A Minimal Cover for Declarative Expressions

Margaret Miró-Julià[1] and Josep Miró[2]

Departament de Ciències Matemàtiques i Informàtica
Universitat de les Illes Balears
07071 Palma de Mallorca, SPAIN
margaret.miro@uib.es[1]
dmijmn0@uib.es[2]

Abstract. Descriptive knowledge about a multivalued data table or Object Attribute Table (OAT) can be expressed in declarative form by means of a binary Boolean based language.
This paper presents a contribution to the study of an arbitrary multivalued OAT by introducing an array algebra (not binary) that allows the treatment of multiple valued data tables with systematic algebraic techniques.
An OAT can be described by means of an algebraic array expression. Furthermore, the same OAT can be described by several distinct, but equivalent, array expressions. Among these, the all-prime-ar expression is singled out. The all-prime-ar expression is a unique expression, although it is not necessarily minimum in the number of prime-ars.
Finally, a completely intensional technique that determines a minimal declarative expression (cover) is presented.

Keywords: artificial intelligence, multivalued algebra, symbolic computation, cover.

1 Introduction

Much of the knowledge one has about its environment is descriptive and can be expressed in declarative form by means of a language. Assuming that the objects to be described are elements of the domain D, different levels of declarations can be established. A first level declaration or itemized description describes one element of the domain. A second level declaration or declarative description refers to subsets of the domain not in terms of the elements of the subset but in terms of the attributes and the values these attributes take. Therefore the idea of attributes and attribute values is equivalent to the idea of a subset of objects having these attribute values. Thus, declarative expressions describe aspects of the reality in terms of subsets of objects described by the attribute values that define them.

The transfer of knowledge from the declarative level to the itemized level is very simple. Given a domain $D = \{x_1, x_2, \ldots, x_n\}$ and an arbitrary subset $A = \{x_i, x_j, \ldots, x_k\}$ the problem consists in finding a definition of the type

$$A = \{x \in D \mid P(x)\}\,.$$

In general the problem is not trivial because it may have multiple solutions. The need to establish computer programs has brought the problem back to the surface and several groups have designed approaches to it. Directly or indirectly, work by Michalski [1], Quinlan [2], Pawlak [3], Skowron [4], Miró [5], Wille [6] and Fiol [7] has to do with this problem. However, their efforts are mainly directed to binary descriptions.

The starting point of this research are the itemized descriptions, usually represented by an Object Attribute Table.

Definition 1. *Let $D = \{d_1, d_2, \ldots, d_i, \ldots, d_m\}$ be an ordered set called domain, of elements d_i representing the m objects, let $R = \{r_g, \ldots, r_c, \ldots, r_a\}$ be a set of the g attributes or properties of the objects. The set of values of attribute c is represented by $C = \{[c_{n_c}], \ldots, [c_j], \ldots, [c_1]\}$. The elements of set C, $[c_j]$, are called 1-spec-sets since the elements are defined by means of one specification. An Object Attribute Table (OAT) is a table whose rows represent the objects, and whose columns represent the attributes of these objects. Each element $[c_i]$ represents the value of attribute r_c that corresponds to object d_i.*

	r_g	$\ldots$	r_c	$\ldots$	r_a
d_1	$[g_1]$	$\ldots$	$[c_1]$	$\ldots$	$[a_1]$
d_2	$[g_2]$	$\ldots$	$[c_2]$	$\ldots$	$[a_2]$
$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\ddots$	$\vdots$
d_i	$[g_i]$	$\ldots$	$[c_i]$	$\ldots$	$[a_i]$
$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\ddots$	$\vdots$
d_m	$[g_m]$	$\ldots$	$[c_m]$	$\ldots$	$[a_m]$

Table 1. Object Attribute Table

In order to handle the multivalued OAT a new multivalued algebra is needed.

2 Theoretical Background

2.1 Symbolic Representation of a Subset

The initial objective is to offer a general and compact symbolic representation of an arbitrary subset $C_h \subseteq C$ and of the set operations between subsets.

It is well known that the set of all subsets of a given set C (the power set of C), $\rho(C)$, constitutes a Boolean algebra $< \rho(C), \cup, \cap, \hat{}, \emptyset, C >$. If a symbolic representation of the subsets is considered, there is a parallel Boolean algebra $< \mathcal{S}_c, +, \cdot, \hat{}, \vee_c, \wedge_c >$ defined on the set $\mathcal{S}_c$ of all possible symbols representing subsets of C. The zero of this algebra is $\vee_c$ (the symbol representing the empty set). The identity is $\wedge_c$ (the symbol representing set C).

Throughout this paper, the symbol $\rightsquigarrow$ may be read as: "is described by". Therefore, $C_h \rightsquigarrow c_h$ expresses: "subset C_h is described by symbol c_h". The symbolic representations of regular set operations complement ($\hat{}$), union ($\cup$) and intersection ($\cap$) are:

$$\widehat{C_h} \rightsquigarrow \hat{c_h} \qquad C_h \cup C_k \rightsquigarrow c_h + c_k \qquad C_h \cap C_k \rightsquigarrow c_h \cdot c_k$$

This symbolic representation has been carefully studied in [9], tables providing operations $+$, $\cdot$ and $\hat{}$ on symbols in hexadecimal representation are also given.

2.2 Fundamental Concepts

All the concepts and operations introduced above make reference to only one set, that is, one attribute. A multivalued OAT has more than one attribute.

Let $R = \{r_c, r_b, r_a\}$ be a set of 3 attributes whose attribute values are $C = \{[c_{n_c}], \ldots, [c_2], [c_1]\}$, $B = \{[b_{n_b}], \ldots, [b_2], [b_1]\}$ and $A = \{[a_{n_a}], \ldots, [a_2], [a_1]\}$. The elements of sets C, B, A are 1-spec-sets (one specification). A 3-spec-set, $[c_k, b_j, a_i]$, is a chain ordered description of 3 specifications, one from set C, one from set B and one from set A. Each spec-set represents itself and all possible permutations. Therefore,

$$[c_k, b_j, a_i] = [c_k, a_i, b_j] = [b_j, c_k, a_i] = [b_j, a_i, c_k] = [a_i, c_k, b_j] = [a_i, b_j, c_k]$$

This idea can be generalized for g attributes. In all definitions that follow, $R = \{r_g, \ldots, r_b, r_a\}$ is the set of g attributes whose attribute values are given by non-empty sets $G, \ldots, B, A$ respectively.

Definition 2. *The cross product $G \otimes \cdots \otimes B \otimes A$ is the set of all possible g-spec-sets formed by one element of G, ..., one element of B and one element of A.*

$$G \otimes \cdots \otimes B \otimes A = \{[g_x, \ldots, b_j, a_i] \mid [g_x] \in G, \ldots, [b_j] \in B, [a_i] \in A\}$$

It is important to mention that the cross product is not the cartesian product. A g-spec-set represents itself and all possible permutations whereas the elements of the cartesian product are different if the order in which there are written varies. There is a need to determine an order in a g-spec-set. The basis T is an ordered chain $< G, \ldots, B, A > \equiv T$ which establishes the sequential order in which the spec-sets are always written. The basis considered in this paper is $T =< G, \ldots, B, A >$.

The set of all possible g-spec-sets induced by sets $G, \ldots, B, A$ is called the **universe** and every subset of the universe is called a **subuniverse**.

Definition 3. *Let $G_i \subseteq G$, ..., $B_i \subseteq B$, $A_i \subseteq A$, an array $|t_i| = |g_i, \ldots, b_i, a_i|$ is the symbolic representation of the cross product $G_i \otimes \ldots \otimes B_i \otimes A_i$ where $G_i \rightsquigarrow g_i$, ..., $B_i \rightsquigarrow b_i$, and $A_i \rightsquigarrow a_i$.*

$$G_i \otimes \cdots \otimes B_i \otimes A_i = \{[g_x, \ldots, b_j, a_i] \mid [g_x] \in G_i, \ldots, [b_j] \in B_i, [a_i] \in A_i\}$$

$$G_i \otimes \cdots \otimes B_i \otimes A_i \rightsquigarrow |t_i| = |g_i, \ldots, b_i, a_i|$$

An array $|t_i|$ is a symbolic representation of a subuniverse. In 2 dimensions (2 attributes) an array can be represented graphically as shown in Fig. 1.

The arrays describe subuniverses (subsets), therefore regular set operations may be performed with them. The following operations between arrays are introduced. Let $|t_i| = |g_i, \ldots, b_i, a_i|$ and $|t_j| = |g_j, \ldots, b_j, a_j|$ be two arrays.

$$G_i \otimes \cdots \otimes B_i \otimes A_i \rightsquigarrow |t_i| \qquad G_j \otimes \cdots \otimes B_j \otimes A_j \rightsquigarrow |t_j|$$

1.- $\sim$ complement of an array respect to the universe

$$\sim (G_i \otimes \cdots \otimes B_i \otimes A_i) \rightsquigarrow \quad \sim |t_i|$$

where $\sim$ is the symbolic representation of the complement respect to the universe (set of all g-spec-sets). In 2 dimensions, the $\sim$ complement of an array can be represented as shown in Fig. 2.

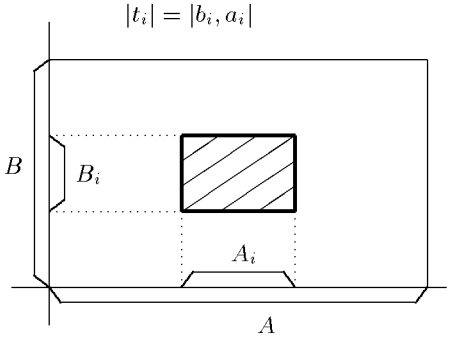

Fig. 1. Arrays in 2-dimensions

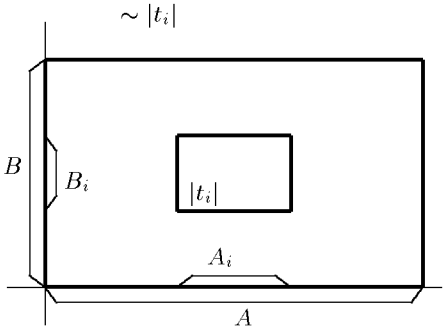

Fig. 2. $\sim$ complement of an array

2.- $\ddagger$ sum of arrays

$$(G_i \otimes \cdots \otimes B_i \otimes A_i) \cup (G_j \otimes \cdots \otimes B_j \otimes A_j) \rightsquigarrow |t_i| \ddagger |t_j|$$

$$|t_i| \ddagger |t_j| = |g_i, \ldots, b_i, a_i| \ddagger |g_j, \ldots, b_j, a_j|$$

where the $\ddagger$ sum is the symbolic representation of the union of subuniverses. If only 2 attributes are considered, the $\ddagger$ sum of arrays can be represented graphically as shown in Fig. 3.

3.- $\circ$ product of arrays

$$(G_i \otimes \cdots \otimes B_i \otimes A_i) \cap (G_j \otimes \cdots \otimes B_j \otimes A_j) \rightsquigarrow |t_i| \circ |t_j|$$

$$|t_i| \circ |t_j| = |g_i, \ldots, b_i, a_i| \circ |g_j, \ldots, b_j, a_j| = |g_i \cdot g_j, \ldots, b_i \cdot b_j, a_i \cdot a_j|$$

where the $\circ$ product is the symbolic representation of the intersection of subuniverses. Furthermore, the $\circ$ product is a closed operation in the set of all arrays. If only 2 attributes are considered, the $\circ$ product can be graphically represented as can be seen in Fig. 4.

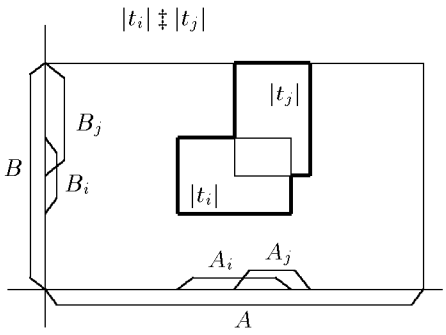

Fig. 3. $\ddagger$ sum of arrays

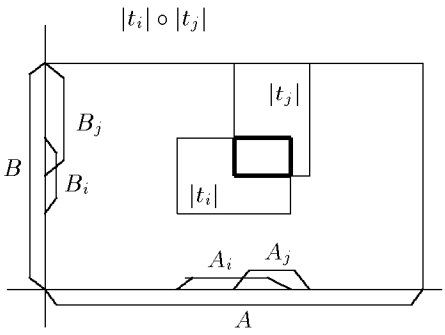

Fig. 4. $\circ$ product of arrays

All the results obtained by use of operations $\sim$, $\ddagger$ and $\circ$ on arrays are symbolic representations of subuniverses. There are two subuniverses that deserve special consideration. First, the identity array $\bigwedge$, which is the array representing the universe:

$$U \rightsquigarrow \bigwedge = |\wedge_g, \ldots, \wedge_b, \wedge_a|$$

Second, the zero array $\bigvee$, the array representing the empty universe:

$$\emptyset \rightsquigarrow \bigvee = |\vee_g, \ldots, \vee_b, \vee_a|$$

3 Array Expressions

Subuniverses can be symbolically represented by arrays or by algebraic expressions of arrays. An expression is a symbolic representation of a subuniverse. An expression represents the reality described by an OAT.

Definition 4. *Any combination of arrays using operations* $\sim$, $\ddagger$ *and* $\circ$ *(well formed formula) is called an expression* E_i.

$$E_i = \sim |t_i| \ddagger |t_j| \circ |t_k| \ldots$$

Generally, a subuniverse can be represented by more than one expression. Expressions that describe the same subuniverse are said to be equivalent (declaratively). The comparison of two distinct expressions, as far as their declarative describing capability, has been studied in [9] and [10].

Expressions represent subuniverses, therefore an order relation that symbolically represents set inclusion may be introduced:

$$(U_i \subseteq U_j) \rightsquigarrow E_i \preceq E_j \,.$$

This order relation has been studied in [9] and has been used to find simplified equivalent expressions.

Definition 5. *An expression* E_i *is called an array expression if it is written as a* $\ddagger$ *sum of arrays.*

$$E_i = |t_z| \ddagger \cdots \ddagger |t_y| \ddagger \cdots \ddagger |t_x|$$

An array expression in 2 dimensions is shown in Fig. 5.

Definition 6. *Given an array expression* $E_i = |t_z| \ddagger \cdots \ddagger |t_y| \ddagger \cdots \ddagger |t_x|$, $|t_y|$ *is a prime array or prime-ar of expression* E_i *if there is no other array* $|t_j|$ *such that:*

$$|t_y| \preceq |t_j| \preceq E_i$$

A prime-ar is a "largest" array contained in E_i.

Consider the array expression given in Fig. 5. Both $|t_1|$ and $|t_2|$ are prime-ars, however $|t_3|$ is not a prime-ar since $|t_3| \preceq |t_j| \preceq E_i$, see Fig. 6. The prime-ars of expression E_i are $|t_1|$, $|t_2|$ and $|t_j|$.

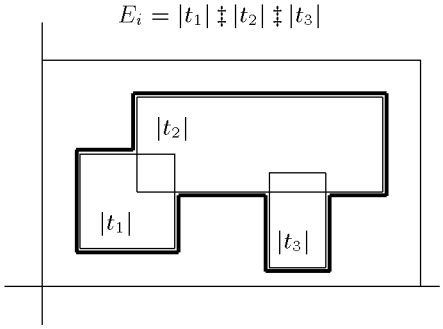

Fig. 5. Array expression

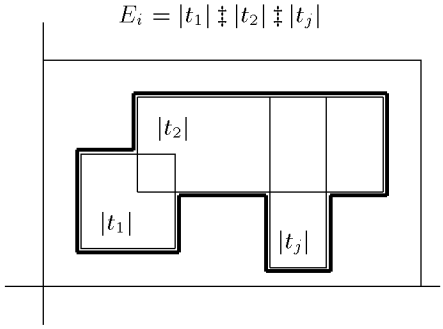

Fig. 6. Prime-ars of expression E_i

4 The All-prime-ar Expression

Definition 7. *The* $\ddagger$ *sum of all the prime-ars of an expression* E_i *is called the all-prime-ar expression of* E_i.

The same subuniverse can be described by more than one prime-ar expression. The all-prime-ar expression is a unique expression, but the number of prime-ars may not be minimal.

Fig. 7 offers a two dimensional insight to these remarks. The unique all-prime-ar expression is $E_i = |t_1| \ddagger |t_2| \ddagger |t_3| \ddagger |t_4|$. Since $|t_1| \preceq |t_2| \ddagger |t_3|$ and $|t_3| \preceq |t_1| \ddagger |t_4|$, two equivalent prime-ar expressions can be found:

$$E_i = |t_2| \ddagger |t_3| \ddagger |t_4| \qquad E_i = |t_1| \ddagger |t_2| \ddagger |t_4|$$

The same subuniverse can be described by more than one prime-ar expression. How can the equivalency between expressions be studied? A possible way to know if two prime-ar expressions are equivalent is to compare their all-prime-ar expression. An algorithm that provides all the prime-ars of an expression is given in [9] together with different theorems that simplify the execution of the algorithm.

5 Covers of an Expression

The all-prime-ar expression is unique and achievable by means of an algorithm. However, the number of arrays appearing in the expression is not necessarily minimal. In set theory, where sets are extensionally given, a cover is easily obtained. But, can a minimal cover be obtained using a declarative treatment? Algorithmic techniques that provide a minimal declarative expression are presented.

Definition 8. *Given a set of arrays* $A = \{|t_1|, |t_2|, \ldots, |t_n|\}$ *an array* $|t_i|$ *is said to be covered by* A *or is called a covered array respect to* A *if and only if*

$$|t_i| \preceq |t_1| \ddagger |t_2| \ddagger \cdots \ddagger |t_n|$$

An array that is not a covered array respect to A *is called an uncovered array respect to* A.

Definition 9. *Given an array expression* $E = |t_1| \ddagger \cdots \ddagger |t_i| \ddagger \cdots \ddagger |t_k| \ddagger \cdots \ddagger |t_z|$ *an array* $|t_i|$ *of the expression is called a redundant array respect to* E *if and only if*

$$|t_i| \preceq |t_1| \ddagger \cdots \ddagger |t_{i-1}| \ddagger |t_{i+1}| \ddagger \cdots \ddagger |t_z|$$

In other words, $|t_i|$ *is covered by* $A - \{|t_i|\} = \{|t_1|, \ldots, |t_{i-1}|, |t_{i+1}|, \ldots, |t_z|\}$.

An array that is not a redundant array respect to E *is said to be an essential array respect to* E.

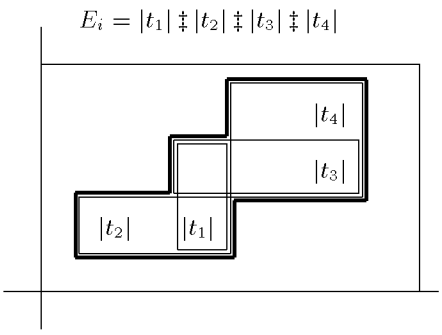

Fig. 7. The all-prime-ar expression

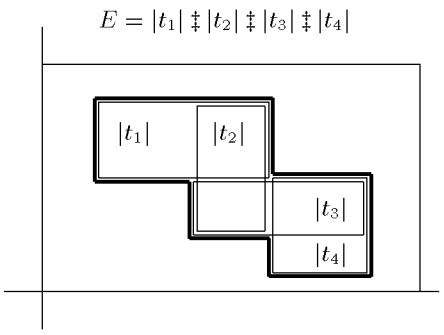

Fig. 8. Redundant and essential arrays

Examining Fig. 8, it can be easily seen that $|t_1|$ and $|t_4|$ are essential arrays respect to expression E, whereas $|t_2|$ and $|t_3|$ are redundant.

$$|t_2| \preceq |t_1| \ddagger |t_3| \ddagger |t_4| \qquad |t_3| \preceq |t_1| \ddagger |t_2| \ddagger |t_4|$$

Definition 10. *Given an expression* E, *whose arrays form set* A, *a subset* $A_c \subseteq A$ *is a cover of* E *if and only if*

$$E \preceq \Sigma_{|t_i| \in A_c} |t_i|$$

A cover is formed by all the essential arrays of the expression and some of the redundant arrays. The two covers C_A and C_B of expression E represented on Fig. 8 are given in Fig. 9.

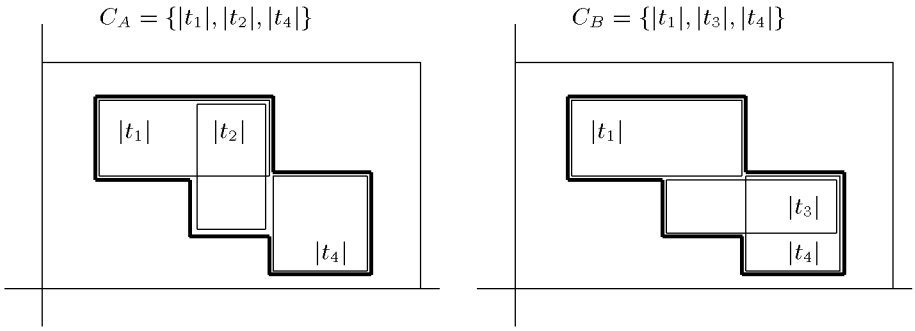

Fig. 9. Covers of an expression

In order to find a cover of an array expression, the essential arrays must be determined.

5.1 Determination of essential arrays

In order to determine the essential arrays, the redundancy of the array will be checked.

Definition 11. *Given an array expression* $E = |t_1| \ddagger \cdots \ddagger |t_i| \ddagger \cdots \ddagger |t_j| \ddagger \cdots \ddagger |t_z|$ *the remainder of* $|t_i|$ *respect to* $|t_1|$, $R_{i,1}$, *is defined as:*

$$R_{i,1} = |t_i| \circ (\sim |t_1|)$$

the remainder of $R_{i,1}$ *respect to* $|t_2|$ *is defined as*

$$R_{i1,2} = R_{i,1} \circ (\sim |t_2|) = |t_i| \circ (\sim |t_1|) \circ (\sim |t_2|)$$

The remainder of $|t_i|$ *respect to* $|t_1|$, $|t_2|$, ..., $|t_n|$ *is defined as*

$$R_{i12\ldots,n} = |t_i| \circ (\sim |t_1|) \circ \cdots \circ (\sim |t_{i-1}|) \circ (\sim |t_{i+1}|) \circ \cdots \circ (\sim |t_n|)$$

Given an array expression $E = |t_1| \ddagger \cdots \ddagger |t_i| \ddagger \cdots \ddagger |t_j| \ddagger \cdots \ddagger |t_z|$ an array $|t_i|$ of the expression is redundant if and only if the remainder $R_{i12\ldots,n} = \bigvee$, an array $|t_i|$ of the expression is essential if and only if the remainder $R_{i12\ldots,n} \neq \bigvee$.

The following algorithm finds the essential arrays of an array expression. The procedure consists in determining if an array $|t_i|$ is covered by all other arrays of the expression. The algorithm uses ARRAYLIST formed by all the arrays of the expression and generates REDLIST and ESSLIST. Initially, all lists are empty.

1. Create ARRAYLIST with all arrays from the expression (delete repeated arrays, if any).
2. For all $|t_i|$ belonging to ARRAYLIST,
   a) remove $|t_i|$ from ARRAYLIST.
   b) obtain the remainder $R_{i12\ldots,n}$ of $|t_i|$ respect to ARRAYLIST.
      - if $R_{i12\ldots,n} = \bigvee$, include $|t_i|$ in REDLIST;
      - if $R_{i12\ldots,n} \neq \bigvee$, include $|t_i|$ in ESSLIST.

The ESSLIST is formed by all the essential arrays of the expression, the redundant arrays appear in REDLIST.

5.2 Determination of a cover

All the essential arrays are included in a cover but only some of the redundant arrays are necessary. The following algorithm determines whether a redundant array can be dispensed with in a cover. The algorithm uses the REDLIST and ESSLIST generated by the previous algorithm, and generates a NEWREDLIST and a NEWESSLIST. The procedure consists in determining whether an array in REDLIST is covered by the arrays in ESSLIST.

1. Remove those arrays in REDLIST covered by other arrays of REDLIST. For all $|t_i|$ and $|t_j|$ belonging to REDLIST, if $R_{i,j} = |t_i| \circ (\sim |t_j|) = \bigvee$ remove $|t_i|$ from REDLIST.
2. For all $|t_i|$ belonging to REDLIST,
   a) remove $|t_i|$ from REDLIST.
   b) obtain the remainder $R_{i12\ldots,k}$ of $|t_i|$ respect to ESSLIST.
      - if $R_{i12\ldots,k} = \bigvee$, include $|t_i|$ in NEWREDLIST;
      - if $R_{i12\ldots,k} \neq \bigvee$, include $|t_i|$ in NEWESSLIST.
3. Create COVERLIST with all arrays from ESSLIST and NEWESSLIST.

A cover is formed by all essential arrays respect to ARRAYLIST and those redundant arrays respect to ARRAYLIST that are essential respect to ESSLIST.

The results obtained suggest the following comments:

- The cover of an array expression (or set of arrays) has been found without making use of the extensional description of sets.
- The cover is not unique. The ordering of the arrays in the sets and the order in which calculations are performed may change the final outcome.
- A shorter cover may be possible if one (or more) of the remainders $R_{i12\ldots,k}$ is covered by the rest of them.
- If the initial array expression is the all-prime-ar expression then the cover is minimum (no smaller cover can be found).

6 Conclusion

An algebra of arrays that allows the description of an arbitrary OAT by means of an array expression has been presented. The proposed array algebra does not handle raw data, it handles declarative descriptions of the data. Declarative expressions from a multivalued OAT can be obtained using arrays and declarative expressions can be transformed by application of computational techniques.

These array expressions are not unique. In order to find a unique array expression the concept of prime-ar is introduced. The all-prime-ar expression is an unique expression, however the number of prime-ars in the expression is not minimal.

From the many equivalent expressions describing an OAT some may be more economic or elegant. A completely intensional technique has been provided to converge on them. The algorithms presented here find a cover of an expression, by determining the essential arrays and which of the redundant arrays need to be included in the cover.

The procedure presented has a feature that should be mentioned. The computation may be interrupted at any time. The results obtained up to the interruption will always allow to a find a cover. In general the latter the computation has been interrupted, the closer the solution will be to the optimum one.

References

1. Michalski, R.S.: A Theory and Methodology of Inductive Learning. Artificial Intelligence **20** (1983) 111–161
2. Quinlan, J. R.: Induction of Decision Trees. Machine Learning **1** (1986) 81–106
3. Pawlak, Z.: Rough Sets: Theoretical Aspects of Reasoning About Data. Kluwer Academic Publisher (1991)
4. Bazan, J. and Skowron, A. and Synak, P.: Discovery of Decision Rules from Experimental Data. Proceedings of the Third International Workshop on Rough Sets and Soft Computing (1994) 346–355
5. Miró, J. and Miró-Julià, J.: Uncertainty and Inference through Approximate Sets. Uncertainty in Intelligent Systems. North Holland (1993) 203–214
6. Wille, R.: Restructuring Lattice Theory: an Approach based on Hierarchies of Concepts. Ordered Sets, Reidel Publishing Company (1982) 445–470
7. Fiol, Gabriel and Miró Nicolau, José and Miró-Julià, José: A New Perspective in the Inductive Acquisition of Knowledge from Examples. Lecture Notes in Computer Science **682** (1992) 219–228
8. Miró, J. and Miró-Julià, M.: A Numerical Computation for Declarative Expressions. Lecture Notes in Computer Science **1333** (1997) 236–251
9. Miró-Julià, M.: A Contribution to Multivalued Systems. Ph.D. thesis. Universitat de les Illes Balears (2000)
10. Miró, J. and Miró-Julià, M.: Equality of Functions in CAST. Lecture Notes in Computer Science **1030** (1995) 129–136