# Action Reducts

Seunghyun Im[1] , Zbigniew Ras [2,3], Li-Shiang Tsay [4]

[1] Computer Science Department, University of Pittsburgh at Johnstown,
Johnstown, PA 15904, USA
sim@pitt.edu
[2] Computer Science Department, University of North Carolina, Charlotte, NC 28223, USA
[3] Institute of Computer Science, Warsaw University of Technology, 00-665 Warsaw, Poland
ras@uncc.edu
[4] ECIT Department, NC A&T State University, Greensboro, NC, 27411, USA
ltsay@ncat.edu

**Abstract.** An action is defined as controlling or changing some of attribute values in an information system to achieve desired result. An action reduct is a minimal set of attribute values distinguishing a favorable object from other objects. We use action reducts to formulate necessary actions. The action suggested by an action reduct induces changes of decision attribute values by changing the condition attribute values to the distinct patterns in action reducts.

**Keywords:** Reduct, Action Reduct, Prime Implicant, Rough Set.

## 1 Introduction

Suppose that the customers of a bank can be classified into several groups according to their satisfaction levels, such as satisfied, neutral, or unsatisfied. One thing the bank can do to improve the business is finding a way to make the customers more satisfied, so that they continue to do the business with the bank. The algorithm described in this paper tries to solve such problem using existing data. Assume that a bank maintains a database for customer information in a table. The table has a number of columns describing the characteristics of the customers, such as personal information, account data, survey result etc. We divide the customers into two groups based on the satisfaction level (decision value). The first group is comprised of satisfied customers who will most likely keep their account active for an extended period of time. The second group is comprised of neutral or unsatisfied customers. We find a set of distinct values or unique patterns from the first group that does not exist in the second group. The unique characteristics of the satisfied customers can be used by the bank to improve the customer satisfaction for the people in the second group. Clearly, some of attribute values describing the customers can be

controlled or changed, which is defined as an action. In this paper, we propose the concept of action reduct to formulate necessary actions. An action reduct has following properties; (1) It is obtained from objects having favorable decision values. (2) It is a distinct set of values not found in the other group, the group not having favorable decision values. (3) It is the minimal set of differences between separate groups of objects. The minimal set has the advantages when formulating actions because smaller changes are easier to undertake.

The rest of this paper is organized as follows. Chapter 2 describes the algorithm. Related works are presented in Chapter 3. Implementation and experimental results are shown in Chapter 4. Chapter 5 concludes the paper.

**Table 1.** Information System S. The decision attribute is D. The condition attributes are B, C, and E. There are eight objects referred as $x_1 \sim x_8$

| | B | C | E | D |
|---|---|---|---|---|
| $x_1$ | $b_2$ | $c_1$ | $e_1$ | $d_2$ |
| $x_2$ | $b_1$ | $c_3$ | $e_2$ | $d_2$ |
| $x_3$ | $b_1$ | $c_1$ | | $d_2$ |
| $x_4$ | $b_1$ | $c_3$ | $e_1$ | $d_2$ |
| $x_5$ | $b_1$ | $c_1$ | $e_1$ | $d_1$ |
| $x_6$ | $b_1$ | $c_1$ | $e_1$ | $d_1$ |
| $x_7$ | $b_2$ | | $e_2$ | $d_1$ |
| $x_8$ | $b_1$ | $c_2$ | $e_2$ | $d_1$ |

# 2 Algorithm

## 2.1 Notations

We will use the following notations throughout the paper.

By an information system [2] we mean a triple $S=(X,A,V)$, where

$X = \{x_1,x_2,...x_i\}$ *is a finite set of objects,*
$A = \{a_1,a_2,...a_j\}$ *is a finite set of attributes, defined as partial functions from X into V,*
$V = \{v_1,v_2,...,v_k\}$ *is a finite set of attribute values.*

We also assume that $V = \cup\{V_a : a \in A\}$, where $V_a$ is a domain of attribute $a$.

For instance, in the information system $S$ presented by Table 1, $x_1$ refers to the first row and $B(x_1) = b_2$. There are four attributes, $A = \{B, C, E, D\}$. We classify the attributes into two types: condition and decision. The condition attributes are B, C, and E, and the decision attribute is D. We assume that the set of condition attributes is further partitioned into stable attributes, $A_S$, and flexible attributes, $A_F$. An attribute is called stable if the values assigned to objects do not change over time. Otherwise, the attribute is flexible. Birth date is an example of a stable attribute. Interest rate is a flexible attribute.

$A_F = \{B,C\}$
$A_S = \{E\}$
$D$ = *decision attribute*

The values in $D$ are divided into two sets.

$d_\alpha = \{ v_i \in V_D;$ $v_i$ *is a desired decision value*$\}$
$d_\beta = V_D - d_\alpha$

For simplicity of presentation, we use an example having only one element $d_2$ in $d_\alpha$ and $d_1$ in $d_\beta$ ($d_\alpha = \{d_2\}$, $d_\beta = \{d_1\}$). However, the algorithm described in the next section directly carries over to the general case where $|d_\alpha| \geq 2$ and $|d_\beta| \geq 2$.

The objects are partitioned into 2 groups based on the decision values.

$X_\alpha = \{x_i \in X; D(x_i) \in d_\alpha\}$; *i.e., objects that the decision values are in* $d_\alpha$
$X_\beta = \{x_j \in X; D(x_j) \in d_\beta\}$; *i.e., objects that the decision values are in* $d_\beta$

In Table 1, $X_\alpha = \{x_1, x_2, x_3, x_4\}$ and $X_\beta = \{x_5, x_6, x_7, x_8\}$.

### 2.2 Algorithm Description

We want to provide the user a list of attribute values that can be used to make changes on some of the objects to steer the unfavorable decision value to a more favorable value. We use the reduct [2][9] to create that list.

By a reduct relative to an object $x$ we mean a minimal set of attribute values distinguishing $x$ from all other objects in the information system. For example, $\{b_2, e_2\}$ is a reduct relative to $x_7$ since $\{b_2, e_2\}$ can differentiate $x_7$ from other objects in $S$ (see Table 1).

Now, we will extend the concept of "reduct relative to an object" to "α-reduct". We partitioned the objects in $S$ into two groups by their decision values. Objects in $X_\alpha$ have $d_2$ that is the favorable decision value. Our goal is to identify which sets of condition attribute values describing objects in $X_\alpha$ make them different from the objects in $X_\beta$. Although there are several different ways to find distinct condition attribute values (e.g. association rules), reducts have clear advantages; it does not require the user to specify the rule extraction criteria, such as support and confidence values, while generating the minimal set of distinct sets. Thereby, the algorithm is much easier to use, and creates a consistent result across different users.

Table 5 shows α-reducts for S. Those are the smallest sets of condition attribute values that are different from the condition attribute values representing objects in $X_\beta$. We obtained the first two α-reducts relative to $x_1$. These are the prime implicants [1] (see the next section for details) of the differences between $x_1$ and $\{x_5, x_6, x_7, x_8\}$. Subsequent α-reducts are extracted using objects $x_2$, $x_3$, and $x_4$.

We need a method to measure the usability of the α-reducts. Two factors, *frequency* and *hit ratio*, determine the usability. (1) Frequency : More than one object can have the same α-reduct. The *frequency* of an α-reduct in $X_\alpha$ is denoted by $f$. (2) Hit Ratio : The *hit ratio*, represented as $h$, is the ratio between the number of applicable objects and the total number of objects in $X_\beta$. An applicable object is the object that the attribute values are different from those in α-reduct, and they are not stable values. The α-reducts may not be used to make changes for some of the attribute values of $x \in X_\beta$ for two reasons. Some objects in $X_\beta$ do not differ in terms of their attribute values. Therefore, changes cannot be made. Second, we cannot modify stable attribute values. It does not make sense to suggest a change of un-modifiable values. We define the following function to measure the usability. The weight of an α-reduct $k$ is,

$$w_k = ( f_k \cdot h_k ) / ( \Sigma(f \cdot h)),$$

where $f_k$ and $h_k$ are the frequency and hit ratio for $k$, and $\Sigma(f \cdot h)$ is the sum of the weights of all α-reducts. It provides a way to prioritize the α-reduct using a normalized value.

**Table 2.** $X_\alpha$. The objects classified as $d_2$. We assume that $d_2$ is the favorable decision.

| | B | C | E | D |
|---|---|---|---|---|
| $x_1$ | $b_2$ | $c_1$ | $e_1$ | $d_2$ |
| $x_2$ | $b_1$ | $c_3$ | $e_2$ | $d_2$ |
| $x_3$ | $b_1$ | $c_1$ | | $d_2$ |
| $x_4$ | $b_1$ | $c_3$ | $e_1$ | $d_2$ |

**Table 3.** $X_\beta$. The objects classified as $d_1$

| | B | C | E | D |
|---|---|---|---|---|
| $x_5$ | $b_1$ | $c_1$ | $e_1$ | $d_1$ |
| $x_6$ | $b_1$ | $c_1$ | $e_1$ | $d_1$ |
| $x_7$ | $b_2$ | | $e_2$ | $d_1$ |
| $x_8$ | $b_1$ | $c_2$ | $e_2$ | $d_1$ |

### 2.3 Example

*Step 1. Finding α-reducts*

Using the partitions in Tables 2 and 3, we find distinct attribute values of $x \in X_\alpha$ against $x \in X_\beta$. The following matrix shows the discernable attribute values for $\{x_1, x_2, x_3, x_4\}$ against $\{x_5, x_6, x_7, x_8\}$

**Table 4.** Discernable attribute values for $\{x_1, x_2, x_3, x_4\}$ against $\{x_5, x_6, x_7, x_8\}$

| | $x_1$ | $x_2$ | $x_3$ | $x_4$ |
|---|---|---|---|---|
| $x_5$ | $b_2$ | $c_3 + e_2$ | $\varnothing$ | $c_3$ |
| $x_6$ | $b_2$ | $c_3 + e_2$ | $\varnothing$ | $c_3$ |
| $x_7$ | $c_1 + e_1$ | $b_1 + c_3$ | $b_1 + c_1$ | $b_1 + c_3 + e_1$ |
| $x_8$ | $b_2 + c_1 + e_1$ | $c_3$ | $c_1$ | $c_3 + e_1$ |

For example, $b_2$ in $x_1$ is different from $x_5$. We need $b_2$ to discern $x_1$ from $x_5$. Either $c_1$ or (or is denoted as + sign) $e_1$ can be used to distinguish $x_1$ from $x_7$. In order to find the *minimal* set of values that distinguishes $x_1$ from all objects in $X_\beta = \{x_5, x_6, x_7, x_8\}$ we multiply all discernable values: $(b_2) \times (b_2) \times (c_1 + e_1) \times (b_2 + c_1 + e_1)$. That is, $(b_2)$ and $(c_1$

or $e_1$) and ($b_2$ or $c_1$ or $e_1$) should be different to make $x_1$ distinct from all other objects. The process is known as finding a prime implicant by converting the conjunction normal form (CNF) to disjunction normal form (DNF)[2][9]. The α-reduct $r(x_1)$ relative to $x_1$ is computed using the conversion and the absorption laws:

$$\begin{aligned} r(x_1) &= (b_2) \times (b_2) \times (c_1 + e_1) \times (b_2 + c_1 + e_1) \\ &= (b_2) \times (b_2) \times (c_1 + e_1) \\ &= (b_2 \times c_1) + (b_2 \times e_1) \end{aligned}$$

A missing attribute value of an object in $X_\alpha$ does not qualify to discern the object from the objects in $X_\beta$ because it is undefined. A missing value in $X_\beta$, however, is regarded as a different value if a value is present in $x \in X_\alpha$. When a discernible value does not exist, we do not include it in the calculation of the prime implicant. We acquired the following α-reducts:

$r(x_1) = (b_2 \times c_1) + (b_2 \times e_1)$
$r(x_2) = (c_3)$
$r(x_3) = \text{NIL}$
$r(x_4) = (c_3)$

*Step 2. Measuring the usability of* α*-reduct*

Table 5 shows the α-reducts for information System S. The frequency of α-reduct {$b_2$, $c_1$} is 1 because it appears in $X_\alpha$ once. The hit ratio is *4/4* = 1, meaning that we can use the reduct for all objects in $X_\beta$. The weight is 0.25, which is acquired by dividing its weight, $f \cdot h = 1$, by the sum of all weights, $\Sigma(f \cdot h) = (1 \cdot 1) + (1 \cdot 0.5) + (2 \cdot 1) + (1 \cdot 0.5) = 4$.

The values in the stable attribute *E* cannot be modified. Therefore, the hit ratio for α-reduct {$b_2$, $e_1$} is *2/4* = 0.5 because the stable value, $e_2$, in $x_7$ and $x_8$ cannot be converted to $e_1$.

**Table 5.** α-reduct for Information Sytem S. * indicate a stable attribute value.

| α-reduct | Weight (*w*) | Frequency (*f*) | Hit Ratio (*h*) |
|---|---|---|---|
| {$b_2$ , $c_1$} | *2/7 (0.29%)* | *1* | *1* |
| {$b_2$ , $e_1^*$} | *1/7 (0.14%)* | *1* | *0.5* |
| {$c_3$} | *4/7 (0.57%)* | *2* | *1* |

Using the α-reduct {$c_3$} that has the highest weight, we can make a recommendation; change the value of *C* in $X_\beta$ to $c_3$ in order to induce the decision value in $X_\beta$ to $d_2$.

## 3 Implementation and Experiment

We implemented the algorithm in Python 2.6 on a MacBook computer running OS X, and tested it using a sample data set (lenses) obtained from [8]. The data set contains information for fitting contact lenses. Table 6 shows the attributes names, descriptions, and the partitions. The decision attribute, lenses, has three classes. We set the second class (soft lenses) as the favorable decision value. The data set has 4 condition attributes. We assumed that *the age of the patient* is a stable attribute, and *prescription*, *astigmatic* and *tear production rate* are flexible attributes. The favorable decision value and the attribute partition are defined only for this experiment. Their actual definitions might be different. All attributes are categorical in the dataset.

**Table 6.** Dataset used for the experiment.

| Attribute | Values | Type |
|---|---|---|
| (1) age of the patient * | young, pre-presbyopic, presbyopic | Stable |
| (2) spectacle prescription | myope, hypermetrope | Flexible |
| (3) astigmatic | no, yes | Flexible |
| (4) tear production rate | reduced, normal | Flexible |
| (5) lenses | the patient fitted with hard contact lenses<br>the patient fitted with soft contact lenses = $d\alpha$<br>the patient not be fitted with contact lenses. | Decision |

Table 7 shows the α-reducts generated during experiment. We interpret them as action reducts. For instance, the second α-reduct can be read as, change the values in attribute 2, 3, and 4 to the suggested values (hypermetrope, no, normal) in order to change the decision to 'soft lenses'. Because there is no stable attribute value in the α-reduct and the same pattern has not been found in $X_\beta$, its hit ratio is 1.

**Table 7.** α-reduct for $d_\alpha$ = soft lenses. * indicate the attribute value is stable. The number in the () is the attribute number in Table 6.

| α-reduct. | Weight (*w*) | Frequency (*f*) | Hit Ratio (*h*) |
|---|---|---|---|
| young(1)*, no(3), normal(4) | *0.14* | *2* | *0.31* |
| hypermetrope(2), no(3), normal(4) | *0.72* | *3* | *1* |
| pre-presbyopic(1)*, no(3), normal(4) | *0.14* | *2* | *0.31* |

## 4 Related Work and Contribution

The procedure for formulating an action from existing database has been discussed in many literatures. A definition of an action as a form of a rule was given in [4]. The method of action rules discovery from certain pairs of association rules was proposed in [3]. A concept similar to action rules known as interventions was introduced in [5]. The action rules introduced in [3] has been investigated further. In [12], authors present a new agglomerative strategy for constructing action rules from single classification rules. Algorithm ARAS, proposed in [13], is also an agglomerative type strategy generating action rules. The method generates sets of terms (built from values of attributes) around classification rules and constructs action rules directly from them. In [10], authors proposed a method that extracts action rules directly from attribute values from an incomplete information system without using pre-existing conditional rules. In these earlier works, action rules are constructed from classification rules. This means that they use pre-existing classification rules or generate rules using a rule discovery algorithm, such as LERS [6], then, construct action rules either from certain pairs of the rules or from a single classification rule. The methods in [10], [13], [14] and [7] do not formulate actions directly from existing classification rules. However, the extraction of classification rules during the formulation of an action rule is inevitable because actions are built as the effect of possible changes in different rules. Action rules constructed from classification rules provide a complete list of actions to be taken. However, we want to develop a method that provides a simple set of attribute values to be modified without using a typical action rule form (e.g. [condition1$\rightarrow$condition2]=>[decision1$\rightarrow$decision2] ) for decision makers who want to have simple recommendations. The recommendations made by $\alpha$-reduct is typically quite simple, *i.e.* change a couple of values to have a better outcome. In addition, it does not require from the user to define two sets of support and confidence values; one set for classification rules, and other for action rules.

## 5 Summary

This paper discussed an algorithm for finding $\alpha$–reducts (also called action reducts) from an information system, and presented an experimental result. An action reduct is a minimal set of attribute values distinguishing a favorable object from other objects, and are used to formulate necessary actions. The action suggested by an action reduct aims to induce the change of the decision attribute value by changing the condition attribute values to the unique pattern in the action reduct. The algorithm is developed as part of on-going research project seeking solution to the problem of reducing college freshman dropouts. We plan to run the algorithm using real world data in the near future.

**Acknowledgments.** This research has been partially supported by the President's Mentorship Fund at the University of Pittsburgh Johnstown.

## References

1. Gimpel, J. F.: A Reduction Technique for Prime Implicant Tables. In: The Fifth Annual Symposium on Switching Circuit Theory and Logical Design, pp. 183--191. IEEE, Washington (1964)

2. Pawlak, Z.: Rough Sets-Theoretical Aspects of Reasoning about Data, Kluwer, Dordrecht. (1991)

3. Ras, Z.W., Wieczorkowska, A.: Action-Rules: How to Increase Profit of a Company. In: Editor: Zighed, D., Komorowski, J., Zytkow, J. (eds.) Principles of Data Mining and Knowledge Discovery, PADD 2000. LNCS, vol. 1920, pp. 587--592. Springer, Heidelberg (2000)

4. Geffner, H., Wainer, J.: Modeling Action, Knowledge and Control. In: ECAI, pp. 532--536. John Wiley & Sons (1998)

5. Greco, S., Matarazzo, B., Pappalardo, N., Slowinski, R.: Measuring Expected Effects of Interventions based on Decision Rules. Journal of Experimental and Theoretical Artificial Intelligence., Vol. 17, No. 1-2, 103--118 (2005)

6. Grzymala-Busse, J.: A New Version of the Rule Induction System LERS, Fundamenta Informaticae, Vol. 31, No. 1, 27--39 (1997)

7. He, Z., Xu, X., Deng, S., Ma, R.: Mining Action Rules from Scratch, Expert Systems with Applications, Vol. 29, No. 3, 691--699 (2005)

8. Hettich, S., Blake, C.L., Merz, C.J.: UCI Repository of Machine Learning Databases, http://www.ics.uci.edu/~mlearn/MLRepository.html, University of California, Irvine, Dept. of Information and Computer Sciences (1998)

9. Skowron, A.: Rough Sets and Boolean Reasoning. In: Pedrycz, W. (eds.) Granular Computing : An Emerging Paradigm, pp. 95--124. Springer, Heidelberg (2001)

10. Im, S., Ras Z.W., Wasyluk, H.: Action Rule Discovery from Incomplete Data. Knowledge and Information Systems. Vol. 25, No. 1, 21--33 (2010)

11. Qiao, Y., Zhong, K., Wangand, H., Li, X.: Developing Event-Condition-Action Rules in Real-time Active Database. In: ACM Symposium on Applied Computing 2007, pp. 511--516. ACM, New York (2007)

12. Ras, Z.W., Dardzinska, A.: Action Rules Discovery, a New Simplified Strategy. In: F. Esposito et al. (eds.), Foundations of Intelligent Systems, ISMIS 2006. LNAI, Vol. 4203, pp. 445--453, Springer, Heidelberg (2006)

13. Ras, Z.W., Tzacheva, A., Tsay, L., Gurdal, O.: Mining for Interesting Action Rules. In: IEEE/WIC/ACM International Conference on Intelligent Agent Technology, pp. 187--193, IEEE, Washington (2005)

14. Ras, Z.W., Dardzinska, A.: Action Rules Discovery based on Tree Classifiers and Meta-Actions. In: J. Rauch et al. (eds.), Foundations of Intelligent Systems, ISMIS 2009. LNAI, Vol. 5722, pp. 66--75, Springer, Heidelberg (2009)