I_R(x) \rightarrow D = I_D(x)$, we can construct a set of attribute-value pairs (i.e., atomic formulas) and the set of their meaning sets, respectively, as follows:

$$\begin{aligned} S(x) &= \{a = I_a(x) \mid a \in R\}, \\ m(S(x)) &= \{m(a = I_a(x)) \mid a \in R\}. \end{aligned} \tag{30}$$

We can use reducts of the two sets to define reducts of of a classification rule in a similar manner as in Definitions 23 and 24.

Definition 28. *For a classification rule $R = I_R(x) \rightarrow D = I_D(x)$, a subset $R(x) \subseteq S(x)$ is called an attribute-value-pair reduct relative to $D = I_D(x)$ if $R(x)$ satisfies the conditions:*

$$\begin{aligned} &\text{(s17)} \quad \cap\{m(\phi) \mid \phi \in R(x)\} \subseteq m(D = I_D(x)); \\ &\text{(n17)} \quad \forall \psi \in R(x), (\neg(\cap\{m(\phi) \mid \phi \in (R(x) - \{\psi\})\} \subseteq m(D = I_D(x)))). \end{aligned} \tag{31}$$

Definition 29. *For a classification rule* $R = I_R(x) \rightarrow D = I_D(x)$*, a subset* $R'(x) \subseteq m(S(x))$ *is called a reduct of* $m(S(x))$ *relative to* $m(D = I_D(x))$ *if* $R'(x)$ *satisfies the conditions:*

$$\begin{aligned} &\text{(s18)} \quad \cap R'(x) \subseteq m(D = I_D(x)); \\ &\text{(n18)} \quad \forall K \in R'(x), (\neg(\cap(R'(x) - \{K\}) \subseteq m(D = I_D(x)))). \end{aligned} \tag{32}$$

Definition 29 is in fact an $\cap$-reduct in Definition 11. Different attribute-value pairs may have the same meaning set, Definition 29 as used by Pawlak is not a very accurate characterization of a relative attribute-value-pair reduct.

In general, similar to the study of attribute-level rules in Section 4.2, we can study attribute-value dependencies for any pair of sets of attributes $A, B \subseteq At$. For example, one can consider attribute-value dependencies by using the set of condition attributes C and the set of decision attributes D in a classification table, instead of using a reduct $R \subseteq C$ from Step 1.

4.4 Step 3: Analysis of Rule Dependencies

After Step 2 analysis, for an attribute reduct $R(x) \subseteq R$ for an object x, we have $[x]_{E_{R(x)}} \subseteq [x]_{E_D}$, which produces a classification rule:

$$R(x) = I_{R(x)}(x) \rightarrow D = I_D(x). \tag{33}$$

The third step of Pawlak data analysis consists of constructing a rule set and simplifying the rule set by removing redundant rules. Pawlak compiles a set of simplified rules by choosing one rule defined by an attributive-value-pair reduct $R(x)$ for each equivalence class $[x]_R$, where R is a relative attribute reduct obtained in Step 1. Let RS denote the rule set obtained in Step 2. There may exist redundant rules in RS. It is therefore necessary to introduce the notion of a rule reduct of RS.

For a classification rule $c \rightarrow d$, we define its meaning as the set of correctly classified objects:

$$m(c \rightarrow d) = m(c \wedge d) = m(c) \cap m(d). \tag{34}$$

Pawlak only considers certain rules derived from the lower approximations. In this case, we have $m(c) \subseteq m(d)$ and $m(c \rightarrow d) = m(c)$. In general, this may not be true. Based on the meaning sets of rules, a rule reduct is related to an $\cup$-reduct of the following family of subsets of U:

$$m(RS) = \{m(c \rightarrow d) \mid (c \rightarrow d) \in RS\}. \tag{35}$$

For an inconsistent table, we have $\cup m(RS) = \mathrm{POS}_{E_C}(U/E_D)$; for a consistent table we have $\cup m(RS) = U$. According to Definition 6, we introduce the notion of a reduct of a rule set.

Definition 30. *A subset of rules $R \subseteq RS$ is called a rule reduct of a set of rules RS if R satisfies the condition:*

$$\begin{aligned} &(\text{s19}) \quad \cup m(R) = \cup m(RS), \\ &(\text{n19}) \quad \forall (c \to d) \in R, (\neg(\cup m(R - \{c \to d\}) = \cup m(RS))), \end{aligned}$$

where $m(R) = \{m(c \to d) \mid (c \to d) \in R\}$.

Condition (s19) states that rules in R are sufficient for correctly classifying all objects as the entire rule set RS and condition (n19) states that each rule in R is necessary.

According to Definition 10, we can directly compute an $\cup$-reduct of the family $m(RS)$ to interpret a reduct of the rule set RS. However, since two rules may have the same meaning set, such an interpretation is not precise.

5 Discussions and Conclusion

In our interpretation and formulations of Pawlak three-step approach, we use a more general and generic notion of reducts. By exploring the monotonicity of evaluations, a reduct of a set is defined as a subset of a set satisfying a pair of conditions, namely, a jointly sufficient condition (s) and an individually necessary condition (n). In total, we consider about twenty definitions of various reducts.

The unified framework based on reducts has a number of advantages. One can apply a generic reduct construction algorithm for constructing any of the three types of reducts. In particular, one may use any of the three classes of algorithms, deletion, addition-deletion, and addition algorithms [22]. All three steps of Pawlak analysis can be viewed as different applications of the same data reduction method. The same framework can be further applied to new situations where a reduct of a set is of interest.

In our formulation, we explicitly express intension and extension of a concept. A classification rule is expressed as a pair of two rules, one for extension and the other for intension: for $x \in U, R \subseteq C$,

$$\begin{cases} [x]_{E_R} \subseteq [x]_{E_D}, \\ \bigwedge_{a \in R} a = I_a(x) \to \bigwedge_{d \in D} d = I_d(x). \end{cases} \tag{36}$$

It enables us to see additional insights into Pawlak three-step approach. Steps 1 and 2 use both intensions and extensions. Step 3 only uses extensions.

The Pawlak three-step approach can be modified in several ways. It can be observed that the first step is not necessary. Thus, a two-step approach can be derived based only on Steps 2 and 3. In Step 2, attribute-value-pair reducts are constructed based on both intensions and extensions. One may consider only extensions of concepts without reference to intensions. This can be formulated as a search for a reduct of the family of subsets of U given by:

$$\{[x]_{E_A} \mid A \subseteq C, [x]_{E_A} \subseteq [x]_{E_D}\}. \tag{37}$$

The results are a new rule learning method [20]. In Step 3, a rule reduct is defined independent of how a rule set is formed.

This paper contributes to Pawlak three-step rough set analysis by introducing a generic notion of reducts, providing multiple interpretations of reducts, and unifying different definitions of reducts in a common framework. We demonstrate that rough set analysis can be formulated based on the central notion of reducts. With some modifications, it is possible to investigate various generalized notions of reducts by using the results from this paper.

Acknowledgements. This work is partially supported by a Discovery Grant from NSERC Canada. The authors thank reviewers for their constructive comments.

References

1. Bazan, J.G., Skowron, A., Synak, P.: Dynamic Reducts as a Tool for Extracting Laws from Decisions Tables. In: Raś, Z.W., Zemankova, M. (eds.) ISMIS 1994. LNCS (LNAI), vol. 869, pp. 346–355. Springer, Heidelberg (1994)
2. Blaszczynski, J., Slowinski, R., Szelag, M.: Sequential covering rule induction algorithm for variable consistency rough set approaches. Information Sciences 181, 987–1002 (2011)
3. Cendrowska, J.: PRISM: An algorithm for inducing modular rules. International Journal of Man-Machine Studies 27, 349–370 (1987)
4. Fürnkranz, J.: Separate-and-conquer rule learning. Artificial Intelligence Review 13, 3–54 (1999)
5. Janusz, A., Ślęzak, D.: Utilization of attribute clustering methods for scalable computation of reducts from high-dimensional data. In: Federated Conference on Computer Science and Information Systems, pp. 307–313 (2012)
6. Grzymala-Busse, J.: LERS - A system for learning from examples based on rough sets. In: Slowinski, R. (ed.) Intelligent Decision Support: Handbook of Applications and Advances of the Rough Sets Theory, pp. 3–18. Kluwer Academic Publishers, Dordrecht (1992)
7. Grzymala-Busse, J., Rzasa, W.: Approximation space and LEM2-like algorithms for computing local coverings. Fundamenta Informaticae 85, 205–217 (2008)
8. Mi, J.S., Leung, Y., Wu, W.Z.: Dependence-space-based attribute reduction in consistent decision tables. Soft Computing 15, 261–268 (2011)
9. Miao, D.Q., Zhao, Y., Yao, Y.Y., Li, H.X., Xu, F.F.: Relative reducts in consistent and inconsistent decision tables of the Pawlak rough set model. Information Sciences 179, 4140–4150 (2009)
10. Moshkov, M.J., Piliszczuk, M., Zielosko, B.: Partial Covers, Reducts and Decision Rules in Rough Sets: Theory and Applications. Springer, Berlin (2008)
11. Nguyen, H.S., Ślęzak, D.: Approximate Reducts and Association Rules. In: Zhong, N., Skowron, A., Ohsuga, S. (eds.) RSFDGrC 1999. LNCS (LNAI), vol. 1711, pp. 137–145. Springer, Heidelberg (1999)
12. Pawlak, Z.: Rough sets. International Journal of Computer and Information Sciences 11, 341–356 (1982)
13. Pawlak, Z.: Rough Sets: Theoretical Aspects of Reasoning About Data. Kluwer Academic Publishers, Dordrecht (1991)

14. Pawlak, Z., Skowron, A.: Rough sets and Boolean reasoning. Information Sciences 177, 41–73 (2007)
15. Quinlan, J.R.: Induction of decision trees. Machine Learning 1, 81–106 (1986)
16. Ślęzak, D.: Association Reducts: A Framework for Mining Multi-attribute Dependencies. In: Hacid, M.-S., Murray, N.V., Raś, Z.W., Tsumoto, S. (eds.) ISMIS 2005. LNCS (LNAI), vol. 3488, pp. 354–363. Springer, Heidelberg (2005)
17. Ślęzak, D., Janusz, A.: Ensembles of Bireducts: Towards Robust Classification and Simple Representation. In: Kim, T.-H., Adeli, H., Slezak, D., Sandnes, F.E., Song, X., Chung, K.-I., Arnett, K.P. (eds.) FGIT 2011. LNCS (LNAI), vol. 7105, pp. 64–77. Springer, Heidelberg (2011)
18. van Mechelen, I., Hampton, J., Michalski, R.S., Theuns, P. (eds.): Categories and Concepts: Theoretical Views and Inductive Data Analysis. Academic Press, New York (1993)
19. Wu, W.Z.: Knowledge reduction in random incomplete decision tables via evidence theory. Fundamenta Informaticae 115, 203–218 (2012)
20. Yao, Y., Deng, X.: A Granular Computing Paradigm for Concept Learning. In: Ramanna, S., Howlett, R.J. (eds.) Emerging Paradigms in ML and Applications. SIST, vol. 13, pp. 307–326. Springer, Heidelberg (2013)
21. Yao, Y., Fu, R.: Partitions, Coverings, Reducts and Rule Learning in Rough Set Theory. In: Yao, J.T., Ramanna, S., Wang, G., Suraj, Z. (eds.) RSKT 2011. LNCS (LNAI), vol. 6954, pp. 101–109. Springer, Heidelberg (2011)
22. Yao, Y., Zhao, Y., Wang, J.: On Reduct Construction Algorithms. In: Wang, G.-Y., Peters, J.F., Skowron, A., Yao, Y. (eds.) RSKT 2006. LNCS (LNAI), vol. 4062, pp. 297–304. Springer, Heidelberg (2006)
23. Yao, Y., Zhou, B., Chen, Y.H.: Interpreting Low and High Order Rules: A Granular Computing Approach. In: Kryszkiewicz, M., Peters, J.F., Rybiński, H., Skowron, A. (eds.) RSEISP 2007. LNCS (LNAI), vol. 4585, pp. 371–380. Springer, Heidelberg (2007)
24. Zhang, W.X., Mi, J.S., Wu, W.Z.: Approaches to knowledge reductions in inconsistent systems. International Journal of Intelligent Systems 18, 989–1000 (2003)
25. Zhao, Y., Yao, Y.Y., Yao, J.T.: Level construction of decision trees for classification. International Journal Software Engineering and Knowledge Engineering 16, 103–126 (2006)
26. Zhu, W., Wang, F.Y.: Reduction and axiomization of covering generalized rough sets. Information Sciences 152, 217–230 (2003)