Proc. Estonian Acad. Sci. Phys. Math., 2005, **54**, 1, 26–45
https://doi.org/10.3176/phys.math.2005.1.02

Searching: models and methods

Volodymyr G. Skobelev

Institute of Applied Mathematics and Mechanics of National Academy of Sciences of Ukraine, Rose Luxemburg St. 74, Donetsk 83114, Ukraine; skbv@iamm.ac.donetsk.ua

Received 15 January 2004

Abstract. Searching in a partially ordered structure under the supposition that the set of elementary operators consists only of monotone ones is developed on the basis of schemes determined via some predicate. Models for the design of preset and adaptive solutions, namely an $\mathcal{M}$-model and an $\mathcal{AM}$-model, are determined and investigated. Interrelations between these models are established in the general case. Schemes for the design of solutions of all basic (minimal, irreducible, cooperative, and adaptive) types are developed. It is shown that the suggested approach can be applied to resolve some basic problems of discrete mathematics, such as the design of irreducible sets of representatives for a family of sets, design of experiments with finite automata, design of minterms and disjunctive normal forms consisting of minterms only.

Key words: searching schemes and models, preset and adaptive solutions, experiments with finite automata, disjunctive normal forms.

1. INTRODUCTION

On the basis of a set-theoretic (linguistic, in essence) approach *searching theory* was developed in terms of *operators* acting in some *space of situations* [1–3] (to avoid collisions, the term *situation* is used instead of the term *state*). The notion *of exhaustive searching* is applied to schemes designed under the suppositions that a set of situations is an abstract one (i.e. it is provided with no structure) and any operator is some (possibly, partial) mapping of this abstract set into itself. High complexity of exhaustive searching played the role of *the catalyst* for the design of more effective specific schemes based on the notion of *an estimator*, i.e. some *easily computable function* estimating *the distance* between a situation and *the target* (in essence, this means that a set of situations is provided with some *pseudo-metric space* structure). Unfortunately, all attempts to apply this approach to resolve some fundamental problems of *discrete mathematics*, such as finite automata identification, disjunctive normal forms design, etc., have been unsuccessful. The main reasons for this situation are the following. Firstly, the requirement for the existence of an estimator is too strong (as a consequence, only for a few problems estimators were designed). Secondly, there is no formal constructive definition of *an estimator* (as a consequence, problems of its existence and effective design remained also unresolved). Thus, there naturally arises the necessity to develop some approach for searching schemes design such that: 1) it is free of the above listed shortages; 2) it includes, as special cases, the set-theoretic approach as well as the one based on the notion of an estimator. In the present paper this problem is investigated via providing a set of situations with some *partial ordering* structure and by dealing with *monotone operators* only to implement the possibility *of comparing the distances* between any two situations and *the target* without estimating *the values* of these *distances* [4−9]. The power of the proposed approach is illustrated by its applications to resolve problems related to the identification of internal states of finite automata, design of irreducible sets of representatives for a family of sets, design of minterms as well as disjunctive normal forms consisting of minterms only [9−12]. Resolving the last problem provides us with powerful tools for controllability/observability analysis of combinational circuits (see [9,13]).

The paper is organized as follows. Section 2 introduces basic notions. In Section 3 schemes for the design of preset solutions, determined by some predicate, are developed. In Section 4 a model for preset solutions design, namely, an $\mathcal{M}$-model, is determined and problems of the design of minimal, irreducible, and cooperative solutions are resolved. In Section 5 a model for the design of adaptive solutions, namely, an $\mathcal{AM}$-model, is determined, its interrelations with the $\mathcal{M}$-model are investigated, and the problem of the design of some adaptive solution is resolved. Concluding remarks are given in Section 6.

2. BASIC NOTIONS

As a rule, in the role of *a model* intended for *preset solutions* design, some system $\mathcal{S} = (S, \mathcal{F}, s_{in}, S_{fin})$ is selected such that S is a finite set of *situations*, $\mathcal{F}$ is a finite set of *elementary operators*, i.e. (possibly, partial) mappings from S to S, s_{in} ($s_{in} \in S$) is *the initial situation*, and S_{fin} ($S_{fin} \subseteq S \backslash \{s_{in}\}$) is *the set of final situations*. The set of *operators* is *the free semigroup* $\mathcal{F}^*$ (i.e. $\mathcal{F}^* = \{\Lambda\} \cup F^+$, where Λ is *the empty string* and $F^+ = \bigcup_{i=1}^{\infty} F^i$) under the assumption that the action of any string $F \in \mathcal{F}^*$ on any situation $s \in S$ is determined by the following identities: $s\Lambda = s$, $sF = (\ldots(sf_1)\ldots)f_r$ ($F = f_1 \ldots f_r \in \mathcal{F}^+$). Let $\mathcal{F}_s^* = \{F \in \mathcal{F}^* | s \in Dom\ F\}$ ($s \in S$). The set $\mathcal{L}(\mathcal{S}) = \{F \in \mathcal{F}_{s_{in}}^* | s_{in}F \in S_{fin}\}$ will be called the set of *winning operators*. It is supposed that *the set of solutions* Ω ($\Omega \subseteq \mathcal{L}(\mathcal{S})$) is determined via some predicate $P : \mathcal{F}^* \to \{0, 1\}$ such that $(\forall F \in \mathcal{F}^*)(P(F) = 1 \iff F \in \Omega)$ (i.e. P is *the characteristic function* of a set Ω). We set $\mathcal{L}_P(\mathcal{S}) = \Omega$. Basic types of preset solutions are determined in the following way ($d(F)$ ($F \in \mathcal{F}^*$) denotes *the length* of a string F).

Definition 1. *For a system $\mathcal{S} = (S, \mathcal{F}, s_{in}, S_{fin})$:*

1. minimal *and* irreducible *solutions are determined, respectively, via the predicates P^{min} and P^{ir} such that*

$$P^{min}(F) = 1 \Longleftrightarrow (F \in \mathcal{L}(\mathcal{S}))$$
$$\&(\forall F_1 \in F^+)((d(F_1) < d(F)) \vee (d(F_1) > d(F)) \Longrightarrow P^{min}(F_1) = 0),$$

$$P^{ir}(F) = 1 \Longleftrightarrow (F \in \mathcal{L}(\mathcal{S}))\&(\forall F_1 \in erase(F))(P^{ir}(F_1) = 0),$$

where $erase(F)$ is the set of all operators obtained by deleting of at least one letter in F;

2. *k-cooperative solutions ($k \geq 2$) are determined via the predicate P^{cp}: $\underbrace{\mathcal{F}^* \times \cdots \times \mathcal{F}^*}_{k\ times} \to \{0,1\}$ such that*

$$P^{cp}(F_1, \ldots, F_k) = 1 \Longleftrightarrow (\omega_k(s_{in}F_1, \ldots, s_{in}F_k) \in S_{fin})$$

$$\&(\forall F_1' \in erase(F_1)) \ldots (\forall F_k' \in erase(F_k))(\omega_k(s_{in}F_1', \ldots, s_{in}F_k') \notin S_{fin}),$$

where $\omega_k : \underbrace{S \times \cdots \times S}_{k\ times} \to S$ is some fixed operation.

Remark. If *k-cooperative* solutions are designed, it is usually supposed that an operation ω_k and a predicate P^{cp} are *symmetric*, i.e. $\omega_k(s_1, \ldots, s_k) = \omega_k(s_{\alpha(1)}, \ldots, s_{\alpha(k)})$ and $P^{cp}(F_1, \ldots, F_k) = P^{cp}(F_{\alpha(1)}, \ldots, F_{\alpha(k)})$ for any permutation α of the set $\{1, \ldots, k\}$. It is worth noting that these conditions are not necessary ones.

As a rule, in the role of a model intended for *adaptive solutions* design, some system $\mathcal{S} = (S, \mathcal{H}, s_{in}, S_{fin})$ ($\mathcal{H} \subseteq \mathcal{F} \times \mathcal{G}$) is selected. An elementary operator $(f, g) \in \mathcal{H}$ is interpreted as follows: f ($f \in \mathcal{F}$) is an elementary operator in the usual sense of this word, while g ($g \in \mathcal{G}$) is the name of some specific case, determined as a result of some additional analysis of the situation that takes place, i.e. g represents some analogy with the CASE operator of a programming language. Let $\mathcal{G}_{s,f} = \{g \in \mathcal{G} | (f, g) \in \mathcal{H}, s \in Dom\ (f, g)\}$.

Definition 2. *For a system $\mathcal{S} = (S, \mathcal{H}, s_{in}, S_{fin})$ ($\mathcal{H} \subseteq \mathcal{F} \times \mathcal{G}$)* an adaptive solution *is any partial mapping $B : \mathcal{G}^* \to \mathcal{F}$ such that*: 1) $\Lambda \in Dom\, B$; 2) $G(r) = g_1 \ldots g_r \in Dom\, B$ $(r = 1, 2, \ldots)$ *if and only if* $G(r-1) \in Dom\ B$, $s_{in} \in Dom\ H(r)$, $s_{in}H(r) \notin S_{fin}$, *where* $H(r) = (B(\Lambda), g_1)(B(g_1), g_2) \ldots (B(g_1 \ldots g_{r-1}), g_r)$; 3) *if $G(r)g \notin Dom\ B$ for all $g \in \mathcal{G}$, then $\mathcal{G}_{s_nH(r),B(G(r))} \neq \emptyset$ and $s_{in}H(r)(B(G(r)), g') \in S_{fin}$ for all $g' \in \mathcal{G}_{s_nH(r),B(G(r))}$*; 4) *there exists $l_{\mathcal{S}} \in \mathbb{N}$ such that $d(G) \leq l_{\mathcal{S}}$ for all $G \in Dom\ B$.*

Any *searching scheme* intended for preset solutions design is reduced to explicit design of some *acceptor*, which presents the set of solutions, as *a language*. It is worth noting that this acceptor is designed by extracting a special *subtree* of (possibly, infinite) *tree*, presented implicitly. This tree can be determined inductively in the following way.

Definition 3. *For a system $\mathcal{S} = (S, \mathcal{F}, s_{in}, S_{fin})$ the rooted oriented labelled tree $\mathcal{D}_{\mathcal{S}}$ is such that*: 1) *the root is labelled by the element s_{in}*; 2) *an arc labelled by an element $f \in \mathcal{F}$ starts from a vertex labelled by an element $s \in S$ if and only if $s \in Dom\ f$, and this arc terminates in a vertex labelled by an element sf*; 3) *different arcs that started from the same vertex are labelled by different elements of the set $\mathcal{F}$.*

If the labels of arcs of some path which started in the root of the tree $\mathcal{D}_{\mathcal{S}}$ form the string $F = f_1 \ldots f_r$ $(r = 0, 1, \ldots)$, then the terminal vertex of this path is denoted by v_F. Thus, the root of the tree $\mathcal{D}_{\mathcal{S}}$ is denoted by v_Λ. The vertices of the tree $\mathcal{D}_{\mathcal{S}}$ are placed into sequential levels enumerated by nonnegative integers, i.e. the ith level $(i = 0, 1, \ldots)$ consists of all vertices v_F such that $F \in \mathcal{F}^*_{s_{in}}$ and $d(F) = i$.

Similarly, the following *tree* is determined inductively if adaptive solutions are designed.

Definition 4. *For a system $\mathcal{S} = (S, \mathcal{H}, s_{in}, S_{fin})$ $(\mathcal{H} \subseteq \mathcal{F} \times \mathcal{G})$ the rooted oriented labelled tree $\mathcal{D}_{\mathcal{S}}$ is such that*: 1) *the root is labelled by the element s_{in} and is placed into the 0th level*; 2) *different arcs that started from any vertex are labelled by different elements*; 3) *let v be a vertex labelled by an element s $(s \in S)$ and placed into the $2i$th level $(i = 0, 1, \ldots)$. An arc labelled by an element f $(f \in \mathcal{F})$ starts from the vertex v if and only if $\mathcal{G}_{s,f} \neq \emptyset$. This arc terminates in an unlabelled vertex $v'_{v,f}$ placed into the $(2i+1)$th level. An arc labelled by an element g $(g \in \mathcal{G})$ starts from the vertex $v'_{v,f}$ if and only if $g \in \mathcal{G}_{s,f}$. This arc terminates in a vertex labelled by the element $s(f, g)$ and placed into the $2(i+1)$th level.*

Remark. It is evident that the tree $\mathcal{D}_{\mathcal{S}}$ determines some *one-person game* if preset solutions are designed (see Definition 3) and some *two-persons game* if adaptive solutions are designed (see Definition 4). Thus, any searching scheme is reduced to the design of some *winning strategy* for the corresponding class of games.

3. SCHEMES FOR $\mathcal{L}_P(\mathcal{S})$ DESIGN

Let $\mathcal{D}(P, \mathcal{S})$ be the minimal (by the number of vertices) subtree of the tree $\mathcal{D}_{\mathcal{S}}$ consisting of the root and the set of vertices $V_{fin} = \{v_F | F \in \mathcal{L}_P(\mathcal{S})\}$. The set $\mathcal{L}_P(\mathcal{S})$ is the language determined by the acceptor $(\mathcal{D}(P, \mathcal{S}), v_\Lambda, V_{fin})$. Any searching scheme for the set $\mathcal{L}_P(\mathcal{S})$ is determined by the mode in which a sequence of trees

$$\mathcal{D}(0), \mathcal{D}(1), \ldots \tag{1}$$

is designed such that: 1) the root of the tree $\mathcal{D}_{\mathcal{S}}$ is the root of every tree $\mathcal{D}(i)$ $(i = 1, 2, \dots)$; 2) $\mathcal{D}(P, \mathcal{S}) \subseteq \mathcal{D}(0) \cup \mathcal{D}(1) \cup \cdots \subseteq \mathcal{D}_{\mathcal{S}}$; 3) any tree $\mathcal{D}(i)$ $(i = 1, 2, \dots)$ is uniquely determined by the sequence $\mathcal{D}(0), \dots, \mathcal{D}(i-1)$; 4) any tree $\mathcal{D}(i)$ $(i = 1, 2, \dots)$ consists of *as few vertices as possible*; 5) if $\mathcal{D}(P, \mathcal{S})$ is a finite tree, then the sequence (1) is also finite; 6) if the sequence (1) is infinite, then

$$\lim_{h\to\infty} \frac{|V((\bigcup_{i=1}^{\infty} \mathcal{D}(i))[h])|}{|V(\mathcal{D}(P, \mathcal{S})[h])|} = 1,$$

where $\mathcal{D}[h]$ $(h = 0, 1, \dots)$ is the maximal subtree of the height h of the tree $\mathcal{D}$ and $V(\mathcal{D})$ is the set of vertices of the tree $\mathcal{D}$.

Remark. The word combination "*a tree $\mathcal{D}$ consists of as few vertices as possible*" is used in the following sense: "*in the process of the design of considered solutions there exist no cutting rules for a tree $\mathcal{D}_{\mathcal{S}}$ producing a proper subtree of the tree $\mathcal{D}$*".

In what follows, it is supposed that the set $\mathcal{L}_P(\mathcal{S})$ is *a regular* one. As a rule, this claim is true if any searching is applied. Moreover, it provides us with a sufficient condition for eliminating any problem connected with *the halting problem,* as well as with a possibility of converting the tree $\mathcal{D}(P, \mathcal{S})$ into some *finite tree* $\mathcal{D}_{\mathcal{S}}^{P}$. The latter tree, in its turn, can be converted into some finite acceptor, which determines the language $\mathcal{L}_P(\mathcal{S})$. Taking the above supposition into account, it is assumed in what follows that the sequence (1) of trees is finite.

Remark. As a rule, the height $L_{\mathcal{D}_{\mathcal{S}}^{P}}$ of the tree $\mathcal{D}_{\mathcal{S}}^{P}$ is not known in advance and is determined only in the process of the design of the tree $\mathcal{D}_{\mathcal{S}}^{P}$ in the explicit form.

In the remainder of the Section three basic schemes for the design of the set $\mathcal{L}_P(\mathcal{S})$ are developed.

3.1. Breadth-first searching

Sequential design of the tree $\mathcal{D}_{\mathcal{S}}^{P}$, one level after another, leads to the design of the finite sequence of trees $\mathcal{D}(0), \mathcal{D}(1), \dots, \mathcal{D}(k)$ such that: 1) the root of the tree $\mathcal{D}_{\mathcal{S}}$ is the root of every tree $\mathcal{D}(i)$ $(i = 0, 1, \dots, k)$; 2) the height of the tree $\mathcal{D}(i)$ is equal to i for all $i = 0, 1, \dots, k-1$, and the height of the tree $D(k)$ is equal to $L_{\mathcal{D}_{\mathcal{S}}^{P}}$; 3) $\mathcal{D}_{\mathcal{S}}^{P}[i] \subseteq \mathcal{D}(i) \subseteq \mathcal{D}_{\mathcal{S}}[i]$ $(i = 0, 1, \dots, k)$; 4) any tree $\mathcal{D}(i)$ $(i = 1, \dots, k)$ is uniquely determined by the sequence $\mathcal{D}(0), \dots, \mathcal{D}(i-1)$; 5) any tree $\mathcal{D}(i)$ $(i = 1, \dots, k)$ consists of as few vertices as possible.

Let V_i $(i = 1, \dots, k)$ be the set of all vertices placed into the ith level of the tree $\mathcal{D}(i)$.

Definition 5. *A vertex $v_F \in V_i$ is considered to be*: 1) a final *one if $F \in \mathcal{L}_P(\mathcal{S})$*; 2) an unprofitable *one if it is wittingly known that F is not a prefix of any element of the set $\mathcal{L}_P(\mathcal{S})$*; 3) a generating *one if v_F generates at least one vertex placed in the $(i+1)$th level of the tree $\mathcal{D}(i+1)$*.

Let $fnl(i)$, $unp(i)$, $gnr(i)$ $(i = 0, 1, \dots, k)$ be the sets of all, respectively, final, unprofitable, and generating vertices of the tree $\mathcal{D}(i)$. To determine these sets in the explicit form, it is necessary to investigate the structure of the set $\mathcal{L}_P(\mathcal{S})$. As a rule, this is done via verification of the validity of propositions of the following type (the symbol ### is placed instead of the word combination "*the following conditions: ... hold*").

Proposition 1. *A vertex v_F placed into the ith level $(i = 1, 2, \dots)$ of the tree $\mathcal{D}_\mathcal{S}$ is a final one if and only if $s_{in}F \in S_{fin}$ and there exists no vertex $v_{F'}$ of the tree $\mathcal{D}_\mathcal{S}[i]$ such that ###.*

Proposition 2. *A vertex v_F placed into the ith level $(i = 1, 2, \dots)$ of the tree $\mathcal{D}_\mathcal{S}$ is an unprofitable one if and only if ### and there exists a vertex $v_{F'}$ of the tree $\mathcal{D}_\mathcal{S}[i]$ such that ###.*

Proposition 3. *The set $gnr(i)$ $(i = 1, 2, \dots)$ is any minimal by cardinality subset of vertices located in the ith level of the tree $\mathcal{D}_\mathcal{S}[i]$ and such that ###.*

It is evident that the set $gnr(i)$ $(i = 0, 1, \dots)$ is determined uniquely if and only if either $gnr(i) = V_i \backslash unp(i)$, or $gnr(i) = V_i \backslash (unp(i) \cup fnl(i))$. Otherwise, it is necessary to search some set $gnr(i)$. The complexity of this searching depends on the cardinality of the sets $V_i \backslash unp(i)$ and $V_i \backslash (unp(i) \cup fnl(i))$, respectively. Besides, it is necessary to verify the validity of the following proposition.

Proposition 4. *Any set $gnr(i)$ $(i = 0, 1, \dots)$ generates some set $gnr(i + 1)$.*

The sets $fnl(i)$, $unp(i)$, $gnr(i)$ $(i = 0, 1, \dots, L_{\mathcal{D}^P_\mathcal{S}})$ form the base for the design of the set $\mathcal{L}_P(\mathcal{S})$ via breadth-first searching. To implement the characteristic "*any tree $\mathcal{D}(i)$ $(i = 1, \dots, L_{\mathcal{D}^P_\mathcal{S}})$ consists of as few vertices as possible*", the following procedure for *garbage deleting* would be applied.

Procedure $GRBG(D(i))$.

Step 1. Provide every vertex $v \in V_i \backslash (gnr(i) \cup fnl(i))$ with the mark ♠, $j := i - 1$.

Step 2. If $j = 0$, then go to Step 5, else go to Step 3.

Step 3. Provide with the mark ♠ every vertex v, placed into the jth level, such that all arcs starting in v terminate in vertices of the $(j + 1)$th level, each provided with the mark ♠.

Step 4. If there exists a vertex v placed into the jth level and provided with the mark ♠, then $j := j - 1$ and go to Step 2, else go to Step 5.

Step 5. Delete all vertices provided with the mark ♠ (and all arcs incidental to these vertices) and HALT.

The validity of the procedure $GRBG$ is evident. Thus, the following algorithm for the set $\mathcal{L}_P(\mathcal{S})$ design via breadth-first searching can be proposed.

Algorithm 1.

Step 1. $\mathcal{D}(0) := \mathcal{D}_{\mathcal{S}}[0]$, $\mathcal{D} := \mathcal{D}_{\mathcal{S}}[1]$, $i := 1$.

Step 2. $\mathcal{D} := GRBG(\mathcal{D})$, $\mathcal{D}(i) := \mathcal{D}$.

Step 3. If $gnr(i) = \emptyset$, then HALT, else go to Step 4.

Step 4. Set $\mathcal{D}$ to be the tree obtained by insertion into $\mathcal{D}(i)$ of the part of the $(i+1)$th level of the tree $\mathcal{D}_{\mathcal{S}}$, generated by the set $gnr(i)$.

Step 5. $i := i + 1$ and go to Step 2.

Theorem 1. *If the sequence $\mathcal{D}(0), \mathcal{D}(1), \ldots, \mathcal{D}(k)$ is designed by Algorithm* 1, *then $\mathcal{D}(k) = \mathcal{D}_{\mathcal{S}}^{P}$.*

Proof. As a result of the execution of the procedure $GRBG$ no element of the set V_{fin} is deleted. Thus, it is sufficient to verify that any leaf placed into the ith level $(i \geq 1)$ of the tree $\mathcal{D}(k)$ (if such a leaf exists) is an element of the set V_{fin}.

Algorithm 1 terminates only in Step 3. Transition to Step 3 takes place only if the procedure $GRBG$ is executed in Step 2. Since $gnr(k) = \emptyset$, as a result of the kth execution of the procedure $GRBG$ any vertex $v \in V_k \backslash fnl(k)$ is provided with the mark ♠ and is deleted. Thus, $V_k \subseteq V_{fin}$.

Let v_F be any leaf placed into the ith level $(i = 1, \ldots, k-1)$ of the tree $\mathcal{D}(k)$. Suppose that $v_F \notin V_{fin}$. As a result of the ith execution of the procedure $GRBG$ all elements of the set $unp(i)$ are deleted. Thus, $s_{in}F \in Dom\ f$, for at least one $f \in \mathcal{F}$. This implies that there exists some $j \in \{i, \ldots, k\}$ such that in the process of the jth execution of the procedure $GRBG$ all descendants of the vertex v_F are provided with the mark ♠. For this reason, in the process of the jth execution of the procedure $GRBG$ the vertex v_F is also provided with the mark ♠ and is deleted, i.e. there is no vertex v_F in the tree $\mathcal{D}(k)$. Contradiction. Thus, the supposition is false, i.e. $v_F \in V_{fin}$. □

3.2. Backtracking

Informally speaking, some routing via some minimal subtree $\mathcal{D}$ is executed, such that

$$\mathcal{D}_{\mathcal{S}}^{P} \subseteq \mathcal{D} \subseteq \mathcal{D}_{\mathcal{S}}[k], \tag{2}$$

in accordance with the following two rules: (a) move forward, while it is possible; (b) if forward move is impossible, then annul the last action, return to the previous situation, and then apply rule (a).

Remark. It is evident that any routing via the tree $\mathcal{D}$ is based on the supposition that the set $\mathcal{F}$ is a linearly-ordered one.

Let $lbl(v)$ be the label of a vertex v, $lvl(v)$ be the number of the level into which the vertex v is placed, and $str(v_\Lambda, v)$ be the string formed by the labels of arcs of the path leading from the root of the tree $\mathcal{D}_{\mathcal{S}}$ to the vertex v. Using the sets of vertices $fnl(i)$, $unp(i)$, $gnr(i)$ $(i = 0, 1, \ldots, k)$, the following backtracking algorithm can be proposed for the design of the set $\mathcal{L}_P(\mathcal{S})$ by executing routing via some subtree of the tree $\bigcup_{i=1}^{k} \mathcal{D}(i)$.

Algorithm 2.

Step 1. $v^* := v_\Lambda$, $\mathcal{L} := \emptyset$.

Step 2. $vrnts(v^*) := \{f \in \mathcal{F} | lbl(v^*) \in Dom\ f\}$.

Step 3. If $vrnts(v^*) \neq \emptyset$, then go to Step 4, else go to Step 11.

Step 4. Set f the first element of the set $vrnts(v^*)$, $vrnts(v^*) := vrnts(v^*) \backslash \{f\}$.

Step 5. Insert a vertex v into the $(lvl(v^*) + 1)$th level. Insert an arc that starts in the vertex v^* and terminates in the vertex v. Label the inserted vertex and the inserted arc, respectively, by the situation $lbl(v^*)f$ and by the elementary operator f.

Step 6. If $v \notin gnr(lvl(v^*) + 1) \cup fnl(lvl(v^*) + 1)$, then go to Step 7, else go to Step 8.

Step 7. Delete the vertex v and the arc that terminates in it and go to Step 3.

Step 8. $v^* := v$ and go to Step 9.

Step 9. If $v^* \in fnl(lvl(v^*) + 1)$, then $\mathcal{L} := \mathcal{L} \cup \{str(v_\Lambda, v^*)\}$.

Step 10. Go to Step 2.

Step 11. If $v^* = v_\Lambda$, then HALT, else go to Step 12.

Step 12. If $v^* = v_{Ff}$ $(F \in \mathcal{F}^*, f \in \mathcal{F})$, then $vrnts(v_F) := vrnts(v_F) \backslash \{f\}$, $v^* := v_F$, delete the vertex v_{Ff} and the arc that terminates in it, and go to Step 3.

Theorem 2. *If the set* $\mathcal{L}$ *is designed by Algorithm* 2, *then* $\mathcal{L} = \mathcal{L}_P(\mathcal{S})$.

Proof. Steps 1 and 9 imply that $\mathcal{L} \subseteq \mathcal{L}_P(\mathcal{S})$. Steps 1–8, 11, and 12 imply that there is implemented some routing via some subtree $\mathcal{D} \subseteq \bigcup_{i=1}^{k} \mathcal{D}(i)$ satisfying the inclusions (2). Thus, any operator $F \in \mathcal{L}_P(\mathcal{S})$ is represented in the tree $\mathcal{D}$ via the labels of arcs of some path which started in the root of the tree $\mathcal{D}$. Taking Steps 1 and 9 into account, we get $\mathcal{L} \supseteq \mathcal{L}_P(\mathcal{S})$. Inclusions $\mathcal{L} \subseteq \mathcal{L}_P(\mathcal{S})$ and $\mathcal{L} \supseteq \mathcal{L}_P(\mathcal{S})$ imply that the identity $\mathcal{L} = \mathcal{L}_P(\mathcal{S})$ holds. $\square$

3.3. Branch-and-bound method

Informally speaking, searching for *the cheapest solutions* can be reduced to the design of some minimal subtree $\mathcal{D}$ of the tree $\bigcup_{i=1}^{k} \mathcal{D}(i)$ in accordance with the following rule: at any step there are generated and analysed direct descendants of *the cheapest vertex*, i.e. some vertex v, such that *the cost* of the string $str(v_\Lambda, v)$ is the lowest.

Let some *estimator* $cst : V(\mathcal{D}_\mathcal{S}) \rightarrow \mathbb{R}_+$ be fixed, satisfying the inequality $cst(Ff) \geq cst(F)$ $(fF \in \mathcal{F}^+_{s_{in}})$. Since there is a one-to-one correspondence between the set $\mathcal{F}^*_{s_{in}}$ and the set of all vertices of the tree $\mathcal{D}_\mathcal{S}$, we set $cost(v_F) = cost(F)$, for all $F \in \mathcal{F}^*_{s_{in}}$. It is supposed that the target is the set of solutions $\mathcal{L}_P(\mathcal{S})$ such that

$$\mathcal{L}_P(\mathcal{S}) \subseteq \{F \in \mathcal{L}_{P_1}(\mathcal{S}) | cst(v_F) = \min_{F' \in \mathcal{L}_{P_1}(\mathcal{S})} cst(v_{F'})\},$$

where P_1 is some given predicate. Let lst be the list of all leaves of the designed subtree, ordered in accordance with the values of the estimator cst, and lub be the least upper bound of previously computed values of the estimator cst. Using the sets of vertices $fnl(i)$, $unp(i)$, $gnr(i)$ $(i = 0, 1, \ldots, k)$, the following algorithm for the set $\mathcal{L}_P(\mathcal{S})$ design via the branch-and-bound method can be proposed.

Algorithm 3.

Step 1. $lub := \infty$, $lst := \{v_\Lambda\}$, $\mathcal{L} := \emptyset$.

Step 2. If $lst = \emptyset$, then HALT, else go to Step 3.

Step 3. Set v^* the first element of the list lst, $lst := lst \backslash \{v^*\}$.

Step 4. $vrnts(v^*) := \{f \in \mathcal{F} | lbl(v^*) \in Dom\ f\}$.

Step 5. If $vrnts(v^*) \neq \emptyset$, then go to Step 6, else go to Step 2.

Step 6. Set f the first element of the set $vrnts(v^*)$, $vrnts(v^*) := vrnts(v^*) \backslash \{f\}$.

Step 7. Insert a vertex v into the $(lvl(v^*) + 1)$th level. Insert an arc that starts in the vertex v^* and terminates in the vertex v. Label the inserted vertex and the inserted arc, respectively, by the situation $lbl(v^*)f$ and by the elementary operator f.

Step 8. If $v \notin gnr_{P_1}(lvl(v^*) + 1) \cup fnl_{P_1}(lvl(v^*) + 1)$, then go to Step 13, else go to Step 9.

Step 9. If $cst(v) \leq lub$, then go to Step 10, else go to Step 13.

Step 10. If $v \in V_{fin}$, then go to Step 11, else go Step 12.

Step 11. If $cst(v) = lub$, then $\mathcal{L} := \mathcal{L} \cup \{str(v_\Lambda, v)\}$, else $lub := cst(v)$ and $\mathcal{L} := \{str(v_\Lambda, v)\}$.

Step 12. $lst := lst \cup \{v\}$ and go to Step 5.

Step 13. Delete the vertex v_{Ff} and the arc that terminates in it, and go to Step 5.

Theorem 3. *If the set $\mathcal{L}$ is designed by Algorithm* 3, *then $\mathcal{L} = \mathcal{L}_P(\mathcal{S})$.*

Proof. The cutting rules used in Algorithm 3 are stronger than the ones used in Algorithm 2. This reinforcement results in deleting any vertex v (and all arcs connected to it), whose cost exceeds the cost of previously designed elements of the set $\mathcal{L}_{P_1}(\mathcal{S})$. As a result of the execution of Steps 9–11 the set $\mathcal{L}$ would consist only of the cheapest previously designed elements of the set $\mathcal{L}_{P_1}(\mathcal{S})$. Since the estimator cst is a nondecreasing one, $F \notin \mathcal{L}_P(\mathcal{S})$ for any operator F, formed by the labels of arcs of any path which started in the root and passed through the vertex v. This fact and the validity of Algorithm 2 imply that any operator $F \in \mathcal{L}_P(\mathcal{S})$ is represented in the subtree $\mathcal{D}$ designed by Algorithm 3 via the labels of arcs of some path which started in the root of the tree $\mathcal{D}$. Thus, $\mathcal{L} = \mathcal{L}_P(\mathcal{S})$. $\square$

The number of vertices of the tree $\mathcal{D}$ designed by any of Algorithms 1–3 is estimated as $O(|\mathcal{F}|^k)$ $(k \to \infty)$. Thus, the time complexity of each of these algorithms is some exponent. Space of exponential size is needed to store the explicit form of the tree $\mathcal{D}$. This storage is necessary for Algorithms 1 and 3, only. Thus, the space complexity of Algorithms 1 and 3, each, is some exponent. As to Algorithm 2, it is necessary to store the single analysed path $s_{in}f_1s_1\ldots s_{i-1}f_is_i$ $(i = 0, 1, \ldots, k)$. Thus, the space complexity of Algorithm 2 is $O(k)$ $(k \to \infty)$. This estimation may be reinforced in the following important special case.

Theorem 4. *If the set $\mathcal{F}$ generates a commutative semigroup, then space complexity of Algorithm* 2 *is $O(|\mathcal{F}|)$ $(k \to \infty)$.*

Proof. Since $f_if_j = f_jf_i$ $(f_i, f_j \in \mathcal{F})$, any operator $F \in \mathcal{F}^*$ can be rewritten in the form $F = f_1^{\alpha_1}\ldots f_m^{\alpha_m}$ $(\alpha_i \in \mathsf{Z}_+ (i = 1, \ldots, m))$. This form is presented completely via the m-tuples $(\alpha_1, \ldots, \alpha_m)$ and $(\xi_1, \ldots, \xi_m)$, where ξ_i $(i = 1, \ldots, m)$ is *a flag* (i.e. *a Boolean variable*) such that $\xi_i = 1$ if and only if the analysis of the operator $f_i^{\alpha_i}$ is completed. Space of the size $O(|\mathcal{F}|)$ $(k \to \infty)$ is sufficient to store the above m-tuples. $\square$

4. DESIGN OF PRESET SOLUTIONS

In [1,2] the structure of the winning operators set is characterized, in essence, in terms of situations *equidistant* from the set of final situations. Thus, it looks attractive to determine some partial ordering on the set of situations, consistent with the sets of *equidistant situations*. This approach leads to the following class of structured models.

Definition 6. *An $\mathcal{M}$-*model *is a system $\mathcal{S} = (S, \mathcal{F}, s_{in}, S_{fin})$ such that on the set S some partial ordering $\leq_S$ is determined such that*: 1) *if $s_1 \in S_{fin}$ and $s_2 \leq_S s_1$, then $s_2 \in S_{fin}$*; 2) *if $s_1 \in Dom\ f$ $(f \in \mathcal{F})$ and $s_2 \leq_S s_1$, then $s_2 \in Dom\ f$*; 3) *if $s_2 \leq_S s_1$, then $s_2 f \leq_S s_1 f$ $(f \in \mathcal{F})$.*

Lemma 1. *If $s_2 \leq_S s_1$, then*: 1) $\mathcal{F}^*_{s_2} \supseteq \mathcal{F}^*_{s_1}$; 2) $s_2 F \leq_S s_1 F$ *for all $F \in \mathcal{F}^*_{s_1}$.*

Proof. By induction over the length of an operator.

It seems that for a fixed $\mathcal{M}$-model $\mathcal{S} = (S, \mathcal{F}, s_{in}, S_{fin})$ the following problems are the basic ones (since a great number of problems of discrete mathematics as well as of its numerous applications are reduced to these).

Problem 1. It is necessary to design the set $\mathcal{L}^{min}(\mathcal{S})$ of all minimal solutions for an $\mathcal{M}$-model $\mathcal{S}$.

Problem 2. It is necessary to design one (no matter which) minimal solution for an $\mathcal{M}$-model $\mathcal{S}$.

Problem 3. It is necessary to design the set $\mathcal{L}^{ir}(\mathcal{S})$ of all irreducible solutions for an $\mathcal{M}$-model $\mathcal{S}$.

It is evident that to resolve Problem 2 it is sufficient to resolve Problem 1. Unfortunately, this approach complicates resolving Problem 2 excessively. For this reason it is a matter of principle to extract some *kernel* $\mathcal{L}^{min}_{krnl}(\mathcal{S})$ of the set $\mathcal{L}^{min}(\mathcal{S})$ such that: 1) $\mathcal{L}^{min}_{krnl}(\mathcal{S}) \subseteq \mathcal{L}^{min}(\mathcal{S})$; 2) the design of a set $\mathcal{L}^{min}_{krnl}(\mathcal{S})$ is as simple as possible; 3) $\mathcal{L}^{min}_{krnl}(\mathcal{S}) \neq \emptyset$ if and only if $\mathcal{L}^{min}(\mathcal{S}) \neq \emptyset$. Thus, Problem 2 is naturally reduced to the following

Problem 4. It is necessary to design some kernel $\mathcal{L}^{min}_{krnl}(\mathcal{S})$ of the set $\mathcal{L}^{min}(\mathcal{S})$ for an $\mathcal{M}$-model $\mathcal{S}$.

To resolve Problems 1, 3, and 4, it is sufficient to determine the corresponding sets of final, unprofitable, and generating vertices and then apply either Algorithm 1 or Algorithm 2.

Theorem 5. *If $s_{in}F_1 \leq_S s_{in}F_2$ and $d(F_1) < d(F_2)$, then $F_2 F \notin \mathcal{L}^{min}(\mathcal{S})$ for all $F \in \mathcal{F}^*_{s_{in}F_2}$.*

Proof. Suppose that there exists some operator $F \in \mathcal{F}^*_{s_{in}F_2}$ such that $F_2 F \in \mathcal{L}^{min}(\mathcal{S})$. Since $s_{in}F_1 \leq_S s_{in}F_2$ and $F \in \mathcal{F}^*_{s_{in}F_2}$, it follows that (see Lemma 1) $F \in \mathcal{F}^*_{s_{in}F_1}$ and $(s_{in}F_1)F \leq_S (s_{in}F_2)F$, i.e. $s_{in}(F_1 F) \leq_S s_{in}(F_2 F)$. Since $F_2 F \in \mathcal{L}^{min}(\mathcal{S})$, it follows that $s_{in}(F_2 F) \in S_{fin}$. Since $s_{in}(F_2 F) \in S_{fin}$ and $s_{in}(F_1 F) \leq_S s_{in}(F_2 F)$, it follows that (see Definition 6) $s_{in}(F_1 F) \in S_{fin}$, i.e. $F_1 F \in \mathcal{L}(\mathcal{S})$. Since $d(F_1 F) = d(F_1) + d(F) < d(F_2) + d(F) = d(F_2 F)$, it follows that $F_1 F \in \mathcal{L}(\mathcal{S})$, $F_2 F \in \mathcal{L}^{min}(\mathcal{S})$ and $d(F_1 F) < d(F_2 F)$. Contradiction. Thus, the supposition is false, i.e. $F_2 F \notin \mathcal{L}^{min}(\mathcal{S})$. $\square$

Definition 7. An i-characteristic set V_i^{chr} $(i = 0, 1, \ldots)$ *is any minimal by cardinality subset of the set V_i satisfying the following condition*: *for every vertex $v_F \in V_i$ there exists some vertex $v_{F'} \in V_i^{chr}$ such that $s_{in}F' \leq_S s_{in}F$.*

Lemma 2. *Any i-characteristic set $(i = 0, 1, \dots)$ generates some $(i + 1)$-characteristic set.*

Proof. By induction over the length of an operator.

Theorem 6. *If an operator $F \in \mathcal{F}^+_{s_{in}}$ can be presented in the form $F = F_1F_2$ $(F_1, F_2 \in \mathcal{F}^+)$ such that $s_{in}F_1 \leq_S s_{in}F$, then $FF_3 \notin \mathcal{L}^{ir}(\mathcal{S})$ for all $F_3 \in \mathcal{F}^+_{s_{in}F}$.*

Proof. Suppose that there exists some operator $F_3 \in \mathcal{F}^*_{s_{in}F}$ such that $FF_3 \in \mathcal{L}^{ir}(\mathcal{S})$. Since $s_{in}F_1 \leq_S s_{in}F$ and $F_3 \in \mathcal{F}^*_{s_{in}F}$, it follows that $F_3 \in \mathcal{F}^*_{s_{in}F_1}$ and $(s_{in}F_1)F_3 \leq_S (s_{in}F)F_3$, i.e. $s_{in}(F_1F_3) \leq_S s_{in}(FF_3)$. Since $FF_3 \in \mathcal{L}^{ir}(\mathcal{S})$, it follows that $s_{in}(FF_3) \in S_{fin}$. Since $s_{in}(F_1F_3) \leq_S s_{in}(FF_3)$ and $s_{in}(FF_3) \in S_{fin}$, it follows that $s_{in}(F_1F_3) \in S_{fin}$, i.e. $F_1F_3 \in \mathcal{L}(\mathcal{S})$. Thus, we get $F_1F_2F_3 \in \mathcal{L}^{ir}(\mathcal{S})$, $F_1F_3 \in \mathcal{L}(\mathcal{S})$ and $F_2 \in \mathcal{F}^+$. Contradiction. Thus, the supposition is false, i.e. $FF_3 \notin \mathcal{L}^{ir}(\mathcal{S})$. □

Let the sets $\mathcal{L}^{min}(\mathcal{S})$, $\mathcal{L}^{min}_{krnl}(\mathcal{S})$, and $\mathcal{L}^{ir}(\mathcal{S})$ be determined via predicates P^{min}, P^{min}_{krnl}, and P^{ir}. These sets are finite. Thus, $\mathcal{D}^P_{\mathcal{S}} = \mathcal{D}(P, \mathcal{S})$ $(P \in \{P^{min}, P^{min}_{krnl}, P^{ir}\})$. To resolve Problems 1, 3, and 4, the sets of final, unprofitable, and generating vertices would be determined in the following way: for all $i = 0, 1, \dots,$

$$fnl_P(i) = \{v_F \in V_i | (d(F) = i) \& (s_{in}F \in S_{fin})\} \quad (P \in \{P^{min}, P^{min}_{krnl}, P^{ir}\}),$$

$$unp_P(i) = \{v_F \in V_i | A \vee (\exists v_{F_1})((d(F_1) < d(F)) \& (s_{in}F_1 \leq_S s_{in}F))\}$$
$$(P \in \{P^{min}, P^{min}_{krnl}\}),$$

$$unp_{P^{ir}}(i)\{v_F \in V_i | A \vee (\exists v_{F_1})((d(F_1) < d(F)) \& (\exists F_2 \in \mathcal{F}^+)(F = F_1F_2)$$
$$\& (s_{in}F_1 \leq_S s_{in}F))\},$$

where $A = (d(F) = i) \& (\forall f \in \mathcal{F})(s_{in}F \notin Dom\ f)$,

$$gnr_{P^{min}}(i) = \begin{cases} V_i \backslash unp_{P^{min}}(i), & \text{if } fnl_{P^{min}}(i) = \emptyset \\ \emptyset, & \text{if } fnl_{P^{min}}(i) \neq \emptyset, \end{cases}$$

$$gnr_{P^{ir}}(i) = V_i \backslash unp_{P^{ir}}(i),$$

and $gnr_{P^{min}_{krnl}}(i)$ is any minimal by cardinality subset of the set $gnr_{P^{min}}(i)$ such that the set $\{s_{in}F | v_F \in gnr_{P^{min}_{krnl}}(i)\}$ consists of all minimal elements of the set $\{s_{in}F | v_F \in gnr_{P^{min}}(i)\}$.

If it is necessary to design some k-cooperative solution, it is natural to insert the following additional restriction into an $\mathcal{M}$-model $\mathcal{S}$: the set $(S, \leq_S)$ is *a semilattice*, i.e. for all $s_1, s_2 \in S$ there exists the unique element $s \in S$ such that: 1) $s \leq_S s_1$ and $s \leq_S s_2$; 2) $(\forall s' \in S)((s' \leq_S s_1) \& (s' \leq_S s_2) \implies (s' \leq_S s))$. Under this condition some slight modification of Algorithm 1 is directly applied, intended for the design of the set $\mathcal{L}^{min}_{krnl}(\mathcal{S})$, if we set $\omega_k = \inf\{s_1, \dots, s_k\}$ $(k \in \mathbb{N})$.

To illustrate the approach developed above, in the rest of this Section some problems of discrete mathematics will be briefly considered.

Example 1. *A finite automaton* is a system $M = (Q, X, Y, \delta, \lambda)$, where Q, X, and Y are finite sets, namely, *the set of states*, *the input alphabet*, and *the output alphabet*, $\delta : Q \times X \to Q$ is *the transition mapping*, and $\lambda : Q \times X \to Y$ is *the output mapping*. Mappings δ and λ are extended to the set $Q \times X^*$ via the identities

$$\tilde{\delta}(q, \Lambda) = q, \quad \tilde{\delta}(q, px) = \delta(\tilde{\delta}(q, p), x), \quad \tilde{\lambda}(q, \Lambda) = \Lambda,$$
$$\tilde{\lambda}(q, px) = \tilde{\lambda}(q, p)\lambda(\tilde{\delta}(q, p), x).$$

A weakly initialized automaton (w.i.a.) is an ordered pair (M, Q_0) such that M is a finite automaton and Q_0 $(Q_0 \subseteq Q, |Q_0| \geq 2)$ is *the set of initial states*. For a w.i.a. (M, Q_0) an input sequence $p \in X^+$ is
1) *a distinguishing* one if $(\forall q_1, q_2 \in Q_0)(\tilde{\lambda}(q_1, p) = \tilde{\lambda}(q_2, p) \Longrightarrow q_1 = q_2)$;
2) *a homing* one if $(\forall q_1, q_2 \in Q_0)(\tilde{\lambda}(q_1, p) = \tilde{\lambda}(q_2, p) \Longrightarrow \tilde{\delta}(q_1, p) = \tilde{\delta}(q_1, p))$;
3) *a synchronizing* one if $(\forall q_1, q_2 \in Q_0)(\tilde{\delta}(q_1, p) = \tilde{\delta}(q_1, p))$.

For a w.i.a. (M, Q_0) the set of all distinguishing, homing, or synchronizing sequences will be denoted by $D(M, Q_0)$, $H(M, Q_0)$, or $S(M, Q_0)$, respectively. These sets determine *simple preset experiments*, intended *for identifying a state* of a w.i.a. (M, Q_0).

Let $\mathbf{B}(|Q|, |Q_0|) = \{W \in \mathbf{P}(Q) | \; |W| \leq |Q_0|\}$ ($\mathbf{P}(Z)$ is *the power set* of a set Z) and $\mathcal{W}(|Q|, |Q_0|)$ be the subset of $\mathbf{P}(\mathbf{P}(Q))$ consisting of all elements W such that: 1) $|w| \geq 2$ for all $w \in W$; 2) if $w_1, w_2 \in W$ $(w_1 \neq w_2)$, then neither $w_1 \subset w_2$ nor $w_1 \subset w_1$; 3) $\sum_{w \in W} |w| \leq |Q_0|$. We set

$$\mathcal{S}^d_{(M,Q_0)} = (\mathcal{W}(|Q|, |Q_0|), X, \{Q_0\}, \{\emptyset\}),$$

$$\mathcal{S}^h_{(M,Q_0)} = (\mathcal{W}(|Q|, |Q_0|), X, \{Q_0\}, \{\emptyset\}),$$

$$\mathcal{S}^s_{(M,Q_0)} = (\mathbf{B}(|Q|, |Q_0|), X, Q_0, \mathbf{B}(|Q|, 1)),$$

where $Wx = \{\delta(q, x)\}(W \in \mathbf{B}(|Q|, |Q_0|), x \in X)$ and Wx $(W = \{w_1, \ldots, w_h\} \in \mathcal{W}(|Q|, |Q_0|), x \in X)$ is computed via the following Procedure 1 for a system $\mathcal{S}^d_{(M,Q_0)}$ and Procedure 2 for a system $\mathcal{S}^h_{(M,Q_0)}$

Procedure 1.

Step 1. $W_1 := \{w_j(y) | j = 1, \ldots, h; y \in Y\}$, where $w_j(y) = \{\delta(q, x) | q \in w_j, \lambda(q, x) = y\}$ $(j = 1, \ldots, h; y \in Y)$.

Step 2. If there exists $j \in \{1, \ldots, h\}$ such that $\sum_{y \in Y} |w_j(y)| < |w_j|$, then Wx is not determined and HALT, else go to Step 3.

Step 3. $Wx := \{w \in W_1 | \; |w| \geq 2\}$ and HALT.

Procedure 2.

Step 1. $W_1 := \{w_j(y) | j = 1, \ldots, h; y \in Y\}$, where $w_j(y) = \{\delta(q,x) | q \in w_j, \lambda(q,x) = y\}$ $(j = 1, \ldots, h; y \in Y)$.

Step 2. $Wx := \{w \in W_1 | \ |w| \geq 2\}$ and HALT.

Each of the sets $\mathcal{W}(|Q|, |Q_0|)$ and $\mathbf{B}(|Q|, |Q_0|)$ is partially ordered in the following way:

$$W_2 \leq_{\mathcal{W}} W_1 \iff (\forall w_2 \in W_2)(\exists w_1 \in W_1)(w_2 \subseteq w_1) \quad (W_1, W_2 \in \mathcal{W}(|Q|, |Q_0|)),$$

$$W_2 \leq_{\mathbf{B}} W_1 \iff W_2 \subseteq W_1 \quad (W_1, W_2 \in \mathbf{B}(|Q|, |Q_0|)).$$

It was verified in [9] that $\mathcal{S}^d_{(M,Q_0)}$, $\mathcal{S}^h_{(M,Q_0)}$, and $\mathcal{S}^s_{(M,Q_0)}$ are $\mathcal{M}$-models such that

$$\mathcal{L}(\mathcal{S}^d_{(M,Q_0)}) = D(M, Q_0), \quad \mathcal{L}(\mathcal{S}^h_{(M,Q_0)}) = H(M, Q_0), \quad \mathcal{L}(\mathcal{S}^s_{(M,Q_0)}) = S(M, Q_0).$$

Thus, *prefix* design (see [13]) of distinguishing, homing, and synchronizing sequences for a w.i.a. can be developed systematically on the basis of $\mathcal{M}$-models.

Let $Q^{(2)} = \{\{q_1, q_2\} | q_1, q_2 \in Q; q_1 \neq q_2\}$, $\mathbf{P}_{Q_0}(Q^{(2)}) = \{W \in \mathbf{P}(Q^{(2)}) | (\forall \{q_1, q_2\} \in W)(\{q_1, q_2\} \not\subseteq Q_0)\}$, Π_Q be the set of all partitions of the set Q, $\Pi^d_Q(Q_0) = \{\pi \in \Pi_Q | \ \pi|_{Q_0} = \mathbf{0}_{Q_0}\}$, $\Pi^s_Q(Q_0) = \{\pi \in \Pi_Q | \ \pi|_{Q_0} = \mathbf{1}_{Q_0}\}$. We set

$$\mathcal{T}^d_{(M,Q_0)} = (\Pi_Q, X, \mathbf{1}_Q, \Pi^d_Q(Q_0)),$$

$$\mathcal{T}^h_{(M,Q_0)} = (\mathbf{P}(Q^{(2)}), X, Q^{(2)}, \mathbf{P}_{Q_0}(Q^{(2)})),$$

$$\mathcal{T}^s_{(M,Q_0)} = (\Pi_Q, X, \mathbf{0}_Q, \Pi^s_Q(Q_0)).$$

Elementary operators are determined in the following way:
1) for a system $\mathcal{T}^d_{(M,Q_0)}$ we set $\pi x = \pi_1$ $(\pi, \pi_1 \in \Pi_Q; x \in X)$, where

$$q_1 \equiv q_2(\pi_1) \iff \delta(q_1, x) \equiv \delta(q_2, x)(\pi) \wedge \lambda(q_1, x) = \lambda(q_2, x);$$

2) for a system $\mathcal{T}^h_{(M,Q_0)}$ we set $Wx = W_1$ $(W, W_1 \in \mathbf{P}(Q^{(2)}); x \in X)$, where

$$W_1 = \{\{q_1, q_2\} \in Q^{(2)} | \{\delta(q_1, x), (q_2, x)\} \in W; \lambda(q_1, x) = \lambda(q_2, x)\};$$

3) for a system $\mathcal{T}^s_{(M,Q_0)}$ we set $\pi x = \pi_1$ $(\pi, \pi_1 \in \Pi_Q; x \in X)$, where

$$q_1 \equiv q_2(\pi_1) \iff \delta(q_1, x) \equiv \delta(q_2, x)(\pi).$$

The sets of situations of systems $\mathcal{T}^d_{(M,Q_0)}$, $\mathcal{T}^h_{(M,Q_0)}$, and $\mathcal{T}^s_{(M,Q_0)}$ are partially ordered in the following way:

1) for a system $\mathcal{T}^d_{(M,Q_0)}$ we set $\pi_1 \leq_d \pi_2 \iff (\forall B_1 \in \pi_1)(\exists B_2 \in \pi_2)(B_1 \subseteq B_2)$ $(\pi_1, \pi_2 \in \Pi_Q)$;
2) for a system $\mathcal{T}^h_{(M,Q_0)}$ we set $W_1 \leq_h W_2 \iff W_1 \subseteq W_2 \quad (W_1, W_2 \in \mathbf{P}(Q^{(2)}))$;
3) for a system $\mathcal{T}^s_{(M,Q_0)}$ we set $\pi_1 \leq_s \pi_2 \iff (\forall B_2 \in \pi_2)(\exists B_1 \in \pi_1)(B_2 \subseteq B_1)$ $(\pi_1, \pi_2 \in \Pi_Q)$.

It was verified in [9] that $\mathcal{T}^d_{(M,Q_0)}$, $\mathcal{T}^h_{(M,Q_0)}$, and $\mathcal{T}^s_{(M,Q_0)}$ are $\mathcal{M}$-models such that

$$\mathcal{L}(\mathcal{S}^d_{(M,Q_0)}) = (D(M,Q_0))^{-1},$$
$$\mathcal{L}(\mathcal{T}^h_{(M,Q_0)}) = (H(M,Q_0))^{-1},$$
$$\mathcal{L}(\mathcal{T}^s_{(M,Q_0)}) = (S(M,Q_0))^{-1}.$$

Thus, *suffix* design (see [14]) of distinguishing, homing, and synchronizing sequences for a w.i.a. can be developed systematically on the basis of $\mathcal{M}$-models. Moreover, since the set of situations for each of the $\mathcal{M}$-models $\mathcal{T}^d_{(M,Q_0)}$ and $\mathcal{T}^h_{(M,Q_0)}$ is a semilattice, these $\mathcal{M}$-models are valid for the design of k-cooperative solutions, i.e. *multiple preset distinguishing* and *homing* experiments with a w.i.a.

Example 2. It is well known that Boolean functions are widely used in the role of mathematical models for combinational circuits. It was verified in [9,15,16] that controllability/observability analysis for Boolean function $f(x_1, \ldots, x_n)$ is reduced to the design of *minterms* (i.e. of *prime implicants*). Indeed (see, for details, [9,12,16]), any *irreducible* set of α-controllability $(\alpha \in \{0,1\})$ for f is (in essence) some minterm for the Boolean function f^α, while any *irreducible* set of (i,α)-observability $(\alpha \in \{0,1\}; i = 1, \ldots, n)$ for f is (in essence) some minterm for the Boolean function $f_{i,\alpha}$, where

$$N_{f_{i,\alpha}} = \{(\beta_1, \ldots, \beta_{i-1}, \alpha, \beta_{i+1}, \ldots, \beta_n) \\ \in N_f | (\beta_1, \ldots, \beta_{i-1}, \overline{\alpha}, \beta_{i+1}, \ldots, \beta_n) \notin N_f\}.$$

For any Boolean function $g(x_1, \ldots, x_n)$ any minterm, which covers some fixed point $(\sigma_1, \ldots, \sigma_n) \in N_g$, can be designed via sequential deleting of literals in the implicant $x_1^{\sigma_1} \ldots x_n^{\sigma_n}$ (see, for details, [9,12,15,16]). Thus, a very simple $\mathcal{M}$-model, with the set of elementary operators being the generating set for some commutative semigroup, is applied to the design of minterms, which cover some fixed point. On this basis, backtracking for the design of disjunctive normal forms, consisting of minterms only, was developed. These forms are applied to estimation of controllability/observability parameters (via computing or estimation of the minimal rank of minterms for the analysed Boolean function), as well as to the design of *stuck-at faults* localization tests. We should note the following factor. For almost all Boolean functions $h \in P_2(n)$ $(n \to \infty)$ the value of the minimal rank of a minterm is an element of the segment $[n - \log n + 2; n - \log\log n + 1]$. Thus, if the rank of any minterm is selected in the role of an estimation of the controllability/observability parameters, then in almost all cases the mistake of approximation does not exceed the value $O(\log \frac{n}{\log n})$ $(n \to \infty)$.

Example 3. The design of *irreducible sets of representatives* for a given family of sets seems to be a model problem of discrete mathematics, to which many problems, both theoretic and applied, are naturally reduced (such as the design of these or other bases for Boolean functions, design of disjunctive normal forms, tests design for one-level combinational circuits, etc.) (see, for details, [9,11]). *An irreducible set of representatives* for a family of nonempty subsets $A = \{\alpha_i\}_{i\in I}$ of some set U is determined to be any set V $(V \subseteq U)$ such that $V \cap \alpha_i \neq \emptyset$ for all $i \in I$. The approach for the design of the set $\mathcal{V}_A$ of all irreducible sets of representatives for the family of sets A was developed in [9,11]. It consists of three stages. Firstly, the family A is partitioned into *maximal connected subfamilies* (and these subfamilies can be analysed independently) A_j $(j \in J)$ via the equivalence relation $\equiv_1$, determined as follows: $\alpha_1 {\equiv_1} \alpha_2$ if and only if there exists a finite sequence $\alpha_1 = \alpha_{i_1}, \alpha_{i_2}, \ldots, \alpha_{i_n} = \alpha_2$ of elements of A such that $\alpha_{i_r} \cap \alpha_{i_{r+1}} \neq \emptyset$ for all $r \in \{1, \ldots, n-1\}$. Secondly, every subfamily A_j $(j \in J)$ is converted into the family of *factor-sets* $\mathcal{A}_j = \{\alpha|_{\equiv_{j,2}} | \alpha \in A_j\}$, where $\equiv_{j,2}$ $(j \in J)$ is the equivalence relation on the set U such that

$$u_1 {\equiv_{j,2}} u_2 \Longleftrightarrow (\forall \alpha \in A_j)(u_1 \in \alpha \Longleftrightarrow u_2 \in \alpha)(u_1, u_2 \in U).$$

Thirdly, under the supposition that any equivalence class of the relation $\equiv_{j,2}$ *covers* all factor-sets $\alpha|_{\equiv_{j,2}}$ in which it is included (and, thus, the set of equivalence class of the relation $\equiv_{j,2}$ is converted into a partially-ordered one), there is implemented sequential design of all *irreducible coverings* (as well as of *minimal coverings*) of the set $\mathcal{A}_j$ via some $\mathcal{M}$-model, with the set of elementary operators being the generating set for some commutative semigroup.

5. DESIGN OF ADAPTIVE SOLUTIONS

The approach developed above can be easily extended for designing adaptive solutions.

Definition 8. *An $\mathcal{AM}$-model is a system $\mathcal{S} = (S, \mathcal{H}, s_{in}, S_{fin})$ $(H \subseteq \mathcal{F} \times \mathcal{G})$ such that some partial ordering $\leq_S$ is determined on the set S, so that*: 1) *if $s_1 \in S_{fin}$ and $s_2 {\leq_S} s_1$, then $s_2 \in S_{fin}$*; 2) *if $s_1 \in Dom\ (f,g)$ $((f,g) \in \mathcal{H})$ and $s_2 {\leq_S} s_1$, then $\emptyset \neq \mathcal{G}_{s_2,f} \subseteq \mathcal{G}_{s_1,f}$*; 3) *if $s_2 {\leq_S} s_1$, then $s_2(f,g) {\leq_S} s_1(f,g)$ $(g \in \mathcal{G}_{s_2,f})$.*

Lemma 3. *If $s_2 {\leq_S} s_1$, then $s_2 H {\leq_S} s_1 H$ for all $H \in \mathcal{H}^*$ such that $s_1, s_2 \in Dom\ H$.*

Proof. By induction over the length of an operator.

It is evident that if $|\mathcal{G}| = 1$, then an $\mathcal{AM}$-model $\mathcal{S}$ is also an $\mathcal{M}$-model. A very important interrelation exists between $\mathcal{AM}$-models and $\mathcal{M}$-models in the general case. Let some $\mathcal{AM}$-model $\mathcal{S} = (S, \mathcal{H}, s_{in}, S_{fin})$ ($H \subseteq \mathcal{F} \times \mathcal{G}$) be fixed. We set $\Sigma = \{\sigma \in \mathbf{P}(S) | (\forall s_1, s_2 \in \sigma)((s_1 \leq_S s_2) \vee (s_2 \leq_S s_1) \Rightarrow (s_1 = s_2))\}$ and will determine the partial ordering on the set Σ in the following way: $\sigma_1 \leq_\Sigma \sigma_2 \iff (\forall s_1 \in \sigma_1)(\exists s_2 \in \sigma_2)(s_1 \leq_S s_2)$. Any $f \in \mathcal{F}$ is considered as (possibly, partial) mapping $f : \Sigma \to \Sigma$ such that the value σf is computed via the following

Procedure 3.

Step 1. If there exists $s \in \sigma$ such that $\mathcal{G}_{s,f} = \emptyset$, then σf is not determined and HALT, else go to Step 2.

Step 2. $\sigma' := \bigcup_{s \in \sigma} \{s(f, g) | g \in \mathcal{G}_{s,f}\}$.

Step 3. $\sigma'' := \{s \in \sigma' | s$ *is the maximal element in* σ' *relative to the relation* $\leq_S\}$.

Step 4. $\sigma f := \sigma'' \backslash S_{fin}$ and HALT.

Thus, some model $\mathcal{C} = (\Sigma, \mathcal{F}, \{s_{in}\}, \emptyset)$ can be associated with any $\mathcal{AM}$-model $\mathcal{S}$. It was verified in [9] that $\mathcal{C}$ is an $\mathcal{M}$-model. Let $F = f_1 \ldots f_l \in \mathcal{L}^{ir}(\mathcal{C})$ and $B_F : \mathcal{G}^* \to \mathcal{F}$ be any partial mapping such that: 1) $\Lambda \in Dom\ B_F$; 2) $G(r) = g_1 \ldots g_r \in Dom\ B_F$ ($r = 1, \ldots, l$) if and only if $G(r-1) \in Dom\ B_F$, $s_{in} \in Dom\ (f_1, g_1) \ldots (f_r, g_r)$, and $s_{in}(f_1, g_1) \ldots (f_r, g_r) \notin S_{fin}$; 3) the identity $B_F(G(r)) = f_{r+1}$ holds for all $G(r) \in Dom\ B_F$ ($r = 0, \ldots, l-1$). It is evident that the mapping B_F ($F \in \mathcal{L}^{ir}(\mathcal{C})$) is some adaptive solution for the $\mathcal{AM}$-model $\mathcal{S}$.

Let $\mathcal{S} = (S, \mathcal{H}, s_{in}, S_{fin})$ ($H \subseteq \mathcal{F} \times \mathcal{G}$) be a fixed $\mathcal{AM}$-model. The relation $\leq_S$ can be extended to the set $\mathbf{P}(S)$ in the following way: $\sigma_1 \leq_S \sigma_2 \iff (\forall s_1 \in \sigma_1)(\exists s_2 \in \sigma_2)(s_1 \leq_S s_2)$ ($\sigma_1, \sigma_2 \in \mathcal{B}(S)$). We set

$$S_{H(f,*)} = \{s_{in} H(f, g) | (f, g) \in \mathcal{H}, s_{in} H \in Dom\ (f, g)\} \quad (H \in \mathcal{H}^*_{s_{in}}, f \in \mathcal{F}).$$

Let B be any adaptive solution for the $\mathcal{AM}$-model $\mathcal{S}$, $G(r) = g_1 \ldots g_r \in Dom\ B$, $f = B(G(r))$ and

$$H(r) = (B(\Lambda), g_1)(B(g_1), g_2) \ldots (B(g_1 \ldots g_{r-1}), g_r).$$

The following two propositions can be verified by backward induction (see, for details, [9]).

Lemma 4. *If there exists* $f' \in \mathcal{F}$ *such that* $\mathcal{G}_{s_{in} H(r), f'} \neq \emptyset$ *and* $S_{H(r)(f',*)} \leq_S S_{H(r)(f,*)}$, *then there exists some adaptive solution* B' *for an* $\mathcal{AM}$*-model* $\mathcal{S}$ *such that*: 1) $B'(G(i)) = B(G(i))$ *for all* $i = 0, 1, \ldots, r-1$; 2) $B'(G(r)) = f'$.

Lemma 5. *If there exists* $j \in \{1, \ldots, r-1\}$ *such that* $s_{in} H(j) \leq_S s_{in} H(r)$, *then there exists some adaptive solution* B' *for an* $\mathcal{AM}$*-model* $\mathcal{S}$ *such that*: 1) $B'(G(i)) = B(G(i))$ *for all* $i = 0, 1, \ldots, j-1$; 2) $B'(G(j)) = B(G(r))$.

Let $\rho : \mathcal{H}^*_{s_{in}} \to \mathbf{P}(\mathcal{F})$ be any mapping such that $\{S_{H(f,*)} | f \in \rho(H), \mathcal{G}_{s_{in}H,f} \neq \emptyset\}$ is a minimal by cardinality set consisting of all minimal (relative to partial ordering $\leq_S$) elements of the set $\{S_{H(f,*)} | f \in \mathcal{F}, \mathcal{G}_{s_{in}H,f} \neq \emptyset\}$. Lemmas 4 and 5 imply that to design some adaptive solution for the $\mathcal{AM}$-model $\mathcal{S}$, it is sufficient to design a subtree $\mathcal{D}^a_{\mathcal{S}}$ of the tree $\mathcal{D}_{\mathcal{S}}$, determined by the following cutting rules: 1) a vertex v_{Hf} $(H \in \mathcal{H}^*, f \in \mathcal{F})$ and all its descendants are deleted if $f \notin \rho(H)$; 2) a vertex v_H $(H \in \mathcal{H}^*)$ and all its descendants are deleted if some vertex $v_{H'}$ $(d(H') < d(H))$ is placed on the path leading from the root v_Λ to v_H such that $s_{in}H' \leq_S s_{in}H$; 3) a vertex v_H and all its descendants are deleted if some vertex $v_{H'}$ $(d(H') < d(H))$ is placed on the path leading from the root v_Λ to v_H such that $s_{in}H \in S_{fin}$. It is evident that the subtree $\mathcal{D}^a_{\mathcal{S}}$ can be transformed into some (possibly partial) initialized finite automaton. Thus, some unified approach intended to present adaptive solutions for $\mathcal{AM}$-models via finite automata is developed.

Example 4. $\mathcal{AM}$-models provide us with powerful tools, sufficient to implement *adaptive experiments* with the given w.i.a. (M, Q_0) in the form of *automata-experimenters*. Indeed, let us consider *a generalized adaptive homing experiment*, i.e. the one intended for identifying the block of the given partition $\pi \in \Pi_Q$, in which the final state of the w.i.a. is contained. It is sufficient to design some adaptive solution for the $\mathcal{AM}$-model

$$\mathcal{S}^{ah}_{(M,Q_0)}(\pi) = (\mathbf{B}(|Q|, |Q_0|), X \times Y, Q_0, \mathbf{B}^{(\pi)}(|Q|, |Q_0|)),$$

where $\mathbf{B}^{(\pi)}(|Q|, |Q_0|) = \{\sigma \in \mathbf{B}(|Q|, |Q_0|) | (\exists B \in \pi)(\sigma \subseteq B)\}$ and $\sigma(x, y)$ $(\sigma \in \mathbf{B}(|Q|, |Q_0|), (x, y) \in X \times Y)$ is computed via the following

Procedure 4.

Step 1. If $\lambda(q, x) \neq y$ for all $q \in \sigma$, then $\sigma(x, y)$ is not determined and HALT, else go to Step 2.

Step 2. $\sigma(x, y) := \{\delta(q, x) | q \in \sigma; \lambda(q, x) = y\}$ and HALT.

If the above adaptive solution exists, the tree $\mathcal{D}^a_{\mathcal{S}^{ah}_{(M,Q_0)}(\pi)}$ can be easily transformed into the corresponding automaton-experimenter.

Similarly, let us consider *an adaptive distinguishing experiment*. It is sufficient to design some adaptive solution for the $\mathcal{AM}$-model $\mathcal{S}^{ad}_{(M,Q_0)} = (\mathbf{B}(|Q|, |Q_0|), X \times Y, Q_0, (\mathbf{B}(|Q|, 1))$, where $\sigma(x, y)$ $(\sigma \in \mathbf{B}(|Q|, |Q_0|), (x, y) \in X \times Y)$ is computed via the following

Procedure 5.

Step 1. If $\lambda(q, x) \neq y$ for all $q \in \sigma$, then $\sigma(x, y)$ is not determined and HALT, else go to Step 2.

Step 2. If there exist $q_1, q_2 \in \sigma$ $(q_1 \neq q_2)$ such that $\delta(q_1, x) = \delta(q_2, x)$ and $\lambda(q_1, x) = \lambda(q_2, x)$, then $\sigma(x, y)$ is not determined and HALT, else go to Step 3.

Step 3. $\sigma(x, y) = \{\delta(q, x) | q \in \sigma; \lambda(q, x) = y\}$ and HALT.

If the above adaptive solution exists, the tree $\mathcal{D}^a_{\mathcal{S}^{ad}_{(M,Q_0)}}(\pi)$ can be easily transformed into the corresponding automaton-experimenter.

6. CONCLUSIONS

In this paper the basics of the *searching metatheory* were worked out. The schemes of breadth-first searching, backtracking, and the branch-and-bound method were developed, in essence, in the form of some transducer's routing via some potentially infinite tree, whose vertices are labelled by finite labelled trees. Transducer's actions were presented in terms of sets $fnl(i)$, $unp(i)$, and $gnr(i)$ for any specific problem and any searching scheme. Thus, these sets form some base for unified analysis of inherent complexity of specific problems. Besides, any researcher is provided with the means *to dispose* of the same transducer to resolve different problems. The last possibility was illustrated in Section 4 by resolving problems for the design of minimal and irreducible solutions.

The notions of an $\mathcal{M}$-model and an $\mathcal{AM}$-model were determined in terms of monotone operators acting in some partially-ordered set of situations. The interrelation established between these models provides us with some means, sufficient to apply some additional information to terminate searching on some intermediate stage. In particular, this is the generalization of the results, established in [^{12}] for experiments with finite automata. The power of $\mathcal{M}$-models and $\mathcal{AM}$-models is characterized by their successful applications to resolving specific problems of discrete mathematics (some illustrations were presented in the paper via examples). Thus, it looks attractive to develop a similar approach for *local search* (including *genetic algorithms*). This is the subject of future research.

REFERENCES

1. Banerji, R. B. *Theory of Problem Solving: An Approach to Artificial Intelligence*. American Elsevier Publishing Company, INC., New York, 1969.
2. Nilson, N. J. *Problem-Solving Methods in Artificial Intelligence*. McGrow-Hill, New York, 1971.
3. Nilson N. J. *Principles of Artificial Intelligence*. Tioga Publishing Company, Palo Alto, California, 1980.
4. Skobelev, V. G. Searching in partially ordered structures. *Ukrainian Math. J.*, 1992, **44**, 253–260 (in Russian).
5. Skobelev, V. G. Problem-solving: combinatorial-algebraic approach. *ZAMM*, 1996, **76**, S3, 563–564.

6. Skobelev, V. G. Searching of adaptive solutions in partially ordered structures; artificial intelligence. *Publ. Inst. of Problems of Artificial Intelligence National Acad. Sci. Ukraine (Donetsk, Ukraine)*, 1999, 6, 6–13 (in Russian).
7. Skobelev, V. G. General schemes for searching of preset solutions. *Rep. National Acad. Sci. Ukraine (Kiev, Ukraine)*, 2000, 12, 82–86 (in Russian).
8. Skobelev, V. G. General schemes for searching in partially ordered structures. *Trans. Inst. Appl. Math. Mech. National Acad. Sci. Ukraine (Donetsk, Ukraine)*, 2000, **5**, 132–140 (in Russian).
9. Skobelev, V. G. *Analysis of Discrete Systems*. Institute of Applied Mathematics and Mechanics of National Academy of Sciences of Ukraine, Donetsk, Ukraine, 2002 (in Russian).
10. Skobelev, V. G. Algorithms and complexity of identification of internal states of finite automata. *Rep. National Acad. Sci. Ukraine (Kiev, Ukraine)*, 1981, 7, 71–74 (in Russian).
11. Skobelev, V. G. Design of irreducible sets of represetatives for a family of sets. *ZAMM*, 1996, **79**, S3, 915–916.
12. Skobelev, V. G. Combinatorial algorithms for design of disjunctive normal form (DNF). *Rep. National Acad. Sci. Ukraine (Kiev, Ukraine)*, 1989, 2, 72–75 (in Russian).
13. Gill, A. *Introduction to the Theory of Finite-State Machines*. McGrow-Hill Book Company, New York, 1962.
14. Starke, P. Über experimente an automaten. *Z. Math. Logik Grundlag Math.*, 1962, **13**, 67–70.
15. Skobelev, V. G. Algebraic models and methods for combinational circuits identification. *Radioelectronics & Informatics (Kharkov, Ukraine)*, 2003, 3, 127–131.
16. Skobelev, V. G. Controllability and observability of Boolean functions and of their compositions. *Rep. National Acad. Sci. Ukraine (Kiev, Ukraine)*, 1987, 7, 65–67 (in Russian).

Otsimine: mudelid ja meetodid

Volodymyr G. Skobelev

Esitatakse üldised teoreetilised mudelid otsimiseks osaliselt järjestatud struktuurides eeldusel, et elementaaroperaatorid on monotoonsed: $\mathcal{M}$-mudel fikseeritud ja $\mathcal{AM}$-mudel adaptiivsete lahendite jaoks. Demonstreeritakse esitatud mudelite rakendust mõnele diskreetse matemaatika probleemile: hulkade pere taandatud esindajate hulga leidmine, eksperimendid lõplike automaatidega ja mintermide ning täieliku disjunktiivse normaalkuju konstrueerimine.