Fundamenta Informaticae 33 (1998) 1–22 1
IOS Press

An Application of Rough Set Methods in Control Design

J.F. Peters
Department of Electrical and Computer Engineering
University of Manitoba
Winnipeg, R3T 2N2, Canada
e-mail: peters@manitoba.ca

A. Skowron
Institute of Mathematics
Warsaw University
Banacha 2, 02-097 Warsaw, Poland
e-mail: skowron@mimuw.edu.pl

Z. Suraj
Institute of Mathematics
Pedagogical University of Rzeszów
Rejtana 16A, 35-310 Rzeszów, Poland
e-mail: zsuraj@univ.rzeszow.pl

To the memory of Professor Adam Mrózek

Abstract. The paper deals with an automatic concurrent control design method derived from the specification of a discrete event control system represented in the form of a decision table. The main stages of our approach are: the control specification by decision tables, generation of rules from the specification of the system behavior, and converting rules set into a concurrent program represented in the form of a Petri net. Our approach is based on rough set theory [17].

1. Introduction

The design of controllers for complex systems or devices is a very important research task from theoretical as well as practical point of view. A substantial effort has been made by rough set

community to develop rough control methods (see, e.g., [3],[4],[7],[8],[9],[10],[16],[18],[20],[35],[36]). The aim of the paper is to demonstrate a methodology for the automatic control design derived from the specification of a discrete event system given in the form of a decision table [17]. An algorithm for computing a highly parallel program represented by a Petri net from a given decision table has been proposed in [27]. In this paper, we show an application of that algorithm after some modifications for a control design. We use ordinary Petri nets with priorities [21] as a model of the target system.

A specification method presented in the paper can be more convenient for the control designers of systems than drawing directly nets especially when they are large. The control designer of systems applying our method concentrates on a specification of local process dependencies in global states. These dependencies are represented by a decision table [17]. Of course, the design process of the solution is iterative. In a successive step, the previously constructed net is automatically redesigned when some new dependencies are discovered and added to the existing specification. The nets produced automatically by application of our method can be simplified by using some reduction procedures. This problem will be discussed in our next paper. We expect that our method can be also applied as a convenient tool for the synthesis of larger systems [24].

We illustrate our ideas by an example of a controller design for a very simple dosing tank presented in [4].

In this paper, we investigate a problem which informally can now be stated as follows:

Control Design Problem (CDP):
`Input:` A specification of a discrete event control system by a decision table S.
`Output:` A description of a control for the system represented by a decision table S.

The text of the paper is organized as follows. Section 2 deals with basic definitions and facts concerning rough sets analysis. In Section 3, we present a standard scheme of controlled system. Section 4 describes informally the control model. An illustrative example used in the rest of the paper is presented in Section 5. Results of rough set analysis of illustrative example are described in Section 6. Section 7 contains basic definitions and notation from Petri net theory. The transformation of rules into a Petri net representing the control for a designed system is presented in Section 8. The last section includes some remarks and comments on methodology outlined in the paper.

2. Preliminaries of Rough Set Theory

Rough sets have been introduced [17] as a tool to deal with inexact, uncertain or vague knowledge in artificial intelligence applications. This section contains basic notions of rough set theory included to help the reader to understand our results.

2.1. Information Systems and Decision Tables

Information systems (sometimes called data tables, attribute-value systems, condition-action tables, knowledge representation systems etc.) are used for representing knowledge. The notion of an information system presented here is due to Z. Pawlak and was investigated by several authors (see, e.g., the selected bibliography on rough sets in [23], pp. 491-552).

An *information system* is a pair $S = (U, A)$, where U - is a non-empty, finite set called the *universe,* A - is a non-empty, finite set of *attributes*, i.e., $a : U \to V_a$ for $a \in A$, where V_a is called the *value set* of a. Elements of U are called *objects*. In the paper attributes are meant to denote the processes of the system, the values of attributes are understood as local states of processes and objects are interpreted as global states of the system.

The set $V = \bigcup_{a \in A} V_a$ is said to be the *domain* of A.

For $S = (U, A)$, a system $S' = (U', A')$ such that $U \subseteq U'$, $A' = \{a' : a \in A\}$, $a'(u) = a(u)$ for $u \in U$ and $V_a = V_{a'}$ for $a \in A$ will be called an $U' - extension$ of S (or an extension of S, in short). S is then called a *restriction* of S'. If $S = (U, A)$ then $S' = (U, B)$ such that $A \subseteq B$ will be referred to as a $B - extension$ of S. S is also called a *subsystem of S'*.

Let $S = (U, A)$ be an information system. With any subset of attributes $B \subseteq A$ we associate a binary relation $ind(B)$, called an *indiscernibility relation* [17], which is defined by $ind(B) = \{(u, u') \in U \times U$ for every $a \in B$, $a(u) = a(u')\}$. Notice that $ind(B)$ is an equivalence relation.

If u $ind(B)$ u', then we say that the objects u and u' are indiscernible with respect to attributes from B. In other words, we cannot distinguish u from u' in terms of attributes in B.

Any information system $S = (U, A)$ determines an *information function* $Inf_A : U \to P(A \times V)$ defined by $Inf_A(u) = \{(a, a(u)) : a \in A\}$, where $V = \bigcup_{a \in A} V_a$ and $P(X)$ denotes the powerset of X. The set $\{Inf_A(u) : u \in U\}$ is denoted by $\mathrm{INF}(S)$.

Hence, $u\ ind(A)\ u'$ if and only if $Inf_A(u) = Inf_A(u')$.

The values of an information function will be sometimes represented by vectors of the form $(v_1, ..., v_m)$, $v_i \in V_a$, for $i = 1, ..., m$, where $A = \{a_1, ..., a_m\}$. Such vectors are called *information vectors* (over V and A).

In the paper, we also consider a special case of information systems called *decision tables* [17].

A *decision table* is any information system of the form $S = (U, A \cup \{d\})$, where $d \notin A$ is a distinguished attribute called *decision.* The elements of A are called *conditional attributes (conditions).*

Any decision table $S = (U, A \cup \{d\})$ can be represented by a data table with the number of rows equal to the cardinality of the universe U and the number of columns equal to the cardinality of the set $A \cup \{d\}$. On the position corresponding to the row u and column a the value $a(u)$ appears.

Let $S = (U, A)$ be an information system, where $A = \{a_1, ..., a_m\}$. Pairs (a, v) with $a \in A, v \in V$ are called *descriptors*. Instead of (a, v) we also write $a = v$ or a_v.

The set of *terms* over A and V is the least set containing descriptors (over A and V) and closed with respect to the classical propositional connectives: $\neg$ (negation), $\vee$ (disjunction), and

$\wedge$ (conjunction), i.e., if τ, τ' are terms over A and V then $\neg\tau, (\tau \vee \tau'), (\tau \wedge \tau')$ are terms over A and V.

The meaning $\| \tau \|_S$ (or in short $\| \tau \|$) of a term τ in S is defined inductively as follows: $\| (a, v) \| = \{u \in U : a(u) = v\}$ for $a \in A$ and $v \in V_a$; $\| \tau \vee \tau' \| = \| \tau \| \cup \| \tau' \|$; $\| \tau \wedge \tau' \| = \| \tau \| \cap \| \tau' \|$; $\| \neg\tau \| = U - \| \tau \|$.

Two terms τ and τ' are equivalent, $\tau \equiv \tau'$, if and only if $\| \tau \| = \| \tau' \|$. In particular, we have: $\neg(a = v) \equiv \bigvee\{a = v' : v' \neq v \text{ and } v' \in V_a\}$.

2.2. Dependencies in Decision Tables

In this subsection, we recall some notions related to dependencies in decision tables and we introduce a new notion connected with an equivalence of attribute sets in decision tables. The last notion plays an important role in the rule reduction generated from a given decision table.

Let $S = (U, A)$ be an information system and let $B, C \subseteq A$. We say that the *set C depends* on B in S in *degree* k $(0 \leq k \leq 1)$, symbolically $B \underset{\mathrm{S,k}}{\longrightarrow} C$, if $k = \frac{card(POS_B(C))}{card(U)}$, where $POS_B(C)$ is the *B-positive region of C* in S [17].

If $k = 1$, we write $B \underset{\mathrm{S}}{\longrightarrow} C$ instead of $B \underset{\mathrm{S,k}}{\longrightarrow} C$ and we say that C is *totally dependent on B* in S. In this case $B \underset{\mathrm{S}}{\longrightarrow} C$ means that $ind(B) \subseteq ind(C)$. If the right hand side of a dependency consists of one attribute only, we say the dependency is *elementary*.

If the set C is totally dependent on B and vice versa, then we say that C and B are *equivalent* in S. In this case $B \underset{\mathrm{S}}{\longleftrightarrow} C$ means that $ind(B) = ind(C)$.

A pair (B, C) of subsets of A is called *a strong component* of S if and only if the following conditions are satisfied: (i) $B \cap C = \emptyset$, (ii) B and C are equivalent in S, (iii) B and C are minimal (with respect to $\subseteq$) in A.

By $\mathrm{COMP}(S)$ we denote the set of all strong components of S.

The condition (iii) of the definition of the strong component of S we can also formulate in the following way:

B and C are minimal (with respect to $\subseteq$) in A if and only if: (i) if $(B, C) \in \mathrm{COMP}(S)$ then $(B - \{a\}, C) \notin \mathrm{COMP}(S)$ for any $a \in B$, (ii) if $(B, C) \in \mathrm{COMP}(S)$ then $(B, C - \{a\}) \notin \mathrm{COMP}(S)$ for any $a \in C$.

2.3. Rules in Decision Tables

Rules express some of the relationships between values of the attributes described in the information systems. This subsection contains the definition of rules as well as other related concepts.

Let $S = (U, A)$ be an information system and let $B \subset A$. For every $a \notin B$, we define a function $d_a^B : U \to P(V_a)$ such that $d_a^B(u) = \{v \in V_a$: there exists $u' \in U$ u' $ind(B)$ u and $a(u') = v\}$, where $P(V_a)$ denotes the powerset of V_a. Hence, $d_a^B(u)$ is the set of all the values of the attribute a on objects indiscernible with u by attributes from B. If the set $d_a^B(u)$ has only

one element, this means that the value $a(u)$ is uniquely defined by the values of attributes from B on u.

A *rule* over A and V is any expression of the following form:

$$(1) \quad a_{i_1} = v_{i_1} \wedge ... \wedge a_{i_r} = v_{i_r} \Rightarrow a_p = v_p$$

where $a_p, a_{i_j} \in A, v_p, v_{i_j} \in V_{a_{i_j}}$ for $j = 1, ..., r$.

A rule of the form (1) is called *trivial* if $a_p = v_p$ appears also on the left hand side of the rule. The rule (1) is *true in* S (or in short: is *true*) if

$$\emptyset \neq \| \ a_{i_1} = v_{i_1} \wedge ... \wedge a_{i_r} = v_{i_r} \ \| \subseteq \| \ a_p = v_p \ \|$$

The fact that the rule (1) is true in S is denoted in the following way:

$$(2) \quad a_{i_1} = v_{i_1} \wedge ... \wedge a_{i_r} = v_{i_r} \overset{\Rightarrow}{\mathrm{S}} a_p = v_p.$$

If (2) holds we also say that the values (local states) $v_{i_1}, ..., v_{i_r}, v_p$ of processes $a_{i_1}, ..., a_{i_r}, a_p$ can *coexist* in S.

By $D(S)$ we denote the set of all rules true in S.

Let $S = (U, A \cup \{d\})$ be a decision table. A rule $\tau \Rightarrow d = v$, where τ is a term over A and V, d is a decision attribute of S, $v \in V_d$, is called a *decision rule*.

Let $R \subseteq D(S)$. An information vector $\mathbf{v} = (\mathbf{v}_1, ..., \mathbf{v}_m)$ is *consistent* with R if and only if for any rule $a_{i_1} = v_{i_1} \wedge ... \wedge a_{i_r} = v_{i_r} \overset{\Rightarrow}{\mathrm{S}} a_p = v_p$ in R if $\mathbf{v}_{i_j} = v_{i_j}$ for $j = 1, ..., r$ then $v_p = \mathbf{v_p}$.

Let $S' = (U', A')$ be an U'-extension of $S = (U, A)$. We say that S' is a *consistent extension* of S if and only if $D(S) \subseteq D(S')$. S' is called a *maximal* consistent extension of S if and only if S' is a consistent extension of S and any consistent extension S'' of S is a restriction of S'.

We use Boolean reasoning approach to the rule generation [25].

The Boolean reasoning approach [2], due to G. Boole, is a general problem solving method consisting of the following steps: (i) construction of a Boolean function corresponding to a given problem; (ii) computation of prime implicants of the Boolean function; (iii) interpretation of prime implicants leading to the solution of the problem. It turns out that this method can be also applied to the generation of rules.

2.4. Reduction of Attributes

Let $S = (U, A)$ be an information system. Any minimal subset $B \subseteq A$ such that $ind(B) = ind(A)$ is called a *reduct* in the information system S [17]. The set of all reducts in S is denoted by RED(S).

Now we recall two basic notions, namely those of *discernibility matrix* and *discernibility function* [26], which will help to compute minimal forms of rules with respect to the number of attributes on the left hand side of the rules.

Let $S = (U, A)$ be an information system, and let us assume that $U = \{u_1, ..., u_n\}$, and $A = \{a_1, ..., a_m\}$. By $M(S)$ we denote an $n \times n$ matrix (c_{ij}), called the *discernibility matrix* of S, such that $c_{ij} = \{a \in A : a(u_i) \neq a(u_j)\}$ for $i, j = 1, ..., n$.

Intuitively an entry c_{ij} consists of all the attributes discerning objects u_i and u_j. Since $M(S)$ is symmetric and $c_{ii} = \emptyset$ for $i = 1, ..., n, M(S)$ can be represented using only elements in the lower triangular part of $M(S)$, i.e., for $1 \leq j < i \leq n$.

With every discernibility matrix $M(S)$ one can uniquely associate a *discernibility function* $f_{M(S)}$, defined as follows.

A *discernibility function* $f_{M(S)}$ for an information system S is a Boolean function of m propositional variables $a_1^*, ..., a_m^*$ (where $a_i \in A$ for $i = 1, ..., m$) defined as the conjunction of all expressions $\bigvee c_{ij}^*$, where $\bigvee c_{ij}^*$ is the disjunction of all elements of $c_{ij}^* = \{a^* : a \in c_{ij}\}$, where $1 \leq j < i \leq n$ and $c_{ij} \neq \emptyset$. In the sequel we write a instead of a^*.

Proposition 2.1 gives an important property which enables us to compute all reducts of S.

Proposition 2.1. [26] Let $S = (U, A)$ be an information system, and let $f_{M(S)}$ be a discernibility function for S. Then the set of all prime implicants [34] of the function $f_{M(S)}$ determines the set RED(S) of all reducts of S, i.e., $a_{i_1} \wedge ... \wedge a_{i_k}$ is a prime implicant of $f_{M(S)}$ if and only if $\{a_{i_1}, ..., a_{i_k}\} \in$RED($S$).

2.5. Minimal Rules in Decision Tables

In this section, we present a method for generating the minimal form of rules in decision tables.

The method is based on the idea of Boolean reasoning [2] applied to discernibility matrices defined in [26] and modified here for our purposes. This section presents the first step in the construction of a concurrent model of knowledge embedded in a given decision table.

Let $S = (U, A \cup \{d\})$ be a decision table and $d \notin A$. We are looking for all minimal rules (i.e., with minimal left hand sides) in S of the form:

$$(1) \quad a_{i_1} = v_{i_1} \wedge ... \wedge a_{i_r} = v_{i_r} \underset{S}{\Rightarrow} a_p = v_p,$$

where $a \in A \cup \{d\}, v \in V_a, a_{i_j} \in A$, and $v_{i_j} \in V_{a_{i_j}}$ for $j = 1, ..., r$.

The above rules express functional dependencies between the values of the conditional attributes of S as well as functional dependencies between the values of the conditional attributes of S and the values of the decision attribute of S. These rules are computed from systems of the form $S' = (U, B \cup \{a\})$ where $B \subset A$ and $a \in A - B$ or $a = d$.

First, for every $v \in V_a$, $u_l \in U$ such that $d_a^B(u_l) = \{v\}$ a modification $\boldsymbol{M}(S'; a, v, u_l)$ of the discernibility matrix is computed from $M(S')$. By $\boldsymbol{M}(S'; a, v, u_l) = (c_{ij}^*)$, (or $\boldsymbol{M}$, in short) we denote the matrix obtained from $M(S')$ in the following way:

if $i \neq l$ **then** $c_{ij}^* = \emptyset$;
if $c_{lj} \neq \emptyset$ **and** $d_a^B(u_j) \neq \{v\}$ **then** $c_{lj}^* = c_{lj} \cap B$ **else** $c_{lj}^* = \emptyset$.

Next, we compute the discernibility function f_M and the prime implicants of f_M taking into account the non-empty entries of the matrix $\boldsymbol{M}$ (when all entries c_{ij}^* are empty we assume f_M to be always true).

Finally every prime implicant $a_{i_1} \wedge ... \wedge a_{i_r}$ of f_M determines a rule $a_{i_1} = v_{i_1} \wedge ... \wedge a_{i_r} = v_{i_r} \underset{S}{\Rightarrow} a = v$, where $a_{i_j}(u_l) = v_{i_1}$ for $j = 1, ..., r$, $a(u_l) = v$.

The set of all rules constructed in this way for any $a \in A \cup \{d\}$ is denoted by $\mathrm{OPT}(S, a)$.

We put $\mathrm{OPT}(S) = \bigcup\{\mathrm{OPT}(S, a) : a \in A \cup \{d\}\}$.

For any $a \in A \cup \{d\}$ and $u \in U$ we take $B = A$, if $a = d$; $B = (A \cup \{d\}) - \{a\}$ otherwise and we take $v = a(u)$. We compute all minimal rules true in $S' = (U, B \cup \{a\})$ of the form $\tau \Rightarrow a = v$, where τ is a term in conjunctive form over B and $V_B = \bigcup_{a \in B} V_a$, with a minimal number of descriptors in any conjunct. To obtain all possible functional dependencies between the attribute values it is necessary to repeat this process for all possible values of a and for all remaining attributes from $A \cup \{d\}$.

2.6. Relationships between Dependencies in Information System and Reducts

The important relationships between the reducts and the dependencies are given by the propositions listed below.

Proposition 2.2. [17] Let S=(U, A) be an information system and let $B \in$RED(S). If $A - B \neq \emptyset$ then $B \underset{\mathrm{S}}{\rightarrow} A - B$.

Proposition 2.3. [17] If $B \underset{\mathrm{S}}{\rightarrow} C$ then $B \underset{\mathrm{S}}{\rightarrow} C'$, for every $\emptyset \neq C' \subseteq C$. In particular, $B \underset{\mathrm{S}}{\rightarrow} C$ implies $B \underset{\mathrm{S}}{\rightarrow} \{a\}$, for every $a \in C$.

Proposition 2.4. Let $S = (U, A)$ be an information system, and $B, C \subseteq A$. The sets B and C are equivalent in S if and only if the following conditions are satisfied: (i) C is {b}-reduct of S for any $b \in B$ [17], (ii) B is {c}-reduct of S for any $c \in C$.

From Proposition 2.4 follows a method for computing the set of all strong components of a given information system $S = (U, A)$:

PROCEDURE for computing COMP(S):

`Input`: An information system $S = (U, A)$, and the set RED(S) of all reducts of S.

`Output`: Strong components of S, i.e., the set COMP(S).

Step 1. For each $R \in \mathrm{RED}(S)$ compute minimal subsets $B \subseteq R$ such that $B \underset{S}{\rightarrow} \{a\}$, for any $a \in A - R$, i.e., for each subsystem $S' = (U, R \cup \{a\})$ of S with $a \in A - R$ compute the discernibility function $f_{M(S')}$. In this step we compute the so called $\{a\}$ - reducts of R, for $a \in A - R$.

Step 2. For each $R \in \mathrm{RED}(S)$ compute minimal subsets $C \subseteq A - R$ such that $C \underset{S}{\rightarrow} \{b\}$, for any $b \in R$, i.e., for each subsystem S" $= (U, (A - R) \cup \{b\})$ of S with $b \in R$ compute the discernibility function $f_{M(S")}$. In this step we compute $\{b\}$ - reducts of $A - R$, for $b \in R$.

Step 3. Choose all pairs (B,C) such that $B \cap C = \emptyset$ and $B \underset{S}{\rightarrow} \{a\}$ has computed in *Step 1* for any $a \in C$ and $C \underset{S}{\rightarrow} \{b\}$ has computed in *Step 2* for any $b \in B$.

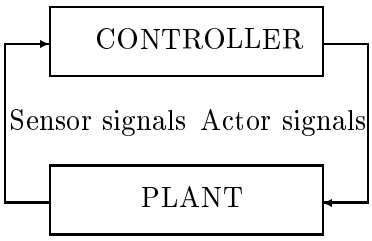

Figure 1. Scheme of a control loop

3. A Standard Scheme of a Controlled System

To make the paper self-contained, the running example, adopted from [4], is shortly sketched. In Figure 1 we show a standard scheme of a controlled system. The system consists of two main elements: the plant and the controller. The plant is the system which is to be controlled by the controller, and the controller is the system which imposes control on the plant. A typical control loop includes these two elements and it is closed by a flow of sensor signals and actor signals as depicted in Figure 1.

4. The Control Model

The traditional approach to the control design of systems requires that a control strategy to achieve a specific goal be known in advance and clearly described in the form of a control algorithm. In our approach the control strategy will be characterized by parameters of the process which are directly related to the imposed control strategy. The values of parameters define the observable states of the process. The proper control decision corresponds to each observable state. Hence, from the control viewpoint, a control process can be characterized by:

(i) the space of observable states determined by the state parameters,
(ii) the space of control determined by control parameters.

In the rough set approach, the space of observable states is explicitly defined by a finite set of condition attributes, and the space of control is explicitly determined by a finite set of decision attributes. For our purposes it will be sufficient to consider decision tables with one decision only because one can always transform a decision table with more than one decision into a decision table with exactly one decision by simple coding. The knowledge about the control is represented by a set of decision rules. In such notation the rules represent dependencies between the set of conditions and the decision. Hence, a process control can be represented in the form of a decision table. The resulting decision table can be analyzed from many viewpoints. We shall analyze the decision table representing control in order to obtain:

1. Elimination of those condition attributes which do not influence the relations between the set of values of condition attributes and the decision.

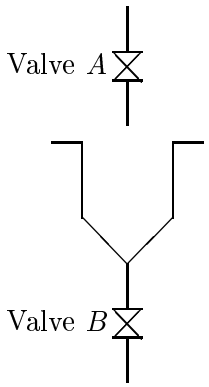

Figure 2. A scheme of the dosing tank

2. Computing the dependencies between the set of values of condition attributes and the decision, and vice versa.
3. Reduction of redundant rules.
4. Checking whether there are no contradictory decision rules, i.e., rules in which different values of the decision correspond to the same values of condition attributes.

5. Illustrative Example

In this section, we discuss how to pass from the informal description of the behavior of an exemplary plant to the synthesis of decision rules for the control of that plant, and next to a concurrent control representation (a control program) in the form a Petri net using rough set methods.

At first we discuss a simplified version of a plant presented in [4]. A scheme of the plant is illustrated in Figure 2. It consists of the dosing tank and two valves. The dosing tank can be filled and discharged by operating the valves denoted by A and B. These valves are operated by a human operator as well as by a controller. The level in the tank as well as the positions of the valves are indicated by switching sensors. It is assumed that a control ensures the following behavior:

1. Both valves can not be opened at the same time.
2. The tank cannot be allowed to overflow.
3. The tank must be filled completely before it must be emptied completely.

It is easy to see that the observable state of the plant can be characterized by the following condition attributes: oA - the valve A operated by the human operator, oB - the valve B operated by the human operator, vA - the valve A, vB - the valve B, tk - the tank.

The characteristic states of the control process (controller) are: ini - the initial state of controller, fil - filling of the tank, wai - waiting for opening vB, emp - emptying of the tank, i.e., $V_{ctrl} = \{ini, fil, wai, emp\}$.

U/A	oA	oB	vA	vB	tk	$ctrl$
u_1	c	c	c	c	e	ini
u_2	o	c	o	c	e	fil
u_3	o	c	o	c	pf	fil
u_4	c	c	c	c	f	wai
u_5	c	o	c	o	f	emp
u_6	c	o	c	o	pf	emp

Table 1 An example of a decision table S

We assume that:

1) the set of attribute values oA, oB, vA and vB are equal and they have the same elements, i.e., $V_{oA} = V_{oB} = V_{vA} = V_{vB} = \{c, o\}$ where c - denotes that the valve A (B) is closed, o - denotes that the valve A (B) is open;

2) the set of attribute values tk consists of elements e, pf and f, where e - denotes that the tank is empty, pf - denotes that the tank is partially emptying, f - denotes that the tank is full, i.e., $V_{tk} = \{e, pf, f\}$.

The above description of the considered control of the plant in the form of a decision table $S = (U, A \cup \{d\})$ shown in Table 1, where the set of objects $U = \{u_1, u_2, u_3, u_4, u_5, u_6\}$, the set of condition attributes $A = \{oA, oB, vA, vB, tk\}$. The decision is denoted by $ctrl$.

6. Rough Sets Analysis

In this section, we present an illustrative example of the basic stages of the rough set analysis used in our method.

6.1. Reduction of Parameters

Rough set methods offer possibility to reduce dispensable information.

¿From Proposition 2.1 follows a method for computing the set of all reducts of a given information system. For the decision table S presented in Table 1, we obtain the discernibility matrix $M(S)$ presented in Table 2 and the discernibility function $f_{M(S)}$ presented below:

$$f_{M(S)}(oA, oB, vA, vB, tk, ctrl) = tk \wedge (tk \vee ctrl) \wedge (oA \vee vA \vee ctrl) \wedge (oB \vee vB \vee ctrl) \wedge (oA \vee oB \vee vA \vee vB \vee ctrl) \wedge (oA \vee vA \vee tk \vee ctrl) \wedge (oB \vee vB \vee tk \vee ctrl) \wedge (oA \vee oB \vee vA \vee vB \vee tk \vee ctrl).$$

After simplification (using the absorption laws), we get the minimal disjunctive normal form of the discernibility function as follows:

$$f_{M(S)}(oA, oB, vA, vB, tk, ctrl) = (oA \wedge oB \wedge tk) \vee (oA \wedge vB \wedge tk) \vee (oB \wedge vA \wedge tk) \vee (vA \wedge vB \wedge tk) \vee (tk \wedge ctrl).$$

U	u_1	u_2	u_3	u_4	u_5	u_6
u_1						
u_2	$oA, vA, ctrl$					
u_3	$oA, vA, tk, ctrl$	tk				
u_4	$tk, ctrl$	$oA, vA, tk, ctrl$	$oA, vA, tk, ctrl$			
u_5	$oB, vB, tk, ctrl$	$oA, oB, vA, vB, tk, ctrl$	$oA, oB, vA, vB, tk, ctrl$	$oB, vB, ctrl$		
u_6	$oB, vB, tk, ctrl$	$oA, oB, vA, vB, tk, ctrl$	$oA, oB, vA, vB, ctrl$	$oB, vB, tk, ctrl$	tk	

Table 2 The discernibility matrix $M(S)$ for the decision table S

There are five reducts: $R_1 = \{oA, oB, tk\}$, $R_2 = \{oA, vB, tk\}$, $R_3 = \{oB, vA, tk\}$, $R_4 = \{vA, vB, tk\}$ and $R_5 = \{tk, ctrl\}$ of the system. Thus $\mathrm{RED}(S) = \{R_1, R_2, R_3, R_4, R_5\}$.

6.2. Extraction of Dependencies

Extraction of all strong components from a given data table allows us among others to delete the redundant rules from the set of rules representing knowledge included in the table.

By Proposition 2.2, we have for the system S the dependencies:

$$\{oA, oB, tk\} \overrightarrow{\mathrm{S}} \{vA, vB, ctrl\}, \{oA, vB, tk\} \overrightarrow{\mathrm{S}} \{vA, oB, ctrl\}, \{oB, vA, tk\} \overrightarrow{\mathrm{S}} \{oA, vB, ctrl\},$$
$$\{vA, vB, tk\} \overrightarrow{\mathrm{S}} \{oA, oB, ctrl\}, \{tk, ctrl\} \overrightarrow{\mathrm{S}} \{oA, oB, vA, vB\}.$$

Now, applying the procedure for computing of the strong components from the decision table S we obtain six strong components in S of the form: (B, C), (D, E), (F, G), (F, H), (F, I), (F, J) with $B = \{oA\}$, $C = \{vA\}$, $D = \{oB\}$, $E = \{vB\}$, $F = \{ctrl\}$, $G = \{oA, oB, tk\}$, $H = \{oA, vB, tk\}$, $I = \{oB, vA, tk\}$, $J = \{vA, vB, tk\}$.

6.3. Generation of Rules

In the subsection, we describe how one can pass from a given decision table to the set of all rules corresponding to non-trivial functional dependencies in S. In our example the rules generated from the given decision table S represent knowledge about the control of the plant.

Let us consider a decision table S and its discernibility matrix presented in Table 2. We compute the set of rules corresponding to non-trivial functional dependencies between the values of conditions and the decision values as well as the set of rules corresponding to functional dependencies between the values of conditions of that decision table. In both cases we apply the method presented in Subsection 2.5.

Let us start by computing of the decision rules corresponding to the conditions $A = \{oA, oB, vA, vB, tk\}$ and the decision $ctrl$.

We have the decision table $S = (U, A \cup \{ctrl\})$ from which we compute the decision rules mentioned above.

In Table 3 the values of the function d^A_{ctrl} are also given. The discernibility matrix $\boldsymbol{M}(S; ctrl, v, u_l)$ where $v \in V_{ctrl}$, $u_l \in U$, $l = 1, 2, 3, 4, 5, 6$, obtained from $M(S)$ in the above way is

U/A	oA	oB	vA	vB	tk	$ctrl$	d^A_{ctrl}
u_1	c	c	c	c	e	ini	$\{ini\}$
u_2	o	c	o	c	e	fil	$\{fil\}$
u_3	o	c	o	c	pf	fil	$\{fil\}$
u_4	c	c	c	c	f	wai	$\{wai\}$
u_5	c	o	c	o	f	emp	$\{emp\}$
u_6	c	o	c	o	pf	emp	$\{emp\}$

Table 3 The decision table S with the function d^A_{ctrl}

U	u_1	u_2	u_3	u_4	u_5	u_6
u_1		oA, vA	oA, vA, tk	tk	oB, vB, tk	oB, vB, tk
u_2	oA, vA			oA, vA, tk	oA, oB, vA, vB, tk	oA, oB, vA, vB, tk
u_3	oA, vA, tk			oA, vA, tk	oA, oB, vA, vB, tk	oA, oB, vA, vB
u_4	tk	oA, vA, tk	oA, vA, tk		oB, vB	oB, vB, tk
u_5	oB, vB, tk	oA, oB, vA, vB, tk	oA, oB, vA, vB, tk	oB, vB		
u_6	oB, vB, tk	oA, oB, vA, vB, tk	oA, oB, vA, vB	oB, vB, tk		

Table 4 The discernibility matrix $\boldsymbol{M}(S; ctrl, v, u_l)$ for the matrix $M(S)$

presented in Table 4.

The discernibility functions corresponding to the values of the function d^A_{ctrl} are the following:

Case 1. For $d^A_{ctrl}(u_1) = \{ini\} : (oA \vee vA) \wedge (oA \vee vA \vee tk) \wedge tk \wedge (oB \vee vB \vee tk) \equiv (oA \vee vA) \wedge tk \equiv oA \wedge tk \vee vA \wedge tk$.

We consider non-empty entries of the column labeled by u_1 (see, Table 4), i.e., oA, vA; oA, vA, tk; tk; oB, vB, tk; and oB, vB, tk, next oA, vA, tk, oB, vB are treated as Boolean variables and the disjunctions $oA \vee vA$, $oA \vee vA \vee tk$, tk, and $oB \vee vB \vee tk$ are constructed from these entries; finally, we take the conjunction of all the computed disjunctions to obtain the discernibility function corresponding to $\boldsymbol{M}(S; ctrl, ini, u_1)$.

Case 2. For $d^A_{ctrl}(u_2) = \{fil\} : (oA \vee vA) \wedge (oA \vee vA \vee tk) \wedge (oA \vee oB \vee vA \vee vB \vee tk) \equiv oA \vee vA$.

Case 3. For $d^A_{ctrl}(u_3) = \{fil\} : (oA \vee vA \vee tk) \wedge (oA \vee oB \vee vA \vee vB \vee tk) \wedge (oA \vee oB \vee vA \vee vB) \equiv oA \vee vA \vee oB \wedge tk \vee vB \wedge tk$.

Case 4. For $d^A_{ctrl}(u_4) = \{wai\} : tk \wedge (oA \vee vA \vee tk) \wedge (oB \vee vB) \wedge (oB \vee vB \vee tk) \equiv tk \wedge oB \vee tk \wedge vB$.

Case 5. For $d^A_{ctrl}(u_5) = \{emp\} : (oB \vee vB \vee tk) \wedge (oA \vee oB \vee vA \vee vB \vee tk) \wedge (oB \vee vB) \equiv oB \vee vB$.

Case 6. For $d^A_{ctrl}(u_5) = \{emp\} : (oB \vee vB \vee tk) \wedge (oA \vee oB \vee vA \vee vB \vee tk) \wedge (oA \vee oB \vee vA \vee vB)$

$\wedge\ (oB \vee vB \vee tk) \equiv (oB \vee vB \vee tk) \wedge (oA \vee oB \vee vA \vee vB) \equiv oB \vee vB \vee oA \wedge tk \vee vA \wedge tk.$

Hence, we obtain the following decision rules:

For *Case 1*: $oA(c) \wedge tk(e) \underset{S}{\Rightarrow} ctrl(ini)$, $vA(c) \wedge tk(e) \underset{S}{\Rightarrow} ctrl(ini)$.

For *Case 2*: $oA(o) \underset{S}{\Rightarrow} ctrl(fil), vA(o) \underset{S}{\Rightarrow} ctrl(fil)$.

For *Case 3*: $oA(o) \underset{S}{\Rightarrow} ctrl(fil), vA(o) \underset{S}{\Rightarrow} ctrl(fil)$, $oB(c) \wedge tk(pf) \underset{S}{\Rightarrow} ctrl(fil)$, $vB(c) \wedge tk(pf) \underset{S}{\Rightarrow} ctrl(fil)$.

For *Case 4*: $vB(c) \wedge tk(f) \underset{S}{\Rightarrow} ctrl(wai), oB(c) \wedge tk(f) \underset{S}{\Rightarrow} ctrl(wai)$.

For *Case 5*: $oB(o) \underset{S}{\Rightarrow} ctrl(emp), vB(o) \underset{S}{\Rightarrow} ctrl(emp)$.

For *Case 6*: $oA(c) \wedge tk(pf) \underset{S}{\Rightarrow} ctrl(emp)$, $vA(c) \wedge tk(pf) \underset{S}{\Rightarrow} ctrl(emp)$,$oB(o) \underset{S}{\Rightarrow} ctrl(emp)$, $vB(o) \underset{S}{\Rightarrow} ctrl(emp)$.

In order to obtain the remaining rules corresponding to all non-trivial functional dependencies between attribute values of the decision table it is sufficient to consider the following five subsystems $(U, B \cup \{tk\})$, $(U, C \cup \{vB\})$, $(U, D \cup \{vA\})$, $(U, E \cup \{oB\})$, $(U, F \cup \{oA\})$ of S, where $B = \{oA, oB, vA, vB, ctrl\}, C = \{oA, oB, vA, tk, ctrl\}$, $D = \{oA, oB, vB, tk, ctrl\}$, $E = \{oA, vA, vB, tk, ctrl\}$, and $F = \{oB, vA, vB, tk, ctrl\}$.

Proceeding analogously as above, eventually, we obtain the set OPT(S) of rules corresponding to all non-trivial functional dependencies in the considered decision table S:
$oA(c) \wedge tk(e) \underset{S}{\Rightarrow} ctrl(ini)$, $vA(c) \wedge tk(e) \underset{S}{\Rightarrow} ctrl(ini)$, $oA(o) \underset{S}{\Rightarrow} ctrl(fil)$, $vA(o) \underset{S}{\Rightarrow} ctrl(fil)$,
$oB(c) \wedge tk(pf) \underset{S}{\Rightarrow} ctrl(fil)$, $vB(c) \wedge tk(pf) \underset{S}{\Rightarrow} ctrl(fil)$, $vB(c) \wedge tk(f) \underset{S}{\Rightarrow} ctrl(wai)$,
$oB(c) \wedge tk(f) \underset{S}{\Rightarrow} ctrl(wai)$, $oB(o) \underset{S}{\Rightarrow} ctrl(emp)$,
$vB(o) \underset{S}{\Rightarrow} ctrl(emp)$, $oA(c) \wedge tk(pf) \underset{S}{\Rightarrow} ctrl(emp)$, $vA(c) \wedge tk(pf) \underset{S}{\Rightarrow} ctrl(emp)$,
$ctrl(ini) \underset{S}{\Rightarrow} tk(e)$, $ctrl(wai) \underset{S}{\Rightarrow} tk(f)$, $ctrl(ini) \underset{S}{\Rightarrow} vB(c)$, $ctrl(fil) \underset{S}{\Rightarrow} vB(c)$, $ctrl(wai) \underset{S}{\Rightarrow} vB(c)$,
$ctrl(emp) \underset{S}{\Rightarrow} vB(o)$, $ctrl(ini) \underset{S}{\Rightarrow} vA(c)$, $ctrl(fil) \underset{S}{\Rightarrow} vA(o)$, $ctrl(wai) \underset{S}{\Rightarrow} vA(c)$, $ctrl(emp) \underset{S}{\Rightarrow} vA(c)$,
$ctrl(ini) \underset{S}{\Rightarrow} oB(c)$, $ctrl(fil) \underset{S}{\Rightarrow} oB(c)$, $ctrl(wai) \underset{S}{\Rightarrow} oB(c)$, $ctrl(emp) \underset{S}{\Rightarrow} oB(o)$, $ctrl(ini) \underset{S}{\Rightarrow} oA(c)$,
$ctrl(fil) \underset{S}{\Rightarrow} oA(o)$, $ctrl(wai) \underset{S}{\Rightarrow} oA(c)$, $ctrl(emp) \underset{S}{\Rightarrow} oA(c)$, $oB(c) \underset{S}{\Rightarrow} vB(c)$, $oB(o) \underset{S}{\Rightarrow} vB(o)$,
$tk(e) \underset{S}{\Rightarrow} vB(c)$, $oA(o) \underset{S}{\Rightarrow} vB(c)$, $vA(o) \underset{S}{\Rightarrow} vB(c)$, $oA(c) \wedge tk(pf) \underset{S}{\Rightarrow} vB(o)$, $vA(c) \wedge tk(pf) \underset{S}{\Rightarrow} vB(o)$,
$oA(c) \underset{S}{\Rightarrow} vA(c)$, $oA(o) \underset{S}{\Rightarrow} vA(o)$, $oB(c) \wedge tk(pf) \underset{S}{\Rightarrow} vA(o)$, $vB(c) \wedge tk(pf) \underset{S}{\Rightarrow} vA(o)$, $tk(f) \underset{S}{\Rightarrow} vA(c)$,
$oB(o) \underset{S}{\Rightarrow} vA(c)$, $vB(o) \underset{S}{\Rightarrow} vA(c)$, $vB(c) \underset{S}{\Rightarrow} oB(c)$, $vB(o) \underset{S}{\Rightarrow} oB(o)$, $tk(e) \underset{S}{\Rightarrow} oB(c)$, $oA(o) \underset{S}{\Rightarrow} oB(c)$,
$vA(o) \underset{S}{\Rightarrow} oB(c)$, $oA(c) \wedge tk(pf) \underset{S}{\Rightarrow} oB(o)$, $vA(c) \wedge tk(pf) \underset{S}{\Rightarrow} oB(o)$, $vA(c) \underset{S}{\Rightarrow} oA(c)$, $vA(o) \underset{S}{\Rightarrow} oA(o)$,
$oB(c) \wedge tk(pf) \underset{S}{\Rightarrow} oA(o)$, $vB(c) \wedge tk(pf) \underset{S}{\Rightarrow} oA(o)$, $tk(f) \underset{S}{\Rightarrow} oA(c)$, $oB(o) \underset{S}{\Rightarrow} oA(c)$, $vB(o) \underset{S}{\Rightarrow} oA(c)$.

Our approach to rule generation is based on procedures for the computation of reduct sets. It is known that in general the reduct set can be of exponential complexity with respect to the number of attributes (conditions). Moreover the minimal reduct problem is NP-hard. Neverthe-

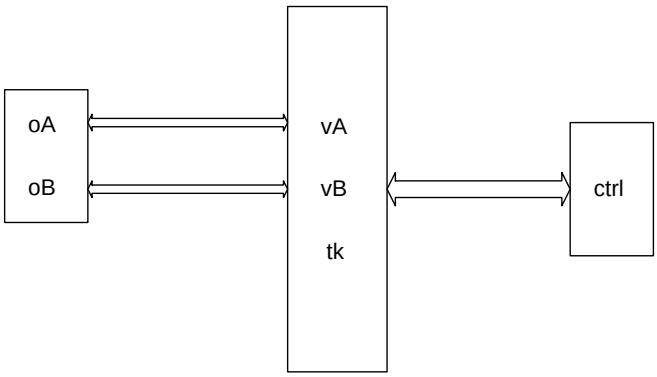

Figure 3. Scheme of a reduction of redundant rules with respect to equivalent dependencies in S

less, there are several methodologies allowing to deal with this problem in practical applications [1],[6],[12],[13].

6.4. Reduction of Redundant Rules

Now we consider the rule reduction problem. In many cases, we have the redundant sets of rules. If we find the strong components of a given decision table then we can use only some part of rules representing all non-trivial functional dependencies between values of attributes from those sets. We explain this using an example presented below.

Let us consider again the set OPT(S) of rules corresponding to all non-trivial functional dependencies in the given decision table S.

Since the following sets of attributes $\{oA\}$ and $\{vA\}$, $\{oB\}$ and $\{vB\}$, $\{ctrl\}$ and $\{oA, oB, tk\}$, $\{ctrl\}$ and $\{oA, vB, tk\}$, $\{ctrl\}$ and I=$\{oB, vA, tk\}$, $\{ctrl\}$ and J=$\{vA, vB, tk\}$ are equivalent, respectively, it is sufficient to take into account, for instance, the rules computed from the dependencies:

Case 1. $\{oA\} \overset{\longleftrightarrow}{\mathrm{S}} \{vA\}$; $\{oB\} \overset{\longleftrightarrow}{\mathrm{S}} \{vB\}$. (Strong components of S.)

Case 2. $\{vA, vB, tk\} \overset{\longrightarrow}{\mathrm{S}} \{ctrl\}$. (Sensor signals, see, Figure 1.)

Case 3. $\{ctrl\} \overset{\longrightarrow}{\mathrm{S}} \{vA, vB, tk\}$. (Actor signals, see, Figure 1.)

Case 4. $tk(e) \underset{S}{\Rightarrow} vB(c)$, $tk(f) \underset{S}{\Rightarrow} vA(c)$, $vA(c) \wedge tk(pf) \underset{S}{\Rightarrow} vB(o)$, $vB(c) \wedge tk(pf) \underset{S}{\Rightarrow} vA(o)$, $vB(o) \underset{S}{\Rightarrow} vA(c)$, $vA(o) \underset{S}{\Rightarrow} vB(c)$. (Internal dependencies.)

If we omit in the set OPT(S) of rules corresponding to all non-trivial functional dependencies in the considered decision table S in which on both left or right hand side appear descriptors with names oA, oB then we obtain the reduced set of rules without the redundant rules with respect to equivalent dependencies in the given data table S (see, Figure 3).

It means that the system of generators presented below generate all non-trivial dependencies

between attribute values of the given table S:

For *Case 1.* $oB(c) \underset{S}{\Rightarrow} vB(c)$, $oB(o) \underset{S}{\Rightarrow} vB(o)$, $vB(c) \underset{S}{\Rightarrow} oB(c)$, $vB(o) \underset{S}{\Rightarrow} oB(o)$, $oA(c) \underset{S}{\Rightarrow} vA(c)$, $oA(o) \underset{S}{\Rightarrow} vA(o)$, $vA(c) \underset{S}{\Rightarrow} oA(c)$, $vA(o) \underset{S}{\Rightarrow} oA(o)$.

For *Case 2.* $vA(c) \wedge tk(e) \underset{S}{\Rightarrow} ctrl(ini)$, $vA(o) \underset{S}{\Rightarrow} ctrl(fil)$, $vB(c) \wedge tk(pf) \underset{S}{\Rightarrow} ctrl(fil)$, $vB(c) \wedge tk(f) \underset{S}{\Rightarrow} ctrl(wai)$, $vB(o) \underset{S}{\Rightarrow} ctrl(emp)$, $vA(c) \wedge tk(pf) \underset{S}{\Rightarrow} ctrl(emp)$.

For *Case 3.* $ctrl(ini) \underset{S}{\Rightarrow} tk(e)$, $ctrl(wai) \underset{S}{\Rightarrow} tk(f)$, $ctrl(ini) \underset{S}{\Rightarrow} vB(c)$, $ctrl(fil) \underset{S}{\Rightarrow} vB(c)$, $ctrl(wai) \underset{S}{\Rightarrow} vB(c)$, $ctrl(emp) \underset{S}{\Rightarrow} vB(o)$, $ctrl(ini) \underset{S}{\Rightarrow} vA(c)$, $ctrl(fil) \underset{S}{\Rightarrow} vA(o)$, $ctrl(wai) \underset{S}{\Rightarrow} vA(c)$, $ctrl(emp) \underset{S}{\Rightarrow} vA(c)$.

For *Case 4.* $tk(e) \underset{S}{\Rightarrow} vB(c)$, $tk(f) \underset{S}{\Rightarrow} vA(c)$, $vA(c) \wedge tk(pf) \underset{S}{\Rightarrow} vB(o)$, $vB(c) \wedge tk(pf) \underset{S}{\Rightarrow} vA(o)$, $vB(o) \underset{S}{\Rightarrow} vA(c)$, $vA(o) \underset{S}{\Rightarrow} vB(c)$.

By using strong components of the given decision table we have reduced the set of rules from 58 up to 30 ones.

7. Petri Nets

In this section Petri nets [21] are used as a tool for computing a highly parallel program from a given decision table. After modeling a decision table by a Petri net states are identified in the net to an extent allowing to take the appropriate decisions.

At first, we recall some basic concepts and notation from Petri net theory.

7.1. Structure and Dynamics

A *Petri net* contains two types of nodes, *circles* P (places) and *bars* T (transitions). The relationship between the nodes is defined by two sets of relations α, defines the relationship between places and transitions, and β defines the relationship between transitions and places. The relations between nodes are represented by directed arcs. A Petri net N is defined as a quadruple $N = (P, T, \alpha, \beta)$. Such Petri nets are called *ordinary*.

A *marking* m of a Petri net is an assignment of black dots (tokens) to the places of the net for specifying the state of the system being modeled with a Petri net. Tokens are used to define the execution of a Petri net. Places represent storage for input or for output. Transitions represent activities (transformations) which transform input into output.

The number of tokens in a place p_i is denoted by m_i and then $m = (m_1, \ldots, m_l)$, where l is the total number of places of the net. The initial distribution of tokens among the places is called the *initial marking* and is denoted by M. A Petri net N with a marking M is called a *marked Petri net* and it is denoted by (N, M). In this paper, we only use nets in that all markings are binary, i.e., $m(p) \in \{0, 1\}$ for any place p. *Input* and *output places* of a transition are those which are initial nodes of an incoming or terminal nodes of an outgoing arc of the transition, respectively. The dynamic behavior of the system is represented by the *firing* of the

corresponding transition, and the evolution of the system is represented by a *firing sequence* of transitions. In the paper, it is assumed Petri nets are governed by the following *transition (firing) rule*:

(1) A transition t is *enabled* if and only if each input place p of t is marked by one token.

(2) A transition can fire only if it is enabled.

(3) When a transition t fires, a token is removed from each input place p of t, and t adds a token to each output place p' of t.

7.2. Petri Nets with Priorities

We also recall an extension of the Petri net model. Petri nets with *priorities* have been suggested in [21]. Priorities can be associated with transitions so if t and t' are both enabled, then the transition with the highest priority will fire first.

In this paper, we use ordinary Petri nets with priorities, although it could be also possible to use signal/event nets introduced in the paper [30]. We shall assume that there are only transitions with three different priorities: 0, 1, and 2. All transitions corresponding to decisions have the highest priority 2, transitions corresponding to dependencies between the values of conditional attributes 1, and all the remaining - 0.

For more detailed information about Petri nets we refer the reader to [11],[19],[21],[35].

8. The Solution of Control Design Problem

We present a procedure for transforming rules representing a given decision table into a Petri net.

Let $S = (U, A \cup d)$ be a given decision table, and let $R \in \mathrm{RED}(S)$.

PROCEDURE for constructing a Petri net (N_S, M_S) from a given decision table S:

Input: The set of all minimal rules of S corresponding to a reduct R.

Output: A Petri net (N_S, M_S) such that any of its computation leading to decision making has the minimal length (c.f. [27]).

Step 1. Construct a net representing the set of all attributes from R of S and describing decision d of S (sensors).

Step 2. Extend the net obtained in *Step 1* by adding the elements (arcs and transitions) of the net defined by the set of rules representing all non-trivial functional dependencies between the attribute values from R (a propagation of a new information throughout a net).

Step 3. Extend the net obtained in *Step 2* by adding the elements of the net defined by the set of rules representing all non-trivial functional dependencies between the attribute values from R and the decision d (a change of a decision value).

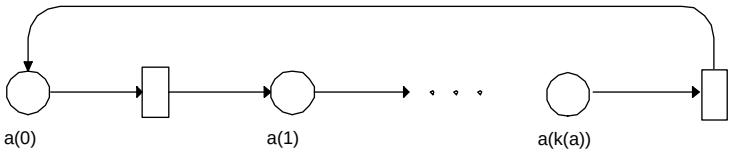

Figure 4. The net corresponding to an attribute a with the priority 0

Step 4. Extend the net obtained in *Step 3* by adding the elements of the net defined by the set of rules representing all non-trivial functional dependencies between the decision value of d and those from R (a change of conditional values from R; to close a control loop).

Step 5. Implement all rules representing all non-deterministic dependencies between attribute values (local states) from R of S.

We present more details of our approach.

Let $S = (U, A \cup \{d\})$ be a decision table, and let $V_a = \{v(0), \cdots, v(k(a))\}$ be a finite set of values for $a \in A$. We assume also that the value sets of all attributes from A are ordered in some way.

For *Step 1.* The construction of a net representing an attribute a from R of S is shown in Figure 4. The place $a(0)$ is called the *start place* of the attribute a. The remaining places of the net represent possible values of the attribute a. The transitions with priority 0 of the net represent the process of identifying the values of the attribute a. We assume that only the start place is marked in the initial marking of the net. (In the following the initial marking of the considered nets is omitted.)

Let us consider again the given decision table S and its reduct $R_4 = \{vA, vB, tk\}$. The conditions vA, vB, tk and the decision $ctrl$ are represented by nets and places shown in Figure 5.

For *Steps 2-4.* The construction of a net representing a set of all rules between: (i) the attribute values of a reduct R of a given decision table, (ii) the attribute values from R and the decision d, (iii) the decision values of d and those from R.

The rules in this case are of the form: (1) $p(j) \underset{S}{\Rightarrow} q(l)$, where $p, q \in A$, $j \in V_p$ and $l \in V_q$, (2) $p(1) \wedge \cdots \wedge p(k) \underset{S}{\Rightarrow} r$, where $k > 1$.

The nets representing the rules (1) and (2) are illustrated in Figures 6 and 7, respectively.

For simplicity of the following pictures let us consider only two rules of the form $vA(o) \underset{S}{\Rightarrow} vB(c)$, $vB(o) \underset{S}{\Rightarrow} vA(c)$ and one rule of the form $vB(c) \wedge tk(pf) \underset{S}{\Rightarrow} ctrl(fil)$ obtained for the reduct R_4 of the considered decision table S.

In Figure 8 a net representing the first two rules is shown, and in Figure 9 it is illustrated the construction of a net representing the third one. The net implementation of non-deterministic

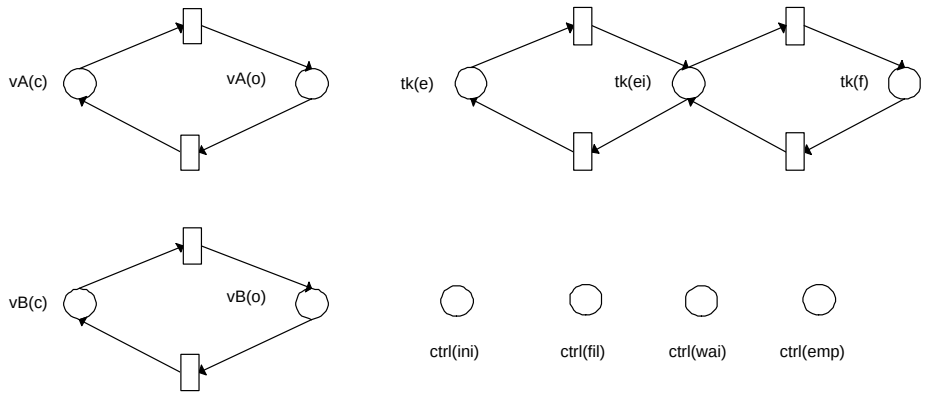

Figure 5. Nets and places representing attributes vA, vB, tk, and $ctrl$

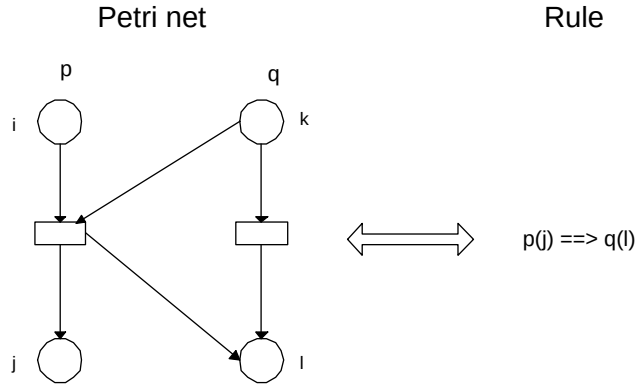

Figure 6. Net representation of rules. Case (1): $p(j) \underset{S}{\Rightarrow} q(l)$, where $p, q \in A$, $j \in V_p$ and $l \in V_q$

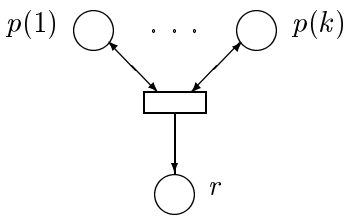

Figure 7. Net representation of rules. Case (2): $p(1) \wedge \cdots \wedge p(k) \underset{S}{\Rightarrow} r$, where $k > 1$

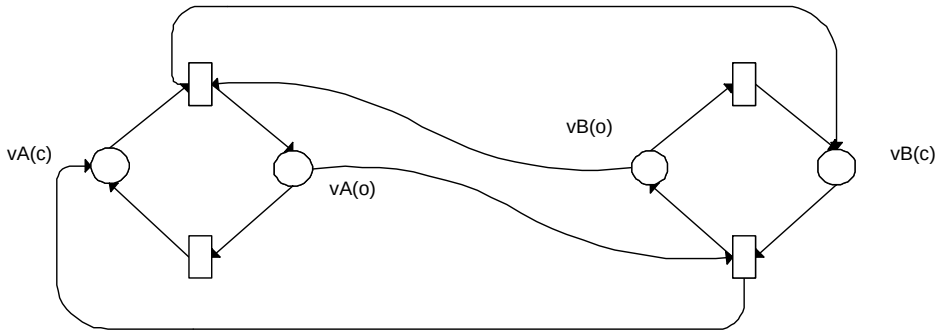

Figure 8. Net representation of rules of the form: $vA(o) \underset{S}{\Rightarrow} vB(c), vB(o) \underset{S}{\Rightarrow} vA(c)$

dependencies between the attribute values is omitted.

In order to construct the whole net representing all rules corresponding to non-trivial functional dependencies between the attribute values of the given decision table it is necessary to repeat our construction for all rules. This construction is also omitted in the paper.

For more detailed information about this implementation we refer the reader to [32].

9. Summary

In this paper, we have demonstrated a methodology for control design of discrete event systems specified by data tables. The presented method seems to be promising for automatic concurrent control design in case of problems which are difficult to model using standard mathematical methods. Our approach is based on rough set approach and Petri nets. It allows to design the control in automatic way. It can be especially applied when behavior requirements of the system are changed in real-time, because programming flexible automation devices by hand is a time-consuming and error-prone task.

Drawing Petri nets by hand one can produce very compact solutions for problems solved rather by small nets. For large models some automatic methods could be accepted even if the nets produced by them are not so compact or small. Comparing the presented example it is possible to see that our method for solving **CDP** can also produce solutions close to those obtained by designers [4].

Our approach allows also to estimate the cost of discrete event system reconstruction when the cost of changed parts of the system is known [32]. It is important to note that our methodology is supposed by software tools, namely the *Rosetta* system [14] for data analysis and rules extraction on the basis of the rough sets theory, as well as the *PN-tools* system for computer aided design and analysis of concurrent models [31]. Although our approach looks quite promising, as demonstrated by the example of dosing tank, more experiments with control processes is needed before the methodology attains its maturity. It will be important to develop meth-

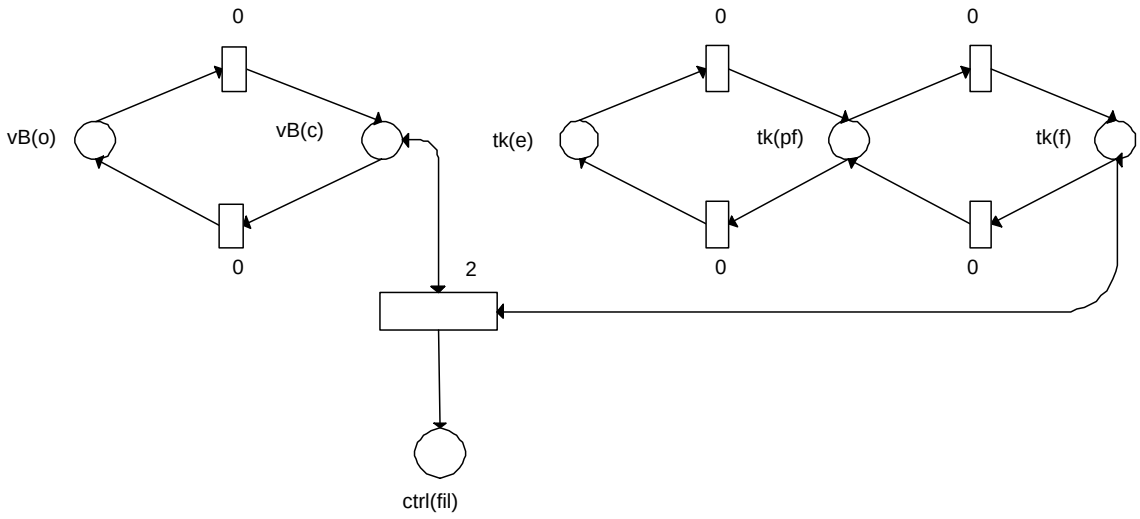

Figure 9. Net representation of rules of the form: $vB(c) \wedge tk(pf) \underset{S}{\Rightarrow} ctrl(fil)$ with the priority 2

ods for converting the received formal models (represented in our case by Petri nets) into the programmable logic controller [35].

We also plan to extend our approach for automatic synthesis of parallel programs from examples [24],[29].

Acknowledgments. The research of Andrzej Skowron and Zbigniew Suraj has been partially supported by the grants from the State Committee for Scientific Research of Poland, ESPRIT-CRIT 2 No. 20288, and from the Wallenberg Foundation. The research of James Peters has been supported by the grant 0170398 from the Natural Sciences and Engineering Research Council of Canada (NSERC).

References

[1] Bazan, J.: A Comparison of Dynamic and Non-Dynamic Rough Set Methods for Extracting Laws from Decision Tables, in: [22], pp. 321-365

[2] Brown, E.M.: *Boolean Reasoning*, Kluwer Academic Publishers, Dordrecht, 1990

[3] Czogała, E., Mrózek, A., and Pawlak, Z.: The idea of rough-fuzzy controller, *Fuzzy Sets and Systems*, **72** (1995) 61-63

[4] Hanisch, H.M., Luder, A.: A Signal Extension for Petri Nets and its Use in Controller Design, *Proc. of the Workshop on Concurrency, Specification and Programming*, September, 1998, Berlin

[5] Lin, T.Y., and Cercone, N. (Eds.): *Rough Sets and Data Mining. Analysis for Imprecise Data*, Kluwer Academic Publishers, Dordrecht, 1997

[6] Michalski, R., Carbonell, J.G., Mitchell, T.M. (Eds.): *Machine Learning: An Artificial Intelligence Approach,* Vol. 1, Tioga/Morgan Publishers, Los Altos, 1983

[7] Mrózek, A.: Rough Sets in Computer Implementation of Rule-Based Control of Industrial Processes, in: [28], pp. 19-31

[8] Mrózek, A., Płonka, L., Kędziera, J.: The methodology of rough controller synthesis, *Proc. of the 5th IEEE International Conference on Fuzzy Systems FUZZ-IEEE'96*, September 8-11, New Orleans, Louisiana (1996), pp. 1135-1139

[9] Munakata, T.: Rough control: Basic ideas and applications, in: P.P. Wang (Ed.), *Second Annual Joint Conference on Information Sciences (JCIS'95)*, September 28 - October 1, Wrightsville Beach, North Carolina, USA (1995), pp. 340-343

[10] Munakata, T.: *Fundamentals of the New Artifficial Intelligence*, Springer-Verlag, Berlin, 1998

[11] Murata, T.: Petri Nets: Properties, Analysis and Applications, in: *Proc. of the IEEE*, **77**-4 (1989), pp. 541-580

[12] Nadler, M., Smith, E.P.: *Pattern Recognition Engineering.* Wiley, New York, 1993.

[13] Nguyen, S.H., Skowron, A.: Quantization of real value attributes, in: *Proc. of the Second Joint Annual Conference on Information Sciences*, Wrightsville Beach, NC, September 28 - October 1, 1995, pp. 34-37

[14] Øhrn, A., Komorowski, J., Skowron, A., and Synak, P.: The Rosetta Software System, in: [23], pp. 572-575

[15] Pal, S.K., and Skowron, A. (Eds.): *Rough-Fuzzy Hybridization. A New Trend in Decision Making*, Springer-Verlag, Singapore, 1999.

[16] Pawlak, Z.: Decision Tables and Decision Algorithms, *Bulletin of the Polish Academy of Sciences*, **33**-9,10 (1985) 487-494

[17] Pawlak, Z.: *Rough sets - theoretical aspects of reasoning about data*, Kluwer Academic Publishers, Dordrecht, 1991

[18] Pawlak, Z., Munakata, T.: Rough Control: Application of rough set theory to control, in: *Proc. of the Fourth European Congress on Intelligent Techniques and Soft Computing (EUFIT'96)*, September 2-5, Germany, Verlag Mainz (1996), **1**, pp. 209-218

[19] Peters, J.F., Skowron, A., Suraj, Z., Pedrycz, W., and Ramanna, S.: Approximate Real-Time Decision Making: Concepts and Rough Fuzzy Petri Net Models, *International Journal of Intelligent Systems*, **14**-4 (1998) 4-37

[20] Peters, J.F., Ziaei, K., and Ramanna, S.: Approximate Time Rough Control: Concepts and Application to Satellite Attitude Control, in: L. Polkowski, and A. Skowron (Eds.), *Proc. of the Int. Conf. on Rough Sets and Current Trends in Computing (RSCTC'98)*, June, 1998, Warsaw, Poland, Lecture Notes in Artificial Intelligence, 1424, pp. 491-498

[21] Peterson, J.L.: *Petri Net Theory and the Modeling of Systems*, Prentice-Hall, Inc., Englewood Cliffs (1981), N.J.

[22] Polkowski, L., and Skowron, A. (Eds.): *Rough Sets in Knowledge Discovery 1. Methodology and Applications*, Physica-Verlag, Heidelberg, 1998

[23] Polkowski, L., and Skowron, A. (Eds.): *Rough Sets in Knowledge Discovery 2. Applications, Case Studies and Software Systems*, Physica-Verlag, Heidelberg, 1998

[24] Shapiro, S.C., and Eckroth, D.: *Encyclopaedia of Artificial Intelligence*, **1**, Wiley, 1987, pp. 18-35

[25] Skowron, A.: Extracting laws from decision tables: a rough set approach, *Computational Intelligence*, **11**/2 (1995) 371-388

[26] Skowron, A., and Rauszer, C.: The discernibility matrices and functions in information systems, in: [28], pp. 331-362

[27] Skowron, A., and Suraj, Z.: A Parallel Algorithm for Real-Time Decision Making: A Rough Set Approach, *Journal of Intelligent Information Systems*, **7**, Kluwer Academic Publishers, Dordrecht (1996) 5-28

[28] Słowiński, R. (Ed.), *Intelligent decision support: Handbook of applications and advances of the rough sets theory*, Kluwer Academic Publishers, Dordrecht, 1992

[29] Smith, D.R.: The synthesis of LISP programs from examples: a survey, in: A. Bierman, G. Guiho, and Y. Kodratoff (Eds.), *Automatic Program Construction Techniques*, Macmillan (1984), pp. 307-324

[30] Starke, P.H., Hanisch, H.M.: Analysis of Signal/Event-Nets, *Proc. of the 6th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA'97)*, Los Angeles, USA, September, 1997, pp. 253-257

[31] Suraj, Z.: PN-tools: Environment for the Design and Analysis of Petri Nets, *Control and Cybernetics*, **24**-2 (1995), Systems Research Institute of Polish Academy of Sciences 199-222

[32] Suraj, Z.: Reconstruction of Cooperative Information Systems under Cost Constraints: A Rough Set Approach, *Journal of Information Sciences*, **111** (1998), Elsevier, The Netherlands 273-291

[33] Suraj, Z.: Rough Set Methods for the Synthesis and Analysis of Concurrent Processes, *ICS PAS Reports 893* (1999), Institute of Computer Science of the Polish Academy of Sciences

[34] Wegener, I.: *The complexity of Boolean functions*, Wiley and B.G. Teubner, Stuttgart, 1987

[35] Zhou, MengChu, DiCesare, F.: *Petri Net Synthesis for Discrete Event Control of Manufacturing Systems*, Kluwer Academic Publishers, Dordrecht, 1993

[36] Ziarko, W.P.: Acquisition of control algorithms from operation data, in: [28], pp. 61-75