# Computing Constructs by Using Typical Testor Algorithms

Manuel S. Lazo-Cortés[1(✉)], Jesús Ariel Carrasco-Ochoa[1], José Fco. Martínez-Trinidad[1], and Guillermo Sanchez-Diaz[2]

[1] Instituto Nacional de Astrofísica, Óptica Y Electrónica, Puebla, Mexico
{mlazo,ariel,fmartine}@inaoep.mx
[2] Universidad Autonóma de San Luis Potosí, San Luis Potosí, Mexico
guillermo.Sanchez@uaslp.mx

**Abstract.** In their classic form, reducts as well as typical testors are minimal subsets of attributes that retain the discernibility condition. Constructs are a special type of reducts and represent a kind of generalization of the reduct concept. A construct reliably provides sufficient amount of discrimination between objects belonging to different classes as well as sufficient amount of resemblance between objects belonging to the same class. Based on the relation between constructs, reducts and typical testors this paper focuses on a practical use of this relation. Specifically, we propose a method that allows applying typical testor algorithms for computing constructs. The proposed method involves modifying the classic definition of pairwise object comparison matrix adapting it to the requirements of certain algorithms originally designed to compute typical testors. The usefulness of our method is shown through some examples.

## 1 Introduction

All data analyses in the Rough Sets Theory [14] start with the so-called closed world assumption. According to this assumption any two objects described by two identical vectors of parameter values must be treated equal in all the subsequent analyses. Formally, the main tool that ensures this property in data analysis is the relation of indiscernibility between objects.

The partition of objects into classes is very interesting for the data analysts, and represents a key aspect in classification problems, since the classes generally represent concepts. The Rough Set Theory makes an effort to examine whether a set of descriptive attributes is sufficient to classify objects into the same classes as the original partition. In this effort, reducts play an important role.

Meanwhile, the theory of pattern recognition study the same problems and uses its own tools. Particularly in the logical combinatorial approach [16], the concept of testor [10] makes an important contribution to the problem of feature selection, reduction of the space of representation and other related problems.

Both concepts, reducts and testors have been widely studied separately. However, in the literature the study of these concepts including their points of convergence and their differences have been also studied [9].


J.A. Carrasco-Ochoa et al. (Eds.): MCPR 2015, LNCS 9116, pp. 44–53, 2015.
DOI: 10.1007/978-3-319-19264-2_5

The property to discern objects belonging to different classes is a common point between both concepts while the information provided by the pairs of objects belonging to the same class is sidelined. Just this point is the main contribution of the concept of construct [24]. Constructs take into account more information contained in the object pairwise comparisons, because they utilizes inter-class and intra-class information together, considering discriminating relations between objects belonging to different classes and resembling relations between objects belonging to the same class.

From the computational point of view, the most challenging problem related to testors, reducts and constructs is that of generating full sets of typical testors, reducts and constructs. This problem has been proven to be NP-hard (it is equivalent to the problem of calculate the set of all prime implicants of a disjunctive normal form) [23].

This paper addresses the problem of computing the whole set of constructs. We specifically study the relation between typical testors and constructs, and show how the algorithms for computing typical testors, designed in the logical-combinatorial pattern recognition approach, can be used to compute constructs.

Even though we do not propose a new algorithm, this research expands the set of available algorithms to calculate constructs and re-valorizes the existing typical testor algorithms, bringing new elements to the study of the relationship between the rough set theory and the testor theory.

The rest of the document is organized as follows. Section 2 provides the formal background for the study, including the definitions of reduct, testor and construct. An example is discussed. Section 3 presents the proposed method that allows using typical testor algorithms for computing constructs, illustrative examples are included. Our conclusions are summarized in Sect. 4.

## 2 Theoretical Foundations

### 2.1 Reducts

The main dataset considered in this paper is a decision table, which is a special case of an information table [8]. Formally, a decision table is defined as

**Definition 1.** *(decision table) A decision table is a pair* $\mathcal{S}_d = (U, A_t = A_t^* \cup \{d\})$ *where* $U$ *is a finite non-empty set of objects,* $A_t$ *is a finite non-empty set of attributes.* $A_t^*$ *is a set of conditional attributes and* $d$ *is a decision attribute indicating the decision class for each object in the universe. Each* $a \in A_t$ *corresponds to the function* $I_a : U \rightarrow V_a$ *called evaluation function, where* $V_a$ *is called the value set of* $a$*. The decision attribute allows partitioning the universe into blocks (classes) determined by all possible decisions.*

Sometimes we will use $D$ for denoting $\{d\}$, i.e. $(\{d\} = D)$.

A *decision table* can be implemented as a two-dimensional array (matrix), in which one usually associates rows to objects, columns to attributes and cells to values of attributes on objects.

When considering decision tables, it is important to distinguish between the so called *consistent* and the *inconsistent* ones. A decision table is said to be *consistent*, if each combination of values of descriptive attributes uniquely determines the value of the decision attribute, and *inconsistent*, otherwise. For the purpose of this paper we only consider consistent decision tables.

It is important to introduce the definition of the indiscernibility relation.

**Definition 2.** *(indiscernibility relation) Given a subset of conditional attributes* $A \subseteq A_t^*$*, the indiscernibility relation is defined as* $IND(A|D) = \{(u, v) \in U \times U : \forall a \in A, [I_a(u) = I_a(v)] \vee [I_d(u) = I_d(v)]\}$

The indiscernibility relation is an equivalence relation, so it induces a partition over the universe. Being $\mathcal{S}_d$ a consistent decision table, the partition induced by any subset of conditional attributes is finer than (or at maximum equal to) the relation determined by all possible values of the decision attribute $d$.

We can find several definitions of reduct (see for example, [13]), nevertheless, according to the aim of this paper, we refer to reducts assuming the classical definition of discerning decision reduct [15] as follows.

**Definition 3.** *(reduct for a decision table) Given a decision table* $\mathcal{S}_d$*, an attribute set* $R \subseteq A_t^*$ *is called a reduct, if* $R$ *satisfies the following two conditions:*

*(i)* $IND(R|D) = IND(A_t^*|D)$*;*
*(ii) For any* $a \in R, IND((R - \{a\})|D) \neq IND(A_t^*|D)$*.*

This definition ensures that a reduct has no lower ability to distinguish objects belonging to different classes than the whole set of attributes, being minimal with regard to inclusion, i.e. a reduct does not contain redundant attributes or, equivalently, a reduct does not include other reducts. The original idea of reduct is based on inter-class comparisons.

### 2.2 Testors

The concept of testor (initially *test*) was created by S. V. Yablonskii as a tool for analysis of problems connected with control and diagnosis of faults in circuits [26]. In publications related to this area the original term *test* is used instead of *testor* and the minimal ones (typical testors) are called *dead-end tests*.

The concept of testor (and typical testor) has had numerous generalizations and adaptations to different environments [10]. In this paper, we focus on the classical concept defined into the pattern recognition area, derived from [5], but using a notation similar to that used for reducts to make this presentation more coherent and understandable.

**Definition 4.** *(testor for a decision table) Let* $\mathcal{S}_d = (U, A_t = A_t^* \cup \{d\})$ *a decision table and let* $T \subseteq A_t^*$*.* $I_T(u)$ *denotes the partial description of* $u$ *considering only attributes belonging to* $T$*.* $T \subseteq A_t^*$ *is a testor with respect to a decision table* $\mathcal{S}_d$ *if* $\forall u, v \in \mathcal{U} : [I_T(u) = I_T(v)] \Rightarrow [I_d(u) = I_d(v)]$*. If* $T$ *is a testor such that none of its subsets is a testor, then* $T$ *is a typical (irreducible) testor.*

This definition means that attributes belonging to a testor are jointly sufficient to discern between any pair of objects belonging to different classes; if a testor is typical, each attribute is individually necessary. That is exactly the same as a reduct. This means that in their classical formulations the concepts of reduct and typical testor coincide.

A more detailed study about the relation between reducts and typical testors can be found in [9].

### 2.3 Constructs

Both, reducts and testors are defined from an inter-class object comparison point of view. They ensure sufficient discernibility of objects belonging to different classes. The novelty of the concept of construct (introduced by R. Susmaga in 2003 [24]) is the combination of inter-class and intra-class comparisons in such a way that a resulting subset of conditional attributes would ensure not only the ability to distinguish objects belonging to different classes, but also preserves certain similarity between objects belonging to the same class.

Let us now consider the following similarity relation defined between objects belonging to the same class in a decision table $\mathcal{S}_d = (\mathcal{U}, A_t^* \cup \{d\})$.

**Definition 5.** *(similarity relation) Given a subset of conditional attributes* $A \subseteq A_t^*$*, the similarity relation is defined as* $SIM(A|D) = \{(u, v) \in U \times U : [I_d(u) = I_d(v)]$ *and* $\exists a \in A\ [I_a(u) = I_a(v)]\}$.

If a pair of objects belongs to $SIM(A|D)$ then these objects belong to the same class and they are indiscernible on at least one attribute from the set $A$. This relation is reflexive and symmetric, but it is not transitive.

The definition of construct may be stated as follows.

**Definition 6.** *(cosntruct) Given a decision table* $\mathcal{S}_d$*, an attribute set* $C \subseteq A_t^*$ *is called a construct, if* $C$ *satisfies the following conditions:*

*(i)* $IND(C|D) = IND(A_t^*|D)$;
*(ii)* $SIM(C|D) = SIM(A_t^*|D)$;
*(iii) For any* $a \in C, IND((C-\{a\})|D) \neq IND(A_t^*|D)$ *and* $SIM((C-\{a\})|D) \neq SIM(A_t^*|D)$;

Condition $(i)$ means that a construct retains the discernibility of objects belonging to different classes. In addition, a construct ensures similarity between objects belonging to the same class, at least at level that the whole set of condition attributes does. Condition $(iii)$ ensures the construct's minimality regarding to inclusion relation.

*Example 1.* The matrix $M$ represents a decision table, for which $U = \{u_1, u_2, u_3, u_4, u_5, u_6, u_7, u_8\}$, $A_t^* = \{a_1, a_2, a_3, a_4\}$ and $D = \{d\}$.

$$M = \begin{array}{c} \\ u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \\ u_6 \\ u_7 \\ u_8 \end{array} \begin{array}{c} \begin{array}{ccccc} a_1 & a_2 & a_3 & a_4 & d \end{array} \\ \begin{pmatrix} a & 1 & 25 & red & 1 \\ a & 0 & 19 & yellow & 1 \\ b & 1 & 25 & green & 1 \\ b & 0 & 19 & red & 1 \\ a & 0 & 25 & blue & 2 \\ d & 0 & 29 & pink & 2 \\ c & 0 & 7 & blue & 3 \\ c & 1 & 5 & yellow & 3 \end{pmatrix} \end{array} \tag{1}$$

From (Eq. 1) we have that $\{a_1, a_2\}$ is not a reduct, see for example that $I_{a_1}(u_2) = I_{a_1}(u_5) = a$ and $I_{a_2}(u_2) = I_{a_2}(u_5) = 0$ being $I_d(u_2) = 1$ and $I_d(u_5) = 2$. For this table $\{a_2, a_3\}$, $\{a_1, a_4\}$ and $\{a_3, a_4\}$ are the reducts (also typical testors). Moreover, $\{a_2, a_3\}$, for example, is not a construct, see for example that $1 = I_{a_2}(u_3) \neq I_{a_2}(u_4) = 0$ and $25 = I_{a_3}(u_3) \neq I_{a_3}(u_4) = 19$ being $I_d(u_3) = I_d(u_4) = 1$ and $I_{a_1}(u_3) = I_{a_1}(u_4)$. For $M$ the only construct is $\{a_1, a_2, a_4\}$.

## 3 Proposed Method

In [9] the authors study the relation between reducts and typical testors. Among the practical applications that results as consequence of this relation, they mention that algorithms for computing reducts can be used for computing typical testors, and vice versa.

In this section, we propose a method that allows using testor algorithms for computing constructs, particularly algorithms that use the binary discernibility matrix [6].

Originally, the binary discernibility matrix was defined as *the distinction table* [25] for an information table by comparing all objects regrading all attributes. If we focus on computing reducts, binary discernibility matrices only need comparisons between objects belonging to different classes. In these matrices, it is common to denote by 0 if values corresponding to the same attribute are equal and 1 otherwise.

As it is known, discernibility matrices contain redundant information, and the binary ones are not an exception, so testor algorithms usually work on basic binary discernibility matrices instead of the original discernibility ones (for more details see [12]).

Using this basic binary discernibility matrix, some testor algorithms are easily adapted for computing reducts [2,11,12,19–21]. The novelty of the research reported in this paper is to provide a method that allows algorithms originally designed for computing typical testors to be used to compute constructs too. For that, we need to pre-calculate a new type of binary comparison matrix as follows.

Let $\mathcal{S}_d = (\mathcal{U}, A_t^* \cup \{d\})$ be a decision table, and let us define for each attribute $a$ in $A_t^*$ a dissimilarity function $\varphi_a$ as follows:

$$\varphi_a : V_a \times V_a \rightarrow \{0, 1\}$$

$$\varphi_a(x,y) = \begin{cases} 0 & \text{if } x = y \\ 1 & \text{otherwise} \end{cases} \tag{2}$$

Thus, we can introduce the following definition which constitutes the basis of our proposed method.

**Definition 7.** *(Binary comparison matrix) Given a decision table* $\mathcal{S}_d = (U, A_t = A_t^* \cup \{d\})$*, with* $A_t^* = \{a_1, a_2, \ldots, a_n\}$*. The binary comparison matrix* $M^{\widehat{01}}$ *of* $\mathcal{S}_d$ *is a matrix, in which each row is associated to a pair of objects* $(u, v)$ *with* $u \neq v$ *and is defined by* $(\zeta_{a_1}(I_{a_1}(u), I_{a_1}(v)), \zeta_{a_2}(I_{a_2}(u), I_{a_2}(v)), \ldots, \zeta_{a_n}(I_{a_n}(u), I_{a_n}(v)))$*, being*

$$\zeta_{a_i}(I_{a_i}(u), I_{a_i}(v)) = \begin{cases} \varphi_{a_i}(I_{a_i}(u), I_{a_i}(v)) & if\ I_d(u) = I_d(v) \\ \neg\varphi_{a_i}(I_{a_i}(u), I_{a_i}(v)) & otherwise \end{cases} \tag{3}$$

Definition 7 states that pairs of objects are compared taking into account if both objects belong to different classes or to the same class. In this way, when we are comparing objects, if an attribute distinguishes two objects belonging to different classes, we put a one in the corresponding entry of the matrix, this means that this attribute should be taken into account when we are building constructs.

On the other hand, if an attribute does not distinguish between two objects (belonging to the same class), we put a one, because this attribute contributes to preserve the similarity between these objects, this means that this attribute also should be taken into account when we are building constructs. The complexity of computing the binary comparison matrix is $O(n|U|^2)$

Let us examine the following example.

*Example 2.* As an example, we can build the binary comparison matrix $M^{\widehat{01}}$ for the decision table (Eq. 1) shown in Example 1. For reasons of space we show $M^{\widehat{01}}$ in form of a table, the last two rows indicates the pair of objects that corresponds to the comparison in this column.

| | | | | | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $a_1$ | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 |
| $a_2$ | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 0 |
| $a_3$ | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 0 |
| $a_4$ | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 0 |
| | $u_1$ | $u_1$ | $u_1$ | $u_1$ | $u_1$ | $u_1$ | $u_1$ | $u_2$ | $u_2$ | $u_2$ | $u_2$ | $u_2$ | $u_2$ | $u_3$ | $u_3$ | $u_3$ | $u_3$ | $u_3$ | $u_4$ | $u_4$ | $u_4$ | $u_4$ | $u_5$ | $u_5$ | $u_5$ | $u_6$ | $u_6$ | $u_7$ |
| | $u_2$ | $u_3$ | $u_4$ | $u_5$ | $u_6$ | $u_7$ | $u_8$ | $u_3$ | $u_4$ | $u_5$ | $u_6$ | $u_7$ | $u_8$ | $u_4$ | $u_5$ | $u_6$ | $u_7$ | $u_8$ | $u_5$ | $u_6$ | $u_7$ | $u_8$ | $u_6$ | $u_7$ | $u_8$ | $u_7$ | $u_8$ | $u_8$ |

As usually, the comparison matrix contains redundant information, so we can apply a process of simplification that is equivalent to applying absorption laws for obtaining a matrix that just contains the information needed to compute the whole set of constructs. Obviously, rows containing only zeros are not considered. This kind of matrix is known as basic matrix [12]. For obtaining the basic binary

comparison matrix from the binary comparison matrix it is necessary to compare each row of $M^{\widehat{01}}$ against the remaining rows, therefore, the complexity of this transformation is $O(n|U|^4)$.

*Example 3.* For the matrix $M^{\widehat{01}}$ in the previous example, we have the following basic binary comparison matrix. From this matrix is immediate that the only construct is $\{a_1, a_2, a_4\}$ as we previously said in Example 1.

$$\begin{pmatrix} 1\,0\,0\,0 \\ 0\,0\,0\,1 \\ 0\,1\,0\,0 \end{pmatrix} \tag{4}$$

So, we can enunciate the principal result of this research.

**Proposition 1.** *Let $\mathcal{S}_d = (U, A_t = A_t^* \cup \{d\})$ be a decision table and $M^{\widehat{01}}$ its binary comparison matrix. Then the set $CONST(\mathcal{S}_d)$ of all constructs of $\mathcal{S}_d$ is the same as the set $TT(M^{\widehat{01}})$ of all typical testors of $M^{\widehat{01}}$.*

*Proof. The proposition is a straightforward consequence of the original definitions of typical testor and construct, because both may be interpreted as the prime implicants of the same disjunctive normal form. Notice that the proposition considers $M^{\widehat{01}}$ instead of the basic binary comparison matrix, nevertheless theoretically one can substitutes one matrix for the other one since the associated disjunctive normal forms are the same. Normally, the basic binary comparison matrix is used because it is simpler and usually much more smaller.*

Summarizing, given a decision table, the steps that comprise the proposed method for computing constructs by using typical testor algorithms are as follows:

1. Compute the binary comparison matrix (Definition 7).
2. From matrix computed in step 1, compute the corresponding basic matrix.
3. Apply a typical testor algorithm.

### 3.1 Illustrative Examples

As a manner of illustrative examples, we include in this section the results obtained for three datasets from the UCI Repository of Machine Learning [3].

In Table 1, columns A, B and C contain general information of the datasets (number of conditional attributes, classes and objects respectively); columns D and E show the number of typical testors and constructs computed for each dataset, respectively. In all cases, both typical testors (reducts) and constructs were computed by using the algorithm CT-EXT [19]. Results for reducts were verified by using RSES [4,22].

These results are shown as a small example of how an area can be benefited from other one, in this case the results in the area of testors, specifically algorithms for computing typical testors, can be applied through our proposed method in the area of reducts-constructs.

**Table 1.** Reducts and constructs for several datasets

| Data set | Conditional attributes (A) | Classes (B) | Objects (C) | $\|TT\|$ (D) | $\|CONST\|$ (E) |
|---|---|---|---|---|---|
| Australian | 14 | 2 | 690 | 44 | 2 |
| German(Statlog) | 20 | 2 | 1000 | 846 | 17 |
| Shuttle | 9 | 7 | 43500 | 19 | 1 |

However, this is not an exhausted matter, the exploration of algorithms for computing different types of typical testors [1,7,8,17,18] could be a source for further contributions in this direction.

Similarly, further study of the relation between testors and constructs can bring new benefits, not only in the area of attribute reduction but possibly also for classification.

## 4 Conclusions

The main purpose of the research reported in this paper has been a presentation of a novel method for computing constructs, which involves modifying the classic definition of pairwise object comparison matrix, specifically the binary comparison matrix, adapting it to the requirements of certain algorithms originally designed to compute typical testors.

As we have discussed along the paper, reducts and typical testors are concepts closely related, and in certain environments they coincide. From another point of view constructs constitute a different contribution to the attribute reduction problem.

In this paper, we show that the relation between these different concepts, which has been insufficiently studied to date, can be exploited. Specifically, we illustrate a way in which algorithms that originally were designed for computing typical testors in the framework of pattern recognition, through our proposed method can be used in a straightforward way for computing constructs, by a different implementation of the pairwise comparison matrix.

It is expected that this is not the unique benefit we can obtain of this relation therefore a deeper study is mandatory as future work.

**Acknowledgements.** This work was partly supported by the National Council of Science and Technology of Mexico (CONACyT) through the project grant CB2008-106366.

## References

1. Alba-Cabrera, E., Lopez-Reyes, N., Ruiz-Shulcloper, J.: Generalization of the concept of testor starting from similarity function. Algorithms, Technical report. Yellow serie 134. CINVESTAV-IPN, pp. 7–28, Mexico (1994) (in Spanish)

2. Alba-Cabrera, E., Santana-Hermida, R., Ochoa-Rodriguez, A., Lazo-Cortes, M.: Finding typical testors by using an evolutionary strategy. In: Muge, F., Piedade, M., Caldas-Pinto, R. (eds.) Proceedings of the V Iberoamerican Symposium on Pattern Recognition, SIARP'2000, Universidade Tecnica de Lisboa, pp. 267–278, Lisbon, Portugal (2000)
3. Bache, K., Lichman, M.: UCI Machine Learning Repository. University of California, School of Information and Computer Science, Irvine, CA (2013). http://archive.ics.uci.edu/ml
4. Bazan, J.G., Szczuka, M.S.: The rough set exploration system. In: Peters, J.F., Skowron, A. (eds.) Transactions on Rough Sets III. LNCS, vol. 3400, pp. 37–56. Springer, Heidelberg (2005)
5. Dmitriyev, A.N., Zhuravlev, Y.I., Krendelev, F.P.: On mathematical principles for classification of objects and phenomena. Diskret. Analiz **7**, 315 (1966). (in Russian)
6. Felix, R., Ushio, T.: Rough sets-based machine learning using a binary discernibility matrix. In: Proceedings of the Second International Conference on Intelligent Processing and Manufacturing of Materials, IPMM 1999, pp. 299–305. IEEE (1999)
7. Godoy-Calderon, S., Lazo-Cortes, M.S.: $\Delta$-Testors. A generalization of the concept of testor for fuzzy environments. In: Proceedings of II Iberoamerican Workshop on Pattern Recognition. International Conference ClMAF'97. pp. 95–103, Havana, Cuba (1997) (in Spanish)
8. Goldman, R.S.: Problems on fuzzy testor theory. Automatika i Telemejanika **10**, 146–153 (1980). (in Russian)
9. Lazo-Cortes, M.S., Martinez-Trinidad, J.F., Carrasco-Ochoa, J.A., Sanchez-Diaz, G.: On the relation between rough set reducts and typical testors. Inf. Sci. **294**, 152–163 (2015)
10. Lazo-Cortes, M., Ruiz-Shulcloper, J., Alba-Cabrera, E.: An overview of the evolution of the concept of testor. Pattern Recogn. **34**(4), 753–762 (2001)
11. Lias-Rodríguez, A., Pons-Porrata, A.: BR: a new method for computing all typical testors. In: Bayro-Corrochano, E., Eklundh, J.-O. (eds.) CIARP 2009. LNCS, vol. 5856, pp. 433–440. Springer, Heidelberg (2009)
12. Lias-Rodriguez, A., Sanchez-Diaz, G.: An algorithm for computing typical testors based on elimination of gaps and reduction of columns. Int. J. Pattern Recogn. Artif. Intell. **27**(8) 1350022, 18 (2013)
13. Miao, D.Q., Zhao, Y., Yao, Y.Y., Li, H.X., Xu, F.F.: Reducts in consistent and inconsistent decision tables of the Pawlak rough set model. Inf. Sci. **179**(24), 4140–4150 (2009)
14. Pawlak, Z.: Rough sets. Int. J. Comput. Inf. Sci. **11**, 341–356 (1982)
15. Pawlak, Z.: Rough sets, Theoretical Aspects of Reasoning About Data. Kluwer Academic Publishers, Dordrecht (1992)
16. Ruiz-Shulcloper, J., Abidi, M.A.: Logical combinatorial pattern recognition: a review. In: Pandalai, S.G. (ed.) Recent Research Developments in Pattern Recognition, pp. 133–176. Transwold Research Network, Kerala, India (2002)
17. Ruiz-Shulcloper, J., Alba-Cabrera, E., Lopez-Reyes, N., Lazo-Cortes, M., Barreto-Fiu, E.: Topics about Testor theory CINVESTAV Technical report 134, Yellow series (1995) (in Spanish)
18. Ruiz-Shulcloper, J., Lazo-Cortes, M.: Prime k-testors. Ciencias Técnicas, Físicas y Matemáticas **9**, 17–55 (1991). (in Spanish)
19. Sanchez-Diaz, G., Lazo-Cortes, M., Piza-Davila, I.: A fast implementation for the typical testor property identification based on an accumulative binary tuple. Int. J. Comput. Intell. Syst. **5**(6), 1025–1039 (2012)

20. Diaz-Sanchez, G., Piza-Davila, I., Sanchez-Diaz, G., Mora-Gonzalez, M., Reyes-Cardenas, O., Cardenas-Tristan, A., Aguirre-Salado, C.: Typical testors generation based on an evolutionary algorithm. In: Yin, H., Wang, W., Rayward-Smith, V. (eds.) IDEAL 2011. LNCS, vol. 6936, pp. 58–65. Springer, Heidelberg (2011)
21. Santiesteban-Alganza, Y., Pons-Porrata, A.: LEX : a new algorithm for computing typical testors. Revista Ciencias Matematicas **21**(1), 85–95 (2003). (in Spanish)
22. Skowron, A., Bazan, J., Szczuka, M., Wroblewski, J.: Rough Set Exploration System (version 2.2.1). http://logic.mimuw.edu.pl/ rses/
23. Skowron, A., Rauszer, C.: The discernibility matrices and functions in information systems. In: Słowiński, R. (ed.) Intelligent Decision Support, Handbook of Applications and Advances of the Rough Sets Theory, System Theory, Knowledge Engineering and Problem Solving. Kluwer Academic Publishers, vol. 11, pp. 331–362. Dordrecht, The Netherlands (1992)
24. Susmaga, R.: Reducts versus constructs: an experimental evaluation. Electron. Notes Theor. Comput. Sci. **82**(4), 239–250 (2003). Elsevier
25. Wróblewski, J.: Genetic algorithms in decomposition and classification problem. In: Skowron, A., Polkowski, L. (eds.) Rough Sets in Knowledge Discovery 1, pp. 471–487. Physica Verlag, Heidelberg (1998)
26. Yablonskii, S.V., Cheguis, I.A.: On tests for electric circuits. Uspekhi Mat. Nauk **10**(4), 182–184 (1955). (in Russian)