Tomus 3. Fasciculus 2.

ACTA CYBERNETICA

FORUM CENTRALE PUBLICATIONUM CYBERNETICARUM HUNGARICUM

Szeged, 1977
Curat: Universitas Szegediensis de Attila József nominata

3. kötet 2. füzet

ACTA CYBERNETICA

A HAZAI KIBERNETIKAI KUTATÁSOK KÖZPONTI PUBLIKÁCIÓS FÓRUMA

Szeged, 1977. augusztus
A Szegedi József Attila Tudományegyetem gondozásában

On graphs satisfying some conditions for cycles, II.

By A. Ádám

Introduction

In this paper we study another class (containing all cycles) of finite directed graphs, than in Part I. Let a class be introduced as follows: (i) all cycles belong to the class, (ii) whenever a graph G_0 is contained in the class and we replace a simple vertex P of G_0 by a cycle, then the new graph G is again an element of the class, (iii) the class is as narrow as possible with respect to the rules (i), (ii). The members of this class are called the A-constructible graphs. (A more detailed definition will be given in § 1.)

An advantage of this recursive definition is its simplicity; it has, however, the disadvantage that is does not give the A-constructible graphs uniquely (the same graph can be produced in essentially different ways). Therefore another recursive procedure (called Construction B) will be exposed such that it admits a decomposition statement (Theorem 1) and it yields all the A-constructible graphs (Theorem 2). (As it may be foreseen, Construction B is described more elaborately, than Construction A.) Finally, it is shown that the class of B-constructible graphs is wider, than the class of the A-constructible ones. We deal with the question (without solving it completely) how the A-constructible graphs can be characterized in terms of Construction B.

§ 1. The Constructions A, B

1.1.

Construction A. The construction consists of an initial step and a finite number ($\geqq 0$) of ordinary steps.

Initial step. Let us consider a cycle of length n ($\geqq 2$).

Ordinary step. Suppose that the preceding (initial or ordinary) step has produced the graph G_0. Consider G_0 and a cycle z of length m ($\geqq 2$) such that G_0, z are disjoint. Choose a simple vertex P in G_0; denote by e_1, e_2 the edges incoming to P or outgoing from P, resp. Furthermore, choose two different vertices A, B in z. Let us unite G_0 and z such that P is deleted, A becomes the new final vertex of e_1 and B is the new initial vertex of e_2.

A graph G is called A-*constructible* if G can be built up by Construction A[1].

1.2. Let G be a graph. We denote by $K(G)$ the maximum of the numbers $Z(e)$ where e runs through the edges of G. An edge e_0 (of G) is called *extremal* if $Z(e_0)=K(G)$. Denote by G' the subgraph of G consisting of the extremal edges (in G) and the vertices incident to them. G' is not connected in general. The connected components of G' are called the *extremal subgraphs* of G. If an extremal subgraph is a path only (having one or more edges), then we call it an *extremal path*.

1.3.

Construction B. The construction consists of a finite number ($\geqq 1$) of steps any of which is either an inital step or an ordinary one in the following sense.

Initial step. Let us consider a graph G such that

either G is a cycle (of length $\geqq 1$),

or G is I*-constructible[2] and G has no cut vertex (and, of course, G has neither a loop nor a pair of parallel edges with the same orientation).

Ordinary step. Let us consider a graph G_0 and a matrix

$$\begin{pmatrix} A_1 & A_2 \dots A_k \\ B_1 & B_2 \dots B_k \\ G_1 & G_2 \dots G_k \\ P_1 & P_2 \dots P_k \end{pmatrix}$$

(having four rows and k ($\geqq 1$) columns) such that

(α) any of the $k+1$ graphs $G_0, G_1, G_2, \dots, G_k$ is isomorphic to a graph produced in some earlier step of the construction,[3]

(β) $K(G_0) \geqq \max(2, K(G_1), K(G_2), \dots, K(G_k))$,

(γ) $A_1, A_2, \dots, A_k, B_1, B_2, \dots, B_k$ are pairwise different simple vertices of G_0,

(δ) for any subscript i ($1 \leqq i \leqq k$), G_0 has an extremal path[4] a_i with the following properties:

A_i precedes B_i along a_i, and

the set of vertices lying between A_i, B_i on a_i is disjoint to the set $\{A_1, A_2, \dots, A_k, B_1, B_2, \dots, B_k\}$,

(ε) for any i ($1 \leqq i \leqq k$), P_i is a simple vertex of G_i and $Z(P_i)=1$ holds (in G_i).

Denote by $e_1^{(i)}$, $e_2^{(i)}$ the edges incoming to P_i and outgoing from P_i, resp. (in G_i).

[1] I.e. if there exists a finite sequence of steps such that the first one is an initial step, the other ones are ordinary steps and the last step produces G.

[2] We call a graph I*-constructible of it can be produced by Construction I exposed in § 3 of [1]. The term "I*-constructible" has been used in the same sense in [2].

[3] It is permitted that both G_{j_1} and G_{j_2} are isomorphic to the result of the *same* previous step, though $j_1 \neq j_2$. G_{j_1} and G_{j_2} are considered to be disjoint even in this case.

[4] The paths $a_1, a_2, \dots, a_k$ are not necessarily different.

Let us construct a new graph such that, for every subscript i $(1 \leqq i \leqq k)$, we delete P_i (out of G_i), A_i becomes the new final vertex of $e_1^{(i)}$ and B_i becomes the new initial vertex of $e_2^{(i)}$. (This means that the situation (a) is replaced by the situation (b) on Fig. 1.)

A graph G is called B-*constructible* if G can be built up by Construction B.

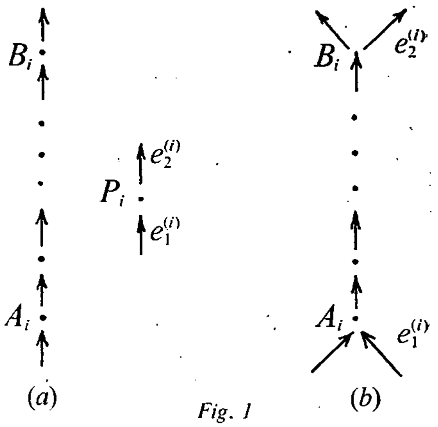


Fig. 1

1.4.

Proposition 1. *Suppose that G is produced by an ordinary step of Construction* B. *Then G has precisely k extremal subgraphs, namely, the part a_i' of a_i from A_i to B_i for each i $(1 \leqq i \leqq k)$.*

Proof. Denote by $Z(e)$, $Z_i(e)$ the number of cycles containing an edge e, meant in G, G_i, respectively. The rules in the ordinary step (chiefly (δ)) imply

$$Z(e) = 1 + Z_0(e) = 1 + K(G_0)$$

whenever e belongs to some a_i'. It is clear that

$$Z(e) = Z_0(e) \leqq K(G_0)$$

is true for the other edges of G_0 and, for any i $(1 \leqq i \leqq k)$,

$$Z(e) = Z_i(e) \leqq K(G_i) \leqq K(G_0)$$

holds (by (β)) if e is an arbitrary edge of G_i.

The above proof and (β) guarantee the following assertion, too:

Proposition 2. *If G can be represented as the result of an ordinary step Construction* B, *then*

$$K(G)(= 1 + K(G_0)) \geqq 3.$$

Proposition 3. *If G is* B-*constructible and $K(G) \geqq 2$, then each extremal subgraph of G is a path and the inner vertices of the extremal paths of G are simple.*

Proof. Case 1. G results by an initial step (of Construction B) only. We assumed $K(G) \geqq 2$, it is hence obvious that $K(G) = 2$ and G is I*-constructible. The conclusion is fulfilled because of Construction I in [1].

Case 2. G is produced by an ordinary step. We use induction: we suppose that G_0 satisfies the conclusion of Proposition 3. Proposition 1 implies that each extremal subgraph of G is a part of an extremal path of G_0, thus Proposition 3 is valid also for G.

The next result is implied immediately by Propositions 1, 2 and the assumptions in Construction B:

Proposition 4. *Let the graph G be represented as the result of an ordinary step of Construction* B. *Denote the extremal paths of G by $a_1, a_2, \ldots, a_k$; let the initial vertex of a_i be A_i and the final vertex of a_i be B_i (where $1 \leqq i \leqq k$). Then*

the degree of A_i is (2, 1) and we have $Z(e_i^{(1)})=1$, $Z(e_i^{(2)}) \geqq 2$ where $e_i^{(1)}$ and $e_i^{(2)}$ are the edges incoming to A_i with appropriate superscripts,

the degree of B_i is (1, 2) and we have $Z(e_i^{(3)})=1$, $Z(e_i^{(4)}) \geqq 2$ where $e_i^{(3)}$ and $e_i^{(4)}$ are the edges outgoing from B_i with appropriate superscripts.[5]

§ 2. Some notions concerning Construction B

2.1. Let us consider a particular application of Construction B consisting of q steps. We say that the relation $i \prec j$ is true (where $\{i, j\} \subseteqq \{1, 2, \ldots, q\}$) precisely if

$i<j$,

the j-th step is ordinary, and

the graph G resulting in the i-th step is isomorphic to one of the graphs $G_0, G_1, G_2, \ldots, G_k$ used in the j-th step.

We denote by $\lessdot$ the transitive extension of the relation $\prec$ (in the set $\{1, 2, \ldots, q\}$). It is obvious that $\lessdot$ is a partial ordering and $i \lessdot j$ may hold only if $i<j$. The definition of Construction B implies that, to any fixed j, $i \lessdot j$ is satisfiable (by some i) exactly if the j-th step is ordinary.

An application of Construction B, consisting of q steps, is called *connected* when all the $q-1$ relations $1 \lessdot q, 2 \lessdot q, \ldots, q-1 \lessdot q$ are true.

2.2. Two initial steps, occurring in particular performances of Construction B, are called *isomorphic* if the graphs appearing in them are isomorphic.

Let us consider two ordinary steps (again in Construction B) such that the number k is common. Denote the graphs and vertices, occurring in the first of these steps, by $G_0', G_1', A_1', B_1', P_1', \ldots, G_k', A_k', B_k', P_k'$; analogously, let the graphs and vertices of the second step in question be $G_0'', G_1'', A_1'', B_1'', P_1'', \ldots, G_k'', A_k'', B_k'', P_k''$. We call the considered steps to be *isomorphic* if there exist

(i) an isomorphism α of G_0' onto G_0'',

(ii) a permutation π of the set $\{1, 2, \ldots, k\}$, and

(iii) for every choice of i ($1 \leqq i \leqq k$), an isomorphism β_i of G_i' onto $G_{\pi(i)}''$

such that the equalities

$$\alpha(A_i') = A_{\pi(i)}'', \quad \alpha(B_i') = B_{\pi(i)}'', \quad \beta_i(P_i') = P_{\pi(i)}''$$

are fulfilled for each i ($1 \leqq i \leqq k$).

If two ordinary steps are isomorphic, then the originating graphs are again isomorphic.

A performance of Construction B is called *simple* if the i-th and j-th steps in it are not isomorphic unless $i=j$.

[5] It is clear that $e_i^{(1)}$, $e_i^{(3)}$ have been taken from G_i; $e_i^{(2)}$, $e_i^{(4)}$ have been taken from G_0.

2.3. Two applications Q_1, Q_2 of Construction B are said to be *similar* if the number q of their steps is the same and there exists a permutation σ of the set $\{1, 2, \ldots, q\}$ such that

the relation $i \prec_1 j$ holds if and only if $\sigma(i) \prec_2 \sigma(j)$ (where $\prec_l$ means the relation $\prec$ with respect to Q_l, $1 \leqq l \leqq 2$), and

in case of any i $(1 \leqq i \leqq q)$, the i-th step of Q_1 is isomorphic to the $\sigma(i)$-th step of Q_2.

§ 3. The inverse construction

3.1. Suppose that a graph G results by an ordinary step of some particular application of Construction B. The main goal of this § is to produce the $k+1$ graphs $G_0, G_1, G_2, \ldots, G_k$ and the $3k$ vertices $A_1, B_1, P_1, A_2, B_2, P_2, \ldots, A_k, B_k, P_k$ (occurring in the ordinary step) by using the properties of G solely. This will lead to the statement that each B-constructible graph can be represented by (one and) only one simple, connected performance of Construction B apart from similarity.

Proposition 5. *If G is a graph mentioned in the initial step of Construction* B, *then there is no Construction* B *which would give G as the result of an ordinary step.*

Proof. Since any graph G occurring in the initial step satisfies $1 \leqq K(G) \leqq 2$ evidently, the statement to be proved follows immediately from Proposition 2.

3.2.

Construction C. Let G be a (finite) graph such that

[α] $$K(G) \geqq 3,$$

[β] every extremal subgraph of G is a path (denote them by $a_1, a_2, \ldots, a_k$; let the initial and final vertex of a_i be A_i, B_i, resp., where $1 \leqq i \leqq k$),

[γ] for any i, each inner vertex of a_i is simple,

[δ] for any i, the degree of A_i is (2, 1) moreover, $Z(e_i^{(1)})=1$ and $Z(e_i^{(2)}) \geqq 2$ hold for the edges incoming to A_i if they are denoted appropriately,

[ε] for any i, the degree of B_i is (1, 2), furthermore, $Z(e_i^{(3)})=1$ and $Z(e_i^{(4)}) \geqq 2$ are true for the edges outgoing from B_i if they are denoted suitably,

[ζ] for any i, the pair $e_i^{(1)}, e_i^{(3)}$ can be connected by a chain which contains neither A_i nor B_i as an inner vertex; the analogous statement is true for the pair $e_i^{(2)}, e_i^{(4)}$ too,

[η] for any i, each chain connecting $e_i^{(1)}$ and $e_i^{(4)}$ contains either A_i or B_i innerly and the chains connecting $e_i^{(2)}, e_i^{(3)}$ do the same.

Let us form $k+1$ new graphs $G_0, G_1, G_2, \ldots, G_k$ (from G) in the following way:

(1) we take k new vertices $P_1, P_2, \ldots, P_k$,

(2) for any i $(1 \leqq i \leqq k)$, let $e_i^{(1)}$ go into P_i (instead of A_i) and let $e_i^{(3)}$ come out of P_i (instead of B_i); denote the resulting (non-connected) graph by G^*,

(3) let $G_0, G_1, G_2, \ldots, G_k$ be the connected components of G^* with such subscripts that[6] whenever $1 \leqq i \leqq k$, then G_i contains $e_i^{(1)}, e_i^{(3)}$, and G_0 contains none of $e_1^{(1)}, e_1^{(3)}, e_2^{(1)}, e_2^{(3)}, \ldots, e_k^{(1)}, e_k^{(3)}$.

[6] [ζ] and [η] guarantee that the number of connected components is $k+1$ and the conditions to be posed are satisfiable.

Thus Construction C is completed.

It is evident that, if [α]—[η] are fulfilled, then G uniquely defines k and the graphs $G_0, G_1, G_2, \ldots, G_k$ resulting by Construction C (apart from the numbering of $G_1, G_2, \ldots, G_k$).

3.3.

Proposition 6. *Assume that the graph G results by an ordinary step of Construction* B *such that the graphs and vertices (occurring in the step) are $G_0', G_1', G_2', \ldots, G_k'$ and $A_1', B_1', P_1', A_2', B_2', P_2', \ldots, A_k', B_k', P_k'$, respectively. Then Construction* C *is applicable for G. Let us apply Construction* C *for G; denote the resulting graphs by $G_0'', G_1'', G_2'', \ldots, G_k''$ and the vertices, playing essential roles in the construction, by $A_1'', B_1'', P_1'', A_2'', B_2'', P_2'', \ldots, A_k'', B_k'', P_k''$. In this case $G_0' = G_0''$ and there exists a permutation π of the set $\{1, 2, \ldots, k\}$ which satisfies*

$$G_i' = G_{\pi(i)}'', \quad A_i' = A_{\pi(i)}'', \quad B_i' = B_{\pi(i)}'', \quad P_i' = P_{\pi(i)}''$$

for each i $(1 \leqq i \leqq k)$.

Proof. Let us take into account the obvious fact that the cycles of G_0' and (essentially) the cycles of $G_1', G_2', \ldots, G_k'$ become the cycles of G, moreover, G does not contain any other cycle.

The conditions [α]—[η] of Construction C are true for G; in detail,

[α] is ensured by Proposition 2,

[β], [γ] are by Proposition 3,

[δ], [ε] are by Proposition 4,

[ζ], [η] follow from the suppositions (γ), (δ), (ε) occurring in the ordinary step of Construction B.

The applicability of Construction C has been shown. Using Proposition 1, we can convince ourselves that G_0'' coincides with G_0' and the system $\{G_1'', G_2'', \ldots, G_k''\}$ equals the system $\{G_1', G_2', \ldots, G_k'\}$ (up to labelling). Hence also the coincidence of the vertices A_i, B_i, P_i (as stated in the Proposition) follows.

Theorem 1. *Let two applications Q_1, Q_2 of Construction* B *be considered such that they produce the same graph G. If Q_1 and Q_2 are simple and connected, then they are similar.*

Proof. Denote the number of steps of Q_1, Q_2 by q_1, q_2 respectively. In the sequel, we shall apply Proposition 6 and the last sentence of Section 3.2 without any particular reference.

Let a relation ϱ be defined between the sets $R_1 = \{1, 2, \ldots, q_1\}$ and $R_2 = \{1, 2, \ldots, q_2\}$ followingly: $\varrho(i, j)$ holds precisely when the graph resulting in the i-th step of Q_1 is isomorphic to the graph originating in the j-th step of Q_2 (where $1 \leqq i \leqq q_1$, $1 \leqq j \leqq q_2$). Because Q_1 and Q_2 are simple, ϱ is a one-to-one assignment between some subset R_1' of R_1 and some subset R_2' of R_2. We can write $\sigma(i) = j$ instead of $\varrho(i, j) = \uparrow$.

Our next purpose is to show that $R_1' = R_1$ and $R_2' = R_2$. Put $i \in R_1$. Since Q_1 is connected, there exists a sequence $i_0, i_1, i_2, \ldots, i_s$ such that

$$i = i_0 \prec_1 i_1 \prec_1 i_2 \prec_1 \ldots \prec_1 i_s = q_1$$

($s \geqq 0$). It is obvious that $\sigma(i_s)=q_2$, thus $i_s \in R_1'$. Whenever i_t belongs to R_1', then i_{t-1} does the same ($1 \leqq t \leqq s$). Consequently, $R_1'=R_1$ and the equality $R_2'=R_2$ follows by an analogous inference (therefore $q_1=q_2$).

We are going to verify that σ establishes a similarity. In order to do this, it remains to show that σ preserves the relation $\lessdot$ (in both directions). If $i \lessdot_1 i^*$, then

$$i = i_0 \prec_1 i_1 \prec_1 i_2 \prec_1 \dots \prec_1 i_w = i^*$$

for suitable numbers $i_0, i_1, \dots, i_w$. For any t ($1 \leqq t \leqq w$), the graph resulting in the $\sigma(t-1)$-th step of Q_2 is utilized in the $\sigma(t)$-th step of Q_2, thus $\sigma(t-1)<\sigma(t)$ (since Q_2 is simple) and $\sigma(t-1) \prec_2 \sigma(t)$. Hence $\sigma(i) \lessdot_2 \sigma(i^*)$. — Conversely, $i \lessdot_2 i^*$ implies $\sigma^{-1}(i) \lessdot_1 \sigma^{-1}(i^*)$ by a symmetrical inference.

Corollary. *Let Q_1, Q_2, G be as in the first sentence of Theorem 1. Denote the number of the steps of these constructions by q_1, q_2, respectively. If Q_1 is simple and connected, then $q_1 \leqq q_2$.*

Proof. We can reduce Q_2 into a simple and connected construction Q_2' followingly:

whenever $1 \leqq i < q_2$ and neither the i-th, q_2-th steps are isomorphic nor the relation $i \lessdot q_2$ holds, then the i-th step is deleted,

whenever $1 \leqq i < j \leqq q_2$ and the i-th, j-th steps are isomorphic, then the j-th step is deleted.

Let us define r as the smallest number with the property that the r-th and q_2-th steps of Q_2 are isomorphic. It is easy to see that

each of the $(r+1)$-th, $(r+2)$-th, ..., q_2-th steps of Q_2 is deleted by virtue of the above rules, and

the r-th step of Q_2 becomes the last step[7] of Q_2'.

We get $q_1=q_2' \leqq q_2$ where q_2' is the number of steps of Q_2'.

§ 4. Interrelations between A-constructibility and B-constructibility

4.1.

Theorem 2. *Each A-constructible graph is B-constructible.*

Proof. For cycles the assertion is trivial. Otherwise, we use induction for the number of edges. Let an A-constructible graph G be considered, suppose that every A-constructible graph, having a fewer number of edges than G, is B-constructible. By the definition of the A-constructibility, there is an A-constructible graph G^* and a simple vertex P of G^* such that G can be produced if we insert a cycle (of length l) for P in G^* (in sense of the ordinary step of Construction A). G^* is B-constructible by the induction hypothesis.

[7] It may happen that some of the first, second, ..., $(r-1)$-th steps of Q_2 are also deleted.

Let us consider a performance Q^* of Construction B which produces G^*. In what follows, our aim is to modify Q^* such that the new construction should give G. For the sake of simplicity, we agree that the construction steps of Q^* will always be mentioned as they are numbered in Q^*.

We define a sequence

$$D_1, D_2, \ldots, D_s \quad (s \geqq 1)$$

of vertices and a sequence

$$j_1, j_2, \ldots, j_s \quad (j_1 > j_2 > \ldots > j_s)$$

of numbers (indicating steps) in the following (recursive) manner:

D_1 is P (a vertex of the graph G^* resulting in the last step of Q^*) and j_1 is the number of the steps of Q^*,

if D_i has already been defined, it belongs to the graph originating in the j_i-th step of Q^* and the step in question is ordinary, then let j_{i+1} $(<j_i)$ be such a number that the result of the j_{i+1}-th step occurs among the graphs appearing (as $G_0, G_1, G_2, \ldots, \ldots, G_k$) in the j_i-th step and D_i corresponds to some vertex D_{i+1} of the result of the j_{i+1}-th step (by virtue of an isomorphism mentioned in Construction B, (α)),

if D_i has been defined as a vertex of a graph originating in the j_i-th step of Q^* such that this step is initial, then we put $s=i$ and the process terminates.

We remark that each D_i is a simple vertex of the containing graph.

Next we define s or $s+1$ new construction steps which are called j_1'-th step, j_2'-th step, ..., j_s'-th step and, in some cases, j_0'-th step.

Case 1. $Z(D_s)=1$ in the graph $G^{(1)}$ resulting by the j_s-th step. $G^{(1)}$ is I^*-constructible. The graph $G'^{(1)}$ originating from $G^{(1)}$ by inserting a cycle of length l at D_s (as in the ordinary step of Construction A) is again I^*-constructible. Let the j_s'-th step be initial, let it produce $G'^{(1)}$. — Suppose that the j_i'-th step has been defined $(1 \leqq i < s)$, we define a new construction step and call it the j_{i+1}'-th one in the following manner: the new step differs from the j_{i+1}'-th one only in that respect that now the (uniquely determined) graph containing D_{s-i} is replaced by the result or the j_i'-th step. (The graph resulting in the j_{i+1}'-th step will contain a cycle of length l instead of D_s, otherwise it will coincide with the graph originating in the j_{s-i}-th step.)

Let us draw up a new construction Q followingly:

it contains all the steps of Q^* except the last one (in the original ordering),

for every i $(1 \leqq i < s)$, let the j_i'-th step be inserted between the j_{s-i+1}-th and $(j_{s-i+1}+1)$-th ones,

the last step of Q is the j_s'-th step.

It is obvious that Q is an application[8] of Construction B and Q produces G.

Case 2. $Z(D_s)=2$ in the result $G^{(1)}$ of the j_s-th step. Let an initial step, called j_0'-th one, be defined in such a manner that it produces a slighthly modified copy of $G^{(1)}$ with the single difference that D_s is replaced by the path a whose length equals the (directed!) distance d of A and B in the last step of the performance of Construction A producing G.

[8] Q is not simple and connected in general even if Q^* has these properties.

Now the j_1'-th step is ordinary such that

$k=1$,

G_0 is the result of the j_0'-th step,

G_1 is the cycle of length $l-d$,

A_1 and B_1 are the beginning and final vertices of a (see how the j_0'-th step is defined), respectively,

P_1 is an arbitrary vertex of G_1.

The further treatment of Case 2 is similar to Case 1. Now both the j_0'-th and i_1'-th steps (in this ordering) are inserted between the j_s-th and (j_s+1)-th ones.

4.2. The collection of A-constructible graphs is properly included in the family of B-constructible ones. An example for a B-constructible graph which is not A-constructible may be the cycle of length 1; a less trivial counter-example can be seen on Fig. 2. (One can check by applying Construction C that this graph is B-constructible. On the other hand, it does not contain any cycle which would be resulted in the last step of Construction A. — The numbers in Fig. 2 indicate the values of $Z(e)$.)

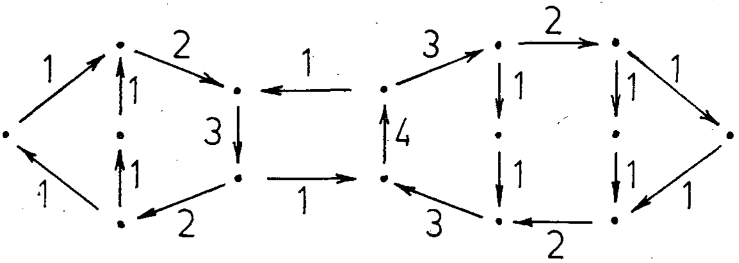


Fig. 2

4.3. The existence of counter-examples (similar to the above one) *disproves* the following statement: whenever each of $G_0, G_1, G_2, \ldots, G_k$ in an ordinary step of Construction B is A-constructible, then G is again A-constructible. However, the converse assertion is valid:

Proposition 7. *Let the graph G be the result of an ordinary step of a performance of Construction* B. *If G is* A-*constructible, then each of the graphs $G_0, G_1, G_2, \ldots, G_k$ (in the step) are likewise* A-*constructible.*

Proof. It is clear that each step of Construction A augments the number of cycles (of the constructed graph) by one. Moreover, let a performance of Construction A be given and denote the number of steps by r. Let us define a mapping γ of the set $\{1, 2, \ldots, r\}$ in the following (recursive) way:

$\gamma(1)$ is the result of the beginning step,

if $(\gamma(1), \gamma(2), \ldots, \gamma(j-1)$ are defined and$)$ we execute the j-th step of the construction, then the meaning of $\gamma(1), \gamma(2), \ldots, \gamma(j-1)$ remains the same in G as in G_0 (with the small modification that P is now substituted by the path from A to B) and $\gamma(j)$ is defined as the new cycle z (of G)[9]. It is clear that γ is a one-to-one correspondence whose range equals the family of cycles of the constructed graph.

[9] G_0, G are now used as in describing the ordinary step of Construction A.

On the other side, we can convince ourselves by analyzing the ordinary step of Construction B that whenever z is an arbitrary cycle of the constructed graph G, then z has been present in exactly one of $G_0, G_1, G_2, ..., G_k$ (if this graph is G_i with $i>0$, then apart from the change that P_i is replaced by the chain from A_i to B_i).

Let now G and some G_i $(0 \leqq i \leqq k)$ be as in the Proposition. Denote by Q_2 the application of Construction B in question (yielding G) and let Q_1 be a performance of Construction A which produces again G. Let us define the increasing sequence

$$j_1, j_2, ..., j_s$$

containing precisely those numbers j for which $\gamma(j)$ is present in G_i (γ is now defined for Q_1). We can compile a performance $Q^{(i)}$ of Construction A from the j_1-th, j_2-th, ..., ..., j_s-th steps of Q_1 (with some modifications which may be left to the reader), it is evident that $Q^{(i)}$ produces G_i. This can be done for every value of i running from 0 to k.

Having Proposition 7, the characterization of A-constructible graphs among the B-constructible ones requires still to clear up the following question:

Problem. Suppose that $G_0, G_1, G_2, ..., G_k$ are A-constructible graphs $(k \geqq 1)$. Let us apply the ordinary step of Construction B for them (with some choices of the vertices having distinguished roles in the step). Let a necessary and sufficient condition be given in order the resulting graph G be again A-constructible.

О графах удовлетворяющих некоторым условиям для циклов, II.

Пусть класс конечных ориентированных графов быть вводим следующим рекурсивным образом: (1) каждый цикл содержается в классе, (2) если G_0 — граф содержаемый в классе и мы заменяем некоторую точку степени (1, 1) графа G_0 циклом, то новый граф находится опять в классе, (3) класс является минимальным ввиду правил (1) и (2). Члены этого класса называются A-конструируемыми графами.

Эта рекурсивная процедура не даёт возможность для однозначного разложения результируемого графа. Вводится другая процедура (называема конструкцией B) так, что она допускает почти единственную декомпозицию и все A-конструируемые графы являются B-конструируемыми.

MATHEMATICAL INSTITUTE OF THE
HUNGARIAN ACADEMY OF SCIENCES
H-1053 BUDAPEST, HUNGARY
REÁLTANODA U. 13–15.
(PERMANENTLY)

MATHEMATICS DEPARTMENT OF THE
ARTS AND SCIENCE UNIVERSITY
RANGOON, BURMA
(IN A PART OF THE TIME OF
PREPARING THIS PAPER)

References

[1] Ádám, A., On some generalizations of cyclic networks, *Acta Cybernet.*, v. 1, 1971, pp. 105—119.
[2] Ádám, A., On graphs satisfying some conditions for cycles, I. *Acta Cybernet.*, v. 3, 1976, pp. 3—13.

(Received Sept. 14, 1976)

On the Bayesian approach to optimal performance of page storage hierarchies

By A. BENCZÚR, A. KRÁMLI, J. PERGEL

Introduction

In connection with the work of operating systems or interactive data base systems or large program systems of any other destination on computers with hierarchical memory arises the optimal page replacement problem. In two level storage hierarchy a reference to a page not in first level storage is called page fault. Optimal replacement algorithms minimize under different conditions the average number of page faults.

A great majority of papers devoted to this problem (see e.g. Aho et al. [1], Franaszek and Wagner [5], Easton [6]) assumes that the stochastic behaviour of the reference string is known. Therefore the algorithms proposed by them are only asymptotically optimal, when the probability distributions have to be estimated in the course of the execution of the program.

In this paper — using the Bayesian method, which first has been applied to this problem by Arató [7] — we prove in two extreme cases of loss function the optimality of the so called "least frequently used" strategy on every finite time interval for reference strings with unknown probability distribution.

Our considerations remember to the solution of the so called "two-armed bandit problem" (see Feldman [4]); we investigate the nature of the basic equation of dynamic programming (Bellman equation).

In § 1 we give the short description of the model and the formulation of the problem.

§ 1.

The program consists of n pages $1, 2, \ldots, n$, and m pages can be stored in the high speed memory and $n-m$ pages (often the whole program) are stored on a slow access memory device. The reference string $\{\eta_1, \ldots, \eta_t, \ldots\}$ from probabilistic point of view forms a sequence of independent identically distributed random variables, the common probability distribution

$$P_{i,w} = P_w(\eta_t = i)$$

of the random variables η_t depends on a parameter w, value of which is unknown. The dependence on w is given as follows: the range of parameter w is the set W of all permutations of natural numbers $1, \ldots, n$; $w(i)$ denotes the one to one mapping of set $\{1, \ldots, n\}$ realized by w. There is given a fixed decreasing sequence $p_1 > \ldots > p_n$ of probabilities $(p_1 + \ldots + p_n = 1)$ and $P_{i,w} = p_{w(i)}$.

Following the Bayesian approach to the decision theory we assume that w itself is a random variable — as we have no preliminary information about the distribution $P(\eta_t = i)$ the a priori distribution of parameter w is the uniform one.

Let us denote by $D_{t,N}$ the set of all possible sequential decision procedures $\{d_t, \ldots, d_{N-1}\}$ on a finite time interval $[t, N]$. A decision $d_{t'}$, which depends only on the initial decision d_0 and the observed reference string $\{\eta_1, \ldots, \eta_{t'}\}$ $(t' \in [t, N])$ means the subset of pages being absent of the central memory after the observation of string $\{\eta_1, \ldots, \eta_{t'}\}$.

The decision d_t consists of $n-m$ elements. By Arató's model (case A) the memory can be rearranged without extra cost before each reference η_t, but a page-fault $(\eta_t \in d_{t-1})$ increases the cost by 1 unity; i.e. the loss function has the following form

$$X_t^{d_{t-1}} = \begin{cases} 1 & \text{if } \eta_t \in d_{t-1}, \\ 0 & \text{otherwise.} \end{cases} \tag{1}$$

In this paper there is investigated another extreme case (case B) too: each change of a page increases the cost by 1 unity and η_t always must be stored in the central memory; i.e., the loss function has the following form

$$X_t^{d_t, d_{t-1}} = |d_t \setminus d_{t-1}|, \tag{2}$$

where $|.|$ denotes the number of elements of a finite set. (Notice that if $\eta_t \in d_{t-1}$, then $X_t^{d_t, d_{t-1}} \geqq 1$.)

§ 2. (Case A)

Our aim is to find the set of sequential decision procedures $\{d_0 \ldots d_{N-1}\}$ which minimize the risk function

$$E\left(\sum_{t=1}^{N} X_t^{d_{t-1}}\right)$$

(E is the expectation taken on the basis of the a priori distribution of w.) In the sequel y_t denotes the fixed value of η_t.

Let

$$v(y_1, \ldots, y_t, N-t) = \min_{\{d_t, \ldots, d_{N-1}\} \in D_{t,N}} E_{y_1, \ldots, y_t}\left(\sum_{\tau=t+1}^{N} X_\tau^{d_{\tau-1}}\right), \tag{3}$$

where $E_{y_1, \ldots, y_t}$ denotes the conditional expectation under a given string $\{y_1, \ldots, y_t\}$. The class of functions $v(y_1, \ldots, y_t, N-t)$ satisfies the Bellman equation (see e.g. [2])

$$v(y_1, \ldots, y_{t-1}, N-t+1) = \min_{d_{t-1}} E_{y_1, \ldots, y_{t-1}}\left(X_t^{d_{t-1}} + v(y_1, \ldots, y_{t-1}, \eta_t, N-t)\right). \tag{4}$$

Solving it recursively we can find the set of optimal strategies. Notice that $v(y_1, \ldots, y_t, N-t)$ does not depend on d_{t-1}, therefore, it is sufficient to minimize for every t the conditional expectation

$$E_{y_1, \ldots, y_{t-1}}(X_t^{d_{t-1}}).$$

We shall prove that the optimal strategies are those for which d_0 is arbitrarily chosen and d_t consists of the $n-m$ least frequently occured pages in the string $\{y_1, \ldots, y_t\}$. Before this we prove a lemma and two corollaries of it.

Lemma 1. Let us suppose that the frequency f_i of the page i in the string $\{y_1, \ldots, y_t\}$ is less than the frequency f_j of the page j. Let w_1 and w_2 be two permutations of natural numbers $1, \ldots, n$, and $k_1 < k_2 \leqq n$ two natural numbers.

If

$$\begin{gathered} w_1(i) = k_1, \quad w_1(j) = k_2, \\ w_2(i) = k_2, \quad w_2(j) = k_1, \\ w_1(k) = w_2(k) \quad \text{for every} \quad k \neq i, j, \end{gathered} \tag{$*$}$$

then

$$P(w_1|y_1, \ldots, y_t) < P(w_2|y_1, \ldots, y_t).$$

Proof. On the basis of Bayes' theorem

$$P(w_1|y_1, \ldots, y_t) = \frac{\prod_{k=1}^{n} p_{w_1(k)}^{f_k}}{\sum_{w \in W} \prod_{k=1}^{n} p_{w(k)}^{f_k}}. \tag{5}$$

Similarly,

$$P(w_2|y_1, \ldots, y_t) = \frac{\prod_{k=1}^{n} p_{w_2(k)}^{f_k}}{\sum_{w \in W} \prod_{k=1}^{n} p_{w(k)}^{f_k}}. \tag{6}$$

The assertion of our Lemma can be obtained by comparison of (5) and (6) using the inequality $p_{k_2} < p_{k_1}$.

Corollary 1. If $\{y_{t+1}, \ldots, y_{t+\tau}\}$, $\{\bar{y}_{t+1}, \ldots, \bar{y}_{t+\tau}\}$ are two strings (sequences of pages) i, j are two pages with the following properties, for every $1 \leqq \tau' \leqq \tau$

(i) if $y_{t+\tau'} \neq i, j$, then $\bar{y}_{t+\tau'} = y_{t+\tau'}$,

(ii) if $y_{t+\tau'} = i$, then $\bar{y}_{t+\tau'} = j$,

(iii) if $y_{t+\tau'} = j$, then $\bar{y}_{t+\tau'} = i$,

(iv) the frequency f_i of the page i in the string $\{y_1, \ldots, y_t\}$ is less, than the frequency f_j of the page j,

(v) if the frequency of the page j in the string $\{y_{t+1}, \ldots, y_{t+\tau}\}$ is greater than the frequency of page i, then

$$P(\eta_{t+1} = y_{t+1}, \ldots, \eta_{t+\tau} = y_{t+\tau}|y_1, \ldots, y_t) > P(\eta_{t+1} = \bar{y}_{t+1}, \ldots, \eta_{t+\tau} = \bar{y}_{t+\tau}|y_1, \ldots, y_t). \tag{7}$$

Proof. The set W can be decomposed into the union of $\frac{n!}{2}$ disjoint pairs of permutations $\{w_1, w_2\}$ of $(\ast)$ property figuring in Lemma 1. Inequality (7) can be obtained by direct comparison applying Lemma 1 to every such pair.

Remark 1. Corollary 1 means for $\tau=1$ that the order of aposteriori probabilities of the pages after having observed the string $\{y_1, \dots, y_t\}$ is the same as the order of their frequencies in this string.

Corollary 2. Let $\{y_{t+1}, \dots, y_{t+\tau}\}$ and $\{\bar{y}_{t+1}, \dots, \bar{y}_{t+\tau}\}$ be the same strings and i, j the same pages as in Corollary 1. Let A and B be two events of the algebra geneiated by random variables $\eta_{t+\tau+1}, \dots, \eta_N$, which are invariant under the changing of i and j; let i' and j' be two pages different from i and j. If

$$A \setminus B = \{\eta_{t+\tau''} = i'\}, \quad B \setminus A = \{\eta_{t+\tau''} = j'\}$$

for a suitable τ'', then

$$\begin{aligned} &|P(\eta_{t+1} = y_{t+1}, \dots, \eta_{t+\tau} = y_{t+\tau}, A | y_1, \dots, y_t) - \\ &- P(\eta_{t+1} = y_{t+1}, \dots, \eta_{t+\tau} = y_{t+\tau}, B | y_1, \dots, y_t)| > \\ &> |P(\eta_{t+1} = \bar{y}_{t+1}, \dots, \eta_{t+\tau} = \bar{y}_{t+\tau}, A | y_1, \dots, y_t) - \\ &- P(\eta_{t+1} = \bar{y}_{t+1}, \dots, \eta_{t+\tau} = \bar{y}_{t+\tau}, B | y_1, \dots, y_t)|. \end{aligned} \tag{8}$$

The proof is analogous to the proof of Lemma 1 and Corollary 1. Inequality (8) can be obtained by comparison of conditional probabilities for every quadruple $\{w_1, w_2, w_3, w_4\}$ of permutations of the following property.

If $k_1, k_2, k_3, k_4 \leqq n$ are 4 different natural numbers, then

$$\begin{aligned} &w_1(i) = k_1, \quad w_1(j) = k_2, \quad w_1(i') = k_3, \quad w_1(j') = k_4, \\ &w_2(i) = k_2, \quad w_2(j) = k_1, \quad w_2(i') = k_4, \quad w_2(j') = k_3, \\ &w_3(i) = k_1, \quad w_3(j) = k_2, \quad w_3(i') = k_3, \quad w_3(j') = k_4, \\ &w_4(i) = k_2, \quad w_4(j) = k_1, \quad w_4(i') = k_4, \quad w_4(j') = k_3, \end{aligned}$$

and

$$w_1(k) = w_2(k) = w_3(k) = w_4(k) \quad \text{for every} \quad k \neq k_1, k_2, k_3, k_4.$$

Theorem 1. The set of sequential decision procedures $\{d_0, \dots, d_{N-1}\}$ which minimize the expected loss $E\left(\sum_{t=1}^{N} X_t^{d_{t-1}}\right)$ in case A consists of the so called least frequently used (LFU) strategies, i.e. d_0 is arbitrary, and for every t, d_t consists of the first $n-m$ least frequently used pages in the string $\{y_1, \dots, y_t\}$.

Proof. Theorem 1 is a straightforward consequence of Remark 1 and the uniformity of the a priori distribution of parameter w.

§ 3. (Case B)

In case B form we follow the method of comparing the expected cost of optimal continuations of different decisions d_t and d_t' after having observed the string $\{y_1, ..., y_t\}$. Therefore, we denote by $v(y_1, ..., y_t, d_t, N-t)$ the risk-function belonging to the observations $y_1, ..., y_t$ the state d_t of memory at time t and the optimal strategy on the time interval $[t+1, N]$, i.e.,

$$v(y_1, ..., y_t, d_t, N-t) = \min_{\{d_{t+1}, ..., d_{N-1}\} \in D_{t+1, N}} E_{y_1, ..., y_t}\left(\sum_{\tau=t+1}^{N} X_\tau^{d_\tau, d_{\tau-1}}\right).$$

Our aim is to determine the set of sequential decision procedures $\{d_0, ..., d_{N-1}\}$ for which

$$\min_{\{d_0, ..., d_{N-1}\} \in D_{0, N}} E\left(\sum_{t=1}^{N} X_t^{d_t, d_{t-1}}\right) = \min_{d_0} v(d_0, N)$$

is reached.

First we prove a lemma, which restricts the set of possible strategies to the so called demand paging algorithms.

Lemma 2. If $\eta_t \notin d_{t-1}$; d_t and d_t' are two different decisions of properties

(*i*) $d_t = d_{t-1}$,

(*ii*) $|d_t' \setminus d_t| = l > 0$,

then

$$v(y_1, ..., y_t, d_t, N-t) \leqq l + v(y_1, ..., y_t, d_t', N-t). \tag{9}$$

Proof. There are 4 possible cases

$$\eta_{t+1} \in d_t' \setminus d_t,$$
$$\eta_{t+1} \in d_t \setminus d_t',$$
$$\eta_{t+1} \in d_t \cap d_t',$$
$$\eta_{t+1} \in \{1, ..., n\} \setminus (d_t \cup d_t').$$

In every case it is easy to show that for an arbitrary decision d_{t+1}' the decision d_{t+1} can be chosen to be equal to d_{t+1}' paying at most l extra cost.

Remark 2. A similar assertion can be verified for $\eta_t \in d_{t-1}$. If d_t and d_t' are two different decisions with properties

(*iii*) $|d_t \setminus d_{t-1}| = 1$,

(*iv*) $|d_t' \setminus d_{t-1}| = l$,

(*v*) $|d_t' \setminus d_t| = l-1$,

then

$$v(y_1, ..., y_t, d_t, N-t) \leqq v(y_1, ..., y_t, d_t', N-t) + l - 1. \tag{10}$$

The Bellman equation for the risk-functions has the form

$$v(y_1, \dots, y_{t-1}, d_{t-1}, N-t+1) =$$

$$= \min_{d_t} E_{y_1, \dots, y_{t-1}} (X_t^{d_t, d_{t-1}} + v(y_1, \dots, y_{t-1}, \eta_t, d_t, N-t)). \tag{11}$$

The relations (9) and (10), applying recursively equation (11), show that the optimal sequential decision procedures are among those which fulfil the conditions

$$d_t = d_{t-1} \quad \text{if} \quad \eta_t \in d_{t-1}$$

and

$$d_{t-1} \setminus d_t = \{\eta_t\} \quad \text{if} \quad \eta_t \in d_{t-1}. \tag{12}$$

The decision procedures of the above type are called "demand paging algorithms" (see e.g. Denning [3]).

We can deduce from the following theorem that the LFU strategies minimize the expected loss in case B, too.

Theorem 2. If d_t and d_t' are two different decisions for which

$$d_t \setminus d_t' = \{i\}, \quad d_t' \setminus d_t = \{j\}$$

and the frequency f_i of the page i in the string $\{y_1, \dots, y_t\}$ is less than the frequency f_j of the page j, then

$$v(y_1, \dots, y_t, d_t, N-t) < v(y_1, \dots, y_t, d_t', N-t). \tag{13}$$

Proof. The proof can be carried out by induction on $\theta = N-t$. If $\theta = 1$, then the assertion of Theorem 2 is an obvious consequence of Corollary 1. Applying the induction hypothesis for $\theta = 1, \dots, \theta = N-1-t$ we get that the optimal decisions in every case $\eta_t \in d_{t-1}$ are those for which $d_t \setminus d_{t-1}$ is one of the least frequently used (in the string $\{y_1, \dots, y_t\}$) pages of the admissible set $\{1, \dots, n\} \setminus d_{t-1}$.

To demonstrate the main idea of the proof of the induction step, first we briefly present it in the case $n=3$; $m=2$. Then $d_t = \{i\}$, $d_t' = \{j\}$. Let us denote by k the third page. If $f_k \geqq f_i$ then, using the induction hypothesis, it is easy to prove that for arbitrary outcome y_{t+1} of η_{t+1},

$$v(y_1, \dots, y_{t+1}, d_{t+1}, N-t-1) \leqq v(y_1, \dots, y_{t+1}, d_{t+1}', N-t-1), \tag{14}$$

where $d_{t+1}(d_{t+1}')$ is the optimal continuation of decision $d_t(d_t')$.
But

$$E_{y_1, \dots, y_t} (X_{t+1}^{d_{t+1}, d_t}) \leqq E_{y_1, \dots, y_t} (X_{t+1}^{d_{t+1}', d_t'})$$

by Corollary 1, thus using the equation (11) (Bellman equation) we get inequality (13).

If $f_k < f_i$, then in the case $\eta_{t+1}=j$ inequality (14) fails, therefore, we need to analyze the strings of the form

$$(a)\ \eta_{t+1}=i, \dots, \eta_{t+\tau'-1}=i, \quad \eta_{t+\tau'}=k,$$

$$(b)\ \eta_{t+1}=i, \dots, \eta_{t+\tau-1}=i, \quad \eta_{t+\tau}=j,$$

$$(a')\ \eta_{t+1}=j, \dots, \eta_{t+\tau'-1}=j, \quad \eta_{t+\tau'}=k,$$

$$(b')\ \eta_{t+1}=j, \dots, \eta_{t+\tau-1}=j, \quad \eta_{t+\tau}=i.$$

Using the same arguments as in case $f_k \geqq f_i$ it is easy to show that after having observed a string of type (*a*), (*a'*), (*b*) or (*b'*) the optimal continuation of decision d_t has a better or equal optimal continuation than those of decision d_t'. For cases (*a*) and (*a'*) the increments of conditional risks for the optimal continuations of the decisions d_t and d_t' can be compared using Corollary 1.

For the cases (*b*) and (*b'*) we have to prove the inequality

$$\begin{gathered} P(\eta_{t+1}=i|y_1, \dots, y_t)+P(\eta_{t+1}=j, \eta_{t+2}=i|y_1, \dots, y_t)+\dots \\ +P(\eta_{t+1}=j, \dots, \eta_{N-1}=j, \eta_N=i|y_1, \dots, y_t) \leqq \\ \leqq P(\eta_{t+1}=j|y_1, \dots, y_t)+P(\eta_{t+1}=i, \eta_{t+2}=j|y_1, \dots, y_t)+\dots \\ +P(\eta_{t+1}=i, \dots, \eta_{N-1}=i, \eta_N=j|y_1, \dots, y_t). \end{gathered} \tag{15}$$

Inequality (15) can be verified using Corollary 1 and the obvious relation

$$\begin{gathered} P(\eta_{t+1}=i|y_1, \dots, y_t)+P(\eta_{t+1}=j, \eta_{t+2}=i|y_1, \dots, y_t)+\dots+ \\ +P(\eta_{t+1}=j, \eta_{t+2}\neq i, \dots, \eta_{t+\tau-1}\neq i, \eta_{t+\tau}=i|y_1, \dots, y_t)+\dots= \\ =P(\eta_{t+1}=i|y_1, \dots, y_t)+P(\eta_{t+1}=j|y_1, \dots, y_t)= \\ =P(\eta_{t+1}=j|y_1, \dots, y_t)+P(\eta_{t+1}=i, \eta_{t+2}=j|y_1, \dots, y_t)+\dots+ \\ +P(\eta_{t+1}=i, \eta_{t+2}\neq j, \dots, \eta_{t+\tau-1}\neq j, \eta_{t+\tau}=j|y_1, \dots, y_t)+\dots \end{gathered} \tag{16}$$

as (15) can be obtained from (16) by leaving pairs on term from the left hand side the other from the right hand side so that in each pair the left hand side term is greater.

Next we give the proof of the general case. Let us denote by I the subset of pages from the set $\{1, \dots, n\}\setminus d_t'$ with less than f_i frequency in the string $\{y_1, \dots, y_t\}$. If $|I|=0$, then the proof is analogous to that of case $f_k \geqq f_i$ for $n=3$.

When $\eta_{t+1}\neq i, j$, then $X_{t+1}^{d_{t+1}, d_t}=X_{t+1}^{d_{t+1}', d_t'}$ and

$$v(y_1, \dots, y_{t+1}, d_{t+1}, N-t-1) \leqq v(y_1, \dots, y_{t+1}, d_{t+1}', N-t-1) \tag{17}$$

by the induction hypothesis.

For $\eta_{t+1}=i$,

$$X_{t+1}^{d_{t+1}, d_t}=1, \quad X_{t+1}^{d_{t+1}', d_t'}=0 \tag{18}$$

and

$$v(y_1, \dots, y_t, i, d_{t+1}, N-t-1) < v(y_1, \dots, y_t, i, d_{t+1}', N-t-1). \tag{19}$$

Similarly, for $\eta_{t+1}=j$,

$$X_{t+1}^{d_{t+1},\,d_t} = 0, \quad X_{t+1}^{d'_{t+1},d'_t} = 1 \tag{20}$$

and by condition $|I|=0$

$$v(y_1, \ldots, y_t, j, d_{t+1}, N-t-1) = v(y_1, \ldots, y_t, j, d'_{t+1}, N-t-1). \tag{21}$$

By Corollary 1, from (18) and (20) follows the inequality

$$E_{y_1, \ldots, y_t}(X_{t+1}^{d_{t+1},\,d_t}) < E_{y_1, \ldots, y_t}(X_{t+1}^{d'_{t+1},d'_t}),$$

which together with (17), (19), (21) and the Bellman equation proves the assertion of Theorem 2 in case $|I|=0$.

Let us assume that $|I|\neq 0$, and introduce the mapping Φ on the set of strings $\{y_{t+1}, \ldots, y_{t+\tau}\}$ of length τ (τ is an arbitrary natural number, and $y_{t+\tau}\in\{1, \ldots, n\}$) as follows

$$\{\bar{y}_{t+1}, \ldots, \bar{y}_{t+\tau}\} = \Phi(\{y_{t+1}, \ldots, y_{t+\tau}\})$$

and for every $1\leqq\tau'\leqq\tau$,

$$\begin{aligned} \bar{y}_{t+\tau'} &= y_{t+\tau'} \quad \text{if} \quad y_{t+\tau'} \neq i, j, \\ \bar{y}_{t+\tau'} &= j \quad \text{if} \quad y_{t+\tau'} = j, \\ \bar{y}_{t+\tau'} &= i \quad \text{if} \quad y_{t+\tau'} = j. \end{aligned}$$

Let us investigate the behaviour of the optimal continuations of decisions d_t and d'_t on the strings of the form

$$\{y_{t+1} = j, y_{t+2} \neq i, \ldots, y_{t+\tau-1} \neq i, y_{t+\tau} = i\}.$$

There exists a $\tau''>1$, such that for $\tau'\leqq\tau''$ the following relations are valid

(i) $d_{t+\tau'}=d_{t+\tau'-1}$ or the unique element of $d_{t+\tau'}\setminus d_{t+\tau'-1}$ has less frequency in the string $\{y_1, \ldots, y_{t+\tau'}\}$ than the page i.

(ii) $d'_{t+\tau'}=d'_{t+\tau'-1}$ or the unique element of $d'_{t+\tau'}\setminus d'_{t+\tau'-1}$ has less frequency in the string $\{y_1, \ldots, y_{t+\tau'}\}$ than the page i.

(iii) $d_{t+\tau'}\setminus d'_{t+\tau'} = \{i\}$,

(iv) $d'_{t+\tau'}\setminus d_{t+\tau'} = \{k_1^*\}$

and there is at most one k_2^* in the set $\{1, \ldots, n\}\setminus d'_{t+\tau}$ which has less frequency in the string $\{y_1, \ldots, y_{t+\tau'}\}$ than the page k_1^*. The properties (*i*), (*ii*) and (*iii*) are obvious consequences of the induction hypothesis, the property (*iv*) can be proved by induction on τ'.

If the first moment τ' for which property (*ii*) fails, is less than τ, then

$$d'_{t+\tau'}\setminus d'_{t+\tau'-1} = \{i\} \quad \text{i.e.} \quad d'_{t+\tau'} = d'_{t+\tau}. \tag{22}$$

(Notice, that for such a τ' property (*i*) is still valid.)

Therefore,

$$v(y_1, \ldots, y_{t+\tau'}, d_{t+\tau'}, N-t-\tau') = v(y_1, \ldots, y_{t+\tau'}, d'_{t+\tau'}, N-t-\tau'). \tag{23}$$

If there is no such $\tau'<\tau$ for which property (*ii*) fails, and

$$d'_{t+\tau-1}\setminus d_{t+\tau-1} = \{k_1^*\}$$

has minimal frequency among the pages $\{1, \ldots, n\}\setminus d'_{t+\tau'}$, then it follows from the relation

$$y_{t+\tau} = i \quad \text{that} \quad d'_{t+\tau} = d_{t+\tau},$$

i.e.,

$$v(y_1, \ldots, y_{t+\tau}, d_{t+\tau}, N-t-\tau) = v(y_1, \ldots, y_{t+\tau}, d'_{t+\tau}, N-t-\tau).$$

If there exists a unique page k_2^* in the set $\{1, \ldots, n\}\setminus d'_{t+\tau}$ with less frequency than k_1^*, i.e.,

$$d'_{t+\tau}\setminus d_{t+\tau} = \{k_2^*\}, \quad d_{t+\tau}\setminus d'_{t+\tau} = \{k_1^*\}, \tag{24}$$

then we have to argue more carefully, which we shall do after having analyzed the behaviour of optimal continuations of decisions d_t and d'_t on the strings of type

$$\Phi(\{y_{t+1}=j, y_{t+2}\neq i, \ldots, y_{t+\tau-1}\neq i, y_{t+\tau}=i\}).$$

Let us denote by $\bar d_{t+\tau}$ and $\bar d'_{t+\tau}$ the optimal continuations of decisions d_t and d'_t on the string

$$\Phi(\{y_{t+1}, \ldots, y_{t+\tau}\}).$$

If for a $\tau'<\tau$, $d_{t+\tau'}$ and $d'_{t+\tau'}$ fulfil the conditions (*i*) and (*ii*) on the string $\{y_{t+1}, \ldots, y_{t+\tau'}\}$, then so do $\bar d_{t+\tau'}$ and $\bar d'_{t+\tau'}$. Moreover,

(*v*) $d'_{t+\tau'} = \bar d_{t+\tau'}$,

(*vi*) $\bar d'_{t+\tau'}\setminus d_{t+\tau'} = \{j\}$,

(*vii*) $d_{t+\tau'}\setminus \bar d'_{t+\tau'} = \{i\}$.

Obviously for $\eta_{t+1}=i$, $\bar d_{t+1}$ and $\bar d'_{t+1}$ satisfy the relation

$$v(y_1, \ldots, y_t, i, \bar d_{t+1}, N-t-1) < v(y_1, \ldots, y_t, i, \bar d'_{t+1}, N-t-1).$$

But this inequality is insufficient for the proof of Theorem 2, as we have to ballance the difference in expected loss between the continuation of decisions d_t and d'_t on the strings of type

$$\{y_{t+1}=j,\ y_{t+2}\neq i, \ldots, y_{t+\tau-1}\neq i,\ y_{t+\tau}=i\}.$$

By properties (*i*)—(*vii*), the symmetry of the mapping Φ and Corollary 1 the sum of expected loss of continuations of decisions d_t and $\bar d_t$ on the strings $\{y_{t+1}, \ldots, y_{t+\tau}\}$ and $\Phi(\{y_{t+1}, \ldots, y_{t+\tau}\})$ is less than those of decisions d'_t and $\bar d'_t$.

The comparison of probabilities of page faults at the moment τ caused by decisions $d_{t+\tau-1}$ and $\bar d'_{t+\tau-1}$ can be carried out analogously to the case $n=3$, using the identity (15). It remains to analyze the case when there exists a page k_2^* in the set $\{1, \ldots, n\}\setminus d'_{t+\tau}$ with less frequency than k_1^*. By the symmetry of the mapping Φ, for the string $\Phi(\{y_{t+1}, \ldots, y_{t+\tau}\})$,

$$\bar{d}_{t+\tau} \setminus \bar{d}'_{t+\tau} = \{k_2^*\}$$

and

$$\bar{d}'_{t+\tau} \setminus \bar{d}_{t+\tau} = \{k_1^*\} \tag{25}$$

(See properties (*iv*), (*v*) and relation (24).)

Let $\tau'' > \tau$ the moment of first page-fault caused by decision $d_{t+\tau}$ or $d'_{t+\tau}$ (respectively by $\bar{d}'_{t+\tau}$ or $\bar{d}_{t+\tau}$) on the string $\{y_{t+1}, \dots, y_{t+\tau}, y_{t+\tau+1}, \dots, y_N\}$ (respectively on $\{\Phi(\{y_{t+1}, \dots, y_{t+\tau}\}), y_{t+\tau+1}, \dots, y_N\}$). Since k_2^* and k_1^* are the first two pages of the set

$$\{1, \dots, n\} \setminus (d_{t+\tau} \cap d'_{t+\tau}) = \{1, \dots, n\} \setminus (\bar{d}_{t+\tau} \cap \bar{d}'_{t+\tau})$$

least frequently used in the string $\{y_1, \dots, y_t, y_{t+1}, \dots, y_{t+\tau}\}$ (respectively $\{y_1, \dots, y_t, \Phi(\{y_{t+1}, \dots, y_{t+\tau}\})\}$ thus we get

$$v(y_1, \dots, y_{t+\tau''}, d_{t+\tau''}, N-t-\tau'') = v(y_1, \dots, y_{t+\tau''}, d'_{t+\tau''}, N-t-\tau''), \tag{26}$$

$$v(y_1, \dots, y_t, \Phi(\{y_{t+1}, \dots, y_{t+\tau}\}), y_{t+\tau+1}, \dots, y_{t+\tau''}, \bar{d}_{t+\tau''}, N-t-\tau'') =$$
$$= v(y_1, \dots, y_t, \Phi(\{y_{t+1}, \dots, y_{t+\tau}\}), y_{t+\tau+1}, \dots, y_{t+\tau''}, \bar{d}'_{t+\tau''}, N-t-\tau''). \tag{27}$$

If the page fault was caused by an event of type

$$\eta_{t+\tau''} \in d_{t+\tau} \cap d'_{t+\tau},$$

then the expected loss of the strategy $\{\dots d'_t, \dots, d'_{t+\tau}, \dots\}$ is greater than the loss of the other one. In the opposite case we can compare the common expected loss of the strategies

$$\{\dots d'_t, \dots, d'_{t+\tau}, \dots\} \quad \text{and} \quad \{\dots \bar{d}'_t, \dots, \bar{d}'_{t+\tau}, \dots\}$$

(respectively

$$\{\dots d_t, \dots, d_{t+\tau}, \dots\} \quad \text{and} \quad \{\dots \bar{d}_t, \dots, \bar{d}_{t+\tau}, \dots\})$$

using Corollary 2, and we get that the former is greater. This last remark together with relations (26) and (27) completes the proof of Theorem 2.

Remark 3. Also in case **B** the decision d_0 can be arbitrarily chosen by the symmetry of the a priori distribution of the parameter w.

Remark 4. Our all considerations remain valid for any a priori distribution in the space of all probability distributions invariant under the permutations of pages.

Abstract

Using the sequential Bayesian method the authors prove that in a two level storage hierarchy the "least frequently used" strategy is optimal for the page fault rate. It is assumed that the reference string forms a sequence of independent identically distributed random variables with unknown distribution. Two kind of loss function is discussed.

References

[1] Aho, A. V., P. J. Denning, J. D. Ullmann, Principles of optimal page replacement, *J. Assoc. Comput. Mach.*, v. 18, 1971, pp. 80—93.
[2] De Groot, M. H., *Optimal statistical decisions*, Mc Graw—Hill, N. Y., 1970.
[3] Denning, P. J., Virtual memory, *Comput. Surveys*, v. 2, 1970, pp. 153—189.
[4] Feldman, D., Contributions to the "two-armed bandit" problem, *Ann. Math. Statist.*, v. 33, 1962, pp. 847—856.
[5] Franaszek, P. A., T. J. Wagner, Some distribution-free aspects of paging algorithm performance, *J. Assoc. Comput. Mach.*, v. 21, 197 , pp. 31—39.
[6] Easton, M. C., Model for interactive data base reference string, *IBM J. Res. Develop.*, v. 19, 1975, pp. 550—556.
[7] Arató, M., A note on optimal performance of page storage, *Acta Cybernet.*, v.3, 1976, pp. 25—30.

(Received March 11, 1976)

Deadlock problems of dynamic memory allocation on minicomputers with multilevel interrupt system

By J. SOMOGYI

1. Introduction

There have been a number of papers in the last decade dealing with control of concurrent processes (see [1] for a comprehensive list of papers). In this paper we investigate the applicability of some of the theoretical results to minicomputers with multilevel interrupt system.

In view of concurrent processes the main characteristics of these machines can be summarised as follows (see Fig. 1). Let i, j and k denote three interrupt levels such that $i<j<k$, and suppose that at time t_0 the machine works on level i. At time t_1 a level k interrupt request arrives. Because of $k>i$ the hardware saves the context (program counter, indicators, etc.) of level i, and loads the context of level k into the appropriate hardware registers. Then program execution goes on at the memory address pointed by the new program counter.

At time t_2 a level j interrupt request arrives. Because of $j<k$ (the current level) the request is recorded by the hardware, but not dealt with. At time t_3 level k completes. At this time level j is the highest waiting level. Therefore the hardware selects (via context changing) level j for execution. At time t_4 level j completes. Then the hardware returns to level i interrupted at time t_1.

In section 2 we shall describe a deadlock problem related to the monitor program of the VT1005, a Hungarian manufactured minicomputer with an interrupt system described before. The design and development of the monitor program was

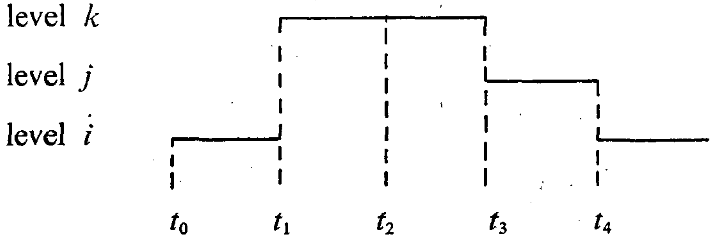


Fig. 1

Hardware scheduling of interrupt levels

performed when the complete specification of the machine had not been freezed by the manufacturer. Therefore, instead of using a simulator the monitor was written in a higher level language, CDL [2], and was debugged on another minicomputer, R10, which is equivalent to CII Mitra 15.

From programmer's point of view the interrupt system of the two minicomputers are identical, so the ascertainments of this paper apply to both machines.

In section 3 two solutions of the deadlock problem will be described and compared in view of their memory requirement. Section 4 is devoted to the implementation details. In section 5 we prove that the solution described in section 4 is deadlock free.

2. The deadlock problem

Suppose that in Fig. 1 the program executed on level i calls for a monitor service (e.g. ASCII—EBCDIC conversion), and at time t_1 the control is in the service routine. At the same time another program on level k enters, and calls for the same monitor service. Then there are two possibilities, either queuing the second and the possible further requests, or writing re-entrant service routines.

In the first case the service routines become resources, each forming a separate resource type. Moreover the service routines may call for further service routines, etc. Avoiding deadlocks so, the deadlock avoidance may become overcomplicated for the limited memory of a minicomputer.

In the second case we have to provide dynamically allocated working areas for the service routines, with memory being the only resource type to be dealt with. Further simplification can be introduced by allocating the memory in blocks of a fixed size. Though a CDL procedure is available handling variable size blocks, it does not fit 8K byte memory of our machine [3].

After all in our system there is a common memory consisting of fixed size blocks, and one CDL procedure can have one such memory block. Suppose that the programs running on interrupt levels greater than zero are all peripheral device handlers. (There is no interrupt associated with level zero, this level is reserved for user programs.) Then, in best case an interrupt level requires two memory blocks, as shown in Fig. 2.

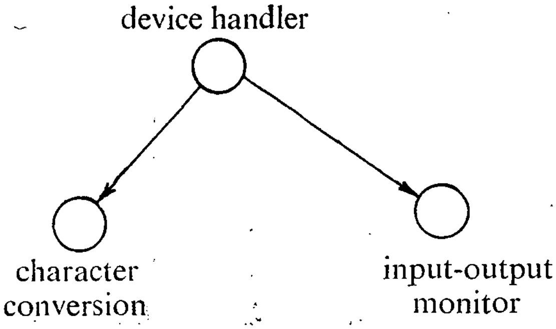


Fig. 2
The best case

In the figure CDL procedures are represented by circles. The procedures within the same row are executed one by one. Therefore the number of rows is equal to the depth of nesting, that is to the maximum number of blocks required by the interrupt level.

The worst case is caused by device errors requiring operator intervention (e.g. card jam, paper low). In this case the input-output monitor is called for sending the appropriate message to the operator's console. Then, as Fig. 3 shows, the interrupt level requires five memory blocks.

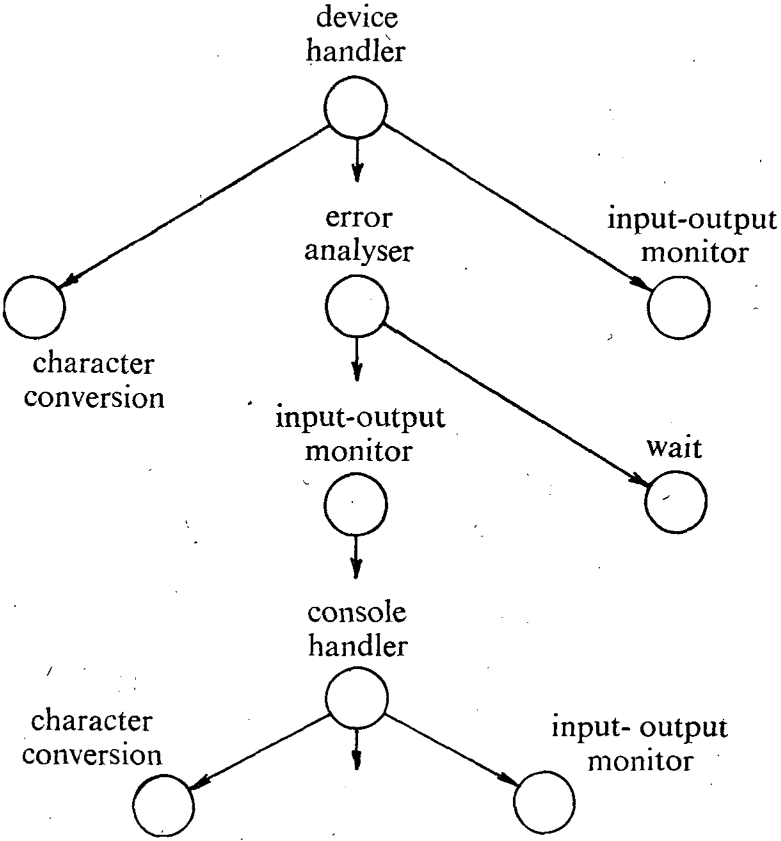


Fig. 3
The worst case

Suppose now that some of the interrupt levels have memory, but none of them has enough to complete, and the common memory has been exhausted. Then the system contains a deadlock.

3. The solution of the deadlock problem

The maximum memory requirement of each interrupt level is given. Therefore the deadlock could be prevented by the known methods [5]. Unfortunately the application of these methods to the machines described before, implies a serious efficiency problem.

When occuring an interrupt, the service must start immediately so as to avoid the loss of data or status informations. Doing so, we need at least one memory block. Therefore, if we decide to postpone the service of the interrupt for avoiding the deadlock, we must do it before starting the input-output operation, that is, the operation must not be started. However, this implies significant loss of time, because the input-output operation could take place during the waiting time.

Because of the small memory size of the VT1005, we are forced to look for a solution as simple as possible. In the following we shall compare two simple solutions in view of their memory requirement.

We are interested only in the differences of the two solutions, so we ignore the memory requirement of the user program, which is the same in the two cases.

3.1. The trivial solution. Let the size of the common memory be large enough to satisfy the memory requirement of each interrupt level simultaneously even in the worst case.

Calculating the total memory requirement we have to take into consideration, that the operator's console can service one request at a time. The other requests are queued by the input-output monitor. Therefore only one of the interrupt levels can have the maximum number of blocks, the others can require one less.

Let M denote the maximum number of blocks required by an interrupt level, let B be the length of one block, and let N be the number of interrupt levels active at a time. Then the total memory requirement is $M\,B+(N-1)(M-1)\,B$.

3.2. A nontrivial solution. Let the size of the common memory be such that the minimal memory requirement of each interrupt level could be satisfied simultaneously, and one of them could have the maximal requirement. Moreover only one of the interrupt levels at a time can have memory exceeding the minimal requirement. As a matter of fact, it would be good enough to grant one memory block per interrupt level for starting the interrupt service routine. However, we want to avoid the unnecessary suspension of levels when the input-output operation was error free.

Let M, N, B be as before, let m denote the minimum number of blocks required by an interrupt level, and let C be the size of code necessary for controlling the memory allocation according to the present solution. Then the total memory requirement is $NmB+(M-m)B+C$.

The nontrivial solution has an advantage over the trivial one, if

$$MB+(N-1)(M-1)B > NmB+(M-m)B+C$$

and, therefore, if

$$N > 1+\frac{C}{(M-m-1)B}.$$

4. The implementation

The common memory consists of $Nm+M-m$ blocks, $M-m$ of which are subject to mutual exclusion. The mutual exclusion is implemented via the enqueue and remove primitives defined below.

Let q denote the waiting queue, consisting of the total number of interrupt levels plus one element, let p be the pointer of the queue, and let n be the actual interrupt level. Then

enqueue (n):

```
L:  $p = 0$ —— $q[0] := n,$   $p := n$ —— return
     |
    $p \neq 0$ —— $q[p] := n,$   $p := n$ —— DIT —— goto $L$
```

where DIT stands for Desactivate InTerrupt. This allows to continue the execution of other interrupt levels with lower priorities.

remove (n):

```
    $p = n$ —— $p := 0$ ———— return   (no levels are waiting)
     |
    $p \neq n$ —— $q[0] := q[n]$ —— PIT   $q[0]$ —— return
```

where PIT stands for Programmed InTerrupt. This activates the interrupt level desactivated by the DIT operation. The execution of the activated level will continue in the enqueue primitive just behind the DIT operation.

Dealing with a single-processor system, the primitives are implemented by interrupt inhibition.

For allocation and deallocation of memory blocks, let R denote a list consisting of the total number of interrupt levels plus one element, and let n be the level requesting or releasing a memory block.

The allocation procedure:

```
$R[n] := R[n]+1$ —— $R[n] \neq m+1$ —— allocate memory
                          |
                    $R[n] = m+1$ —— enqueue ($n$) —— allocate memory
```

The deallocation procedure:

```
deallocate memory —— $R[n] := R[n]-1$ —— $R[n] \neq m$ —— exit
                                              |
                                        $R[n] = m$ —— remove ($n$).
```

Returning to the question of choosing the one of the two solutions to be advantaged our numerical results are: $C=160$ bytes, $B=32$ bytes, $M=5$ and $m=2$, therefore $N \geqq 4$. Hence the second solution needs less memory, if at least four of the peripheral devices are expected to work simultaneously.

5. Proof of the nontrivial solution being deadlock free

For formal treatment of the deadlock problem we adopt the definition of [1] with the number of resource types equal to one. To begin with we convert the description of the interrupt servicing procedure into a chain. Fig. 4 shows the chain corresponding to Fig. 2.

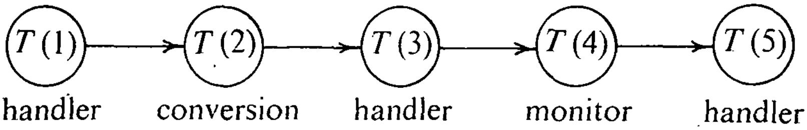


Fig. 4

Chain corresponding to Fig. 2

Within the chain $T(j)$ represents a unit of execution during which the resource usage of the chain remains constant. Such a unit will be called a task. The execution of a chain implies a sequence of task initiation and termination events. The task termination events are associated with both the releasing of resources not needed by the next task, and the immediate requesting of the additional resources necessary for initiating the next task. The task initiation events are associated with the allocation of the resources requested at the termination of the previous task.

The interrupt servicing system consists of the parallel combination of chains defined above. Then the state of the system is described by the pair of vectors

$$\underline{P}(k) = (P_1(k), \ldots, P_N(k))$$

and

$$\underline{Q}(k) = (Q_1(k), \ldots, Q_N(k))$$

where $P_i(k)$ and $Q_i(k)$ denote, respectively, the number of memory blocks held and requested by the ith chain after the kth event.

Let $\overline{T}(j)$ and $\underline{T}(j)$ denote, respectively, the initiation and termination events of task $T(j)$. Then Fig. 5 shows the P_i and Q_i values of the chain of Fig. 4.

$\overline{T}(1)$	$\underline{T}(1)$	$\overline{T}(2)$	$\underline{T}(2)$	$\overline{T}(3)$	$\underline{T}(3)$	$\overline{T}(4)$	$\underline{T}(4)$	$\overline{T}(5)$	$\underline{T}(5)$
1	1	2	1	1	1	2	1	1	0
0	1	0	0	0	1	0	0	0	0

Fig. 5

P_i and Q_i values of chain of Fig. 4

Note that Q_i can have values of 0 or 1 only. The interpretation of $Q_i(k) \neq 0$ is that the ith chain is awaiting the allocation of a memory block. $P_i(k) \neq 0$ and $Q_i(k) = 0$ implies that the ith chain is in execution.

Let w denote the system capacity, that is, the total number of memory blocks. We say that the system in the kth state contains a deadlock, if there exists a nonempty set D of chain indices such that for each i in D

$$Q_i(k) > w - \sum_{j \in D} P_j(k).$$

Applying the notation to our case, we see that $0 \leqq Q_i(k) \leqq 1$, $0 \leqq P_i(k) \leqq M$ and $w = Nm + M - m$. Suppose that there exists a subset D of chain indices such that

$$Q_i(k) > w - \sum_{j \in D} P_j(k)$$

for each $i \in D$.

We shall come to a contradiction by this. There are three cases to consider.

Case 1. $P_j(k) \leqq m$ for $j=1, 2, \ldots, N$. Then

$$w - \sum_{j \in D} P_j(k) \geqq w - |D|m \geqq w - Nm = M - m = 3 > Q_i(k)$$

for $i=1, 2, \ldots, N$.

Case 2. There is a chain index r such that

$$P_j(k) \leqq m \quad \text{if} \quad j \neq r$$

and

$$m < P_r(k) \leqq M.$$

Then we have two subcases.

Case 2A: $r \notin D$. Then

$$w - \sum_{j \in D} P_j(k) \geqq w - |D|m \geqq w - (N-1)m = M > Q_i(k)$$

for $i=1, 2, \ldots, N$.

Case 2B: $r \in D$. Then

$$w - \sum_{j \in D} P_j(k) \geqq w - P_r(k) - (|D|-1)m \geqq w - P_r(k) - (N-1)m = M - P_r(k)$$

but $P_r(k) + Q_r(k) \leqq M$, therefore $M - P_r(k) \geqq Q_r(k)$ so there is an index $r \in D$, for which

$$Q_r(k) > w - \sum_{j \in D} P_j(k)$$

does not hold.

6. Conclusions

Because of the modest instruction set of the minicomputers the size of the code increases rapidly with the complexity of the algorithm. The optimal solution can hardly be found, in general it needs lengthy experimentation. The price given for the simplicity of our algorithm is the poor utilization of the memory. We could solve the problem using less memory blocks via more complex algorithm.

The machine independently defined algorithms for deadlock avoidance could hardly decrease the timing efficiency. It is worth noting, that the machine independency versus efficiency problem did not occured in the other parts of the system, in spite that a machine independent higher level language (CDL) was used.

RESEARCH INSTITUTE FOR APPLIED
COMPUTER SCIENCE
BUDAPEST

References

[1] Coffman, E. G. & P. J. Denning, *Operating systems theory*, Prentice-Hall, Englewood Cliffs, N. J., 1973.

[2] Koster, C. H. A., A compiler compiler, *Report* MR 127, Mathematics Centrum, Amsterdam, 1971.

[3] Jacobson, M. & J. Müller, The buddy system in CDL, Machine oriented higher level languages, North Holland Pub. Co. Amsterdam, 1974.

(Received June 8, 1976)

A strategy for scheduling splittable tasks to reduce schedule length

By J. BŁAŻEWICZ, W. CELLARY, J. WĘGLARZ

1. Introduction

In deterministic problems of scheduling tasks on processors (static job-shop problems) it is usually assumed that task execution times are known in advance. Of course, in practice this assumption is not always met, but even then the solution of deterministic problems of scheduling has an important practical meaning. Firstly, when the expected values of task execution times are known, then it is possible, using established techniques [3], to find an optimistic estimate of the expected value of the schedule length. Secondly, upper bounds of execution times of individual tasks may be known. Then scheduling using these values corresponds to the analysis of the worst case and is applied in hard-real-time problems with strict deadlines that must be observed.

Independently of this one can measure task execution times after processing a given set of tasks and use them to find an optimal schedule. This allows one to estimate an operational scheduler and to draw conclusions about possible improvments.

It becomes more and more important to schedule splittable (preemptable) tasks i.e. those that may be preempted; the processing of preempted task may resume where it left off without any extra work or time being occasioned by the preemption. Examining splittable tasks is of a great importance in systems of parallel processors using a common operating store. Such systems have increasingly many applications in the control of such processes as traffic, telephone switch control organization [5, 11] in which several processors using a common data base and computational procedures are being used. It is easy to verify that the possibility of preemption is profitable for improving the schedule length.

Scheduling splittable tasks was considered in [8, 9, 10]. Algorithms, presented in these papers, concern only homogeneous processors and relatively simple precedence relations among tasks. In [4, 7], the problem was considered of scheduling independent tasks on processors that are consistently fast or donsistently slow for all the tasks. In the papers mentioned above non-enumerative algorithms were presented. However the problem of scheduling dependent, splittable tasks, in the general case, is known to be polynomial complete [4] and hence unlikely to admit a non-enumerative solution. Thus, for this case the direct use of scheduling strategies in an operating system has rather restricted applications. Finding such strategies has, however, practical significance for the following reasons. Firstly, one can use them to estimate an operational scheduler. Secondly, the distance between an optimal solution and a suboptimal one for a heuristic, non-enumerative approach, may be found. Lastly, enumerative algorithms may be used in computer centres that perform large and complex numerical computations but not in a real-time environment.

In this paper such a scheduling strategy will be presented, which gives some particular advantages. Then it will be compared for the case of homogeneous processors with the strategy described in [2].

2. Scheduling on heterogeneous processors

There are given a set of m processors $P_1, P_2, ..., P_m$ and a set of n tasks $T_1, T_2, ..., T_n$. The execution time of task T_j on processor P_i will be denoted by τ_{ij}, where τ_{ij} is a positive real number.

We will assume, that precedence relations among tasks are given in the form of an activity network in which arcs correspond to tasks and nodes to events. Let the number of nodes of the network be equal to $r+1$. It will be assumed that the events are ordered in such a way that event j does not occur earlier than event i if $i<j$.

The concept of the algorithm for scheduling splittable tasks on heterogeneous processors to minimize schedule length was given in [1]. For this purpose the following denotations were introduced:

— S_k, $k=1, 2, ..., r$, the set of all tasks which may be processed between the occurrence of event k and $k+1$. This set is called the main set;

— $K_j, j=1, 2, ..., n$, the set of indices of these main sets in which task T_j may be processed.

For a given schedule we denote:

— $x_{ijk} \in \langle 0, 1\rangle$, $i=1, 2, ..., m$; $j=1, 2, ..., n$; $k \in K_j$, a part of task T_j processed on processor P_i in S_k;

— $t_{ijk}=\tau_{ij} \cdot x_{ijk}$, the processing time of a part x_{ijk};

— $t_{ijk}=\sum_{i=1}^{m} t_{ijk}$, $j=1, 2, ..., n$; $k \in K_j$ the processing time of a part of task T_j processed in S_k;

— $t_j=\sum_{k \in K_j} t_{jk}$, $j=1, 2, ..., n$, the processing time of the whole task T_j;

— y_k, $k=1, 2, ..., r$, the schedule length in S_k;

— $y=\sum_{k=1}^{r} y_k$, the schedule length.

Using the above denotations, the following linear programming (LP) problem may be formulated:

Minimize y
Subject to

$$\sum_{k\in K_j}\sum_{i=1}^{m} x_{ijk} = 1 \quad j = 1, 2, ..., n, \tag{1}$$

$$y_k - \sum_{j\in S_k} x_{ijk}\tau_{ij} \geqq 0 \quad \begin{matrix} i = 1, 2, ..., m, \\ k = 1, 2, ..., r, \end{matrix} \tag{2}$$

$$y_k - \sum_{i=1}^{m} x_{ijk}\tau_{ij} \geqq 0 \quad \begin{matrix} j = 1, 2, ..., n, \\ k\in K_j. \end{matrix} \tag{3}$$

Equation (1) guarantees that every task will be processed; inequality (2) defines y_k's as the schedule lengths; inequality (3) assures that obtaining a feasible schedule will be a possible, i.e. one such that no task is processed simultaneously on more than one processor.

As the result of solving the described LP problem the optimal values y^*, x^*_{ijk}, t^*_{ijk} and t^*_j, $i=1, 2, ..., m$; $j=1, 2, ..., n$; $k\in K_j$, are obtained. However, all starting points of parts of tasks are unknown. These points may be found by using the rule shown in Fig.1. As the initial values for the rule, the optimal values, obtained by solving the LP problem formulated above, are taken. In Fig. 1 $t(i)$, $i=1, 2, ..., m$, denotes the processing time t_{jk} of the i'th assigned task, and $t(m+1)$ the processing time t_{jk} of the first unassigned task.

3. Development of the algorithm

In the problem described in Section 2. the first feasible solution is known in advance — it is the sequential processing of all the tasks on a single processor.

The number of variables is $v=(m+1)\cdot(r+\sum_{j=1}^{n}|K_j|)$, where $|K_j|$ denotes the number of elements of set K_j. The number of constraints is $c=n+mr+\sum_{j=1}^{n}|K_j|$. For solving this problem the Revised Simplex Method [6] is worthwhile, because for most cases $v>3c$. It is clear that the number of variables and constraints increases with increasing of the number of tasks and processors. For example for 5 processors and not very complicated networks containing 10 tasks the number of variables is about 100, 30 tasks — 500, 60 tasks — $1.5\cdot10^3$ and 100 tasks — $5\cdot10^3$. The numbers of constraints for the same networks are respectively about 50, 200, 400 and 800. If we want to use directly one of the simplex methods for solving the LP problem, about 10^7 memory cells for 100 tasks will be needed because of the necessity of memorizing the matrix of coefficients which is the largest one in the problem. Thus the direct use of simplex methods has here a very restricted application.

Below an approach which allows for the great reduction of the difficulty mentioned above will be shown.

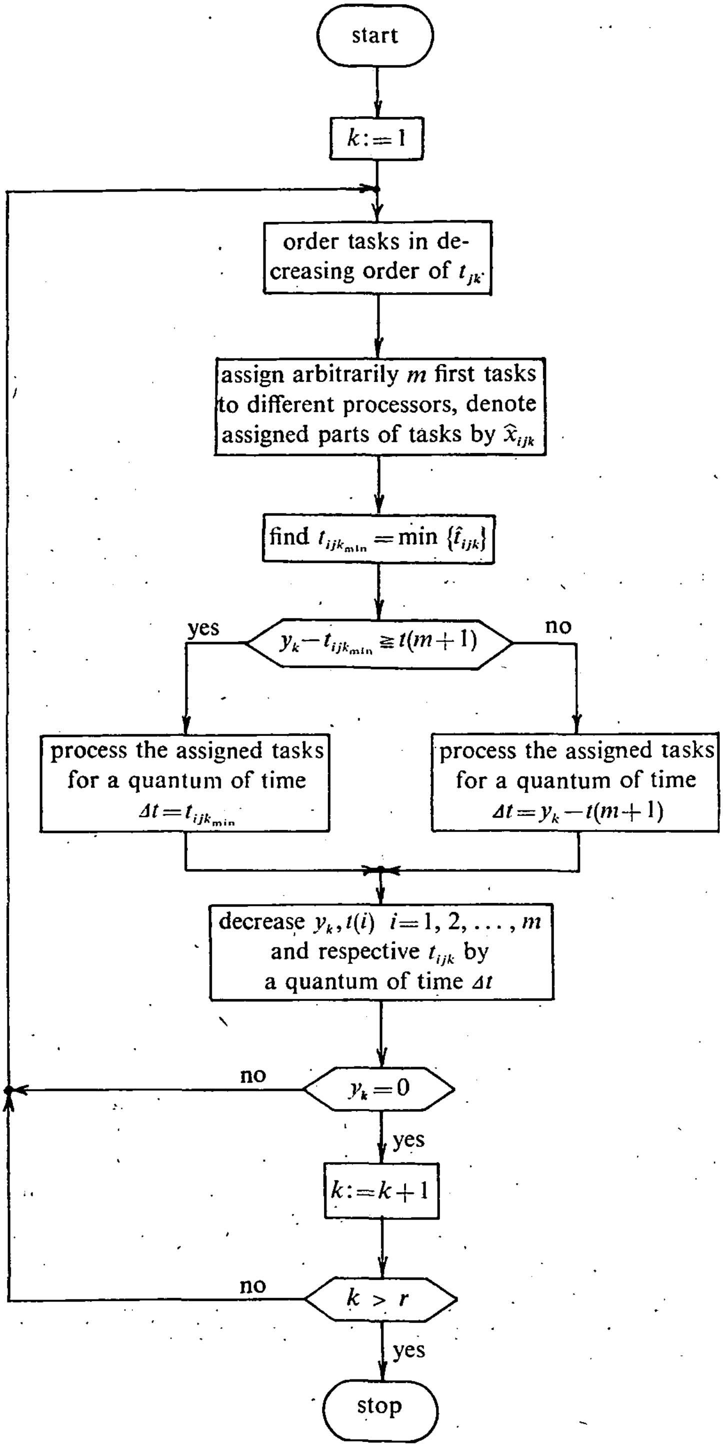

start
$k:=1$
order tasks in decreasing order of t_{jk}.
assign arbitrarily m first tasks to different processors, denote assigned parts of tasks by $\hat{x}_{ijk}$
find $t_{ijk_{\min}} = \min \{\hat{t}_{ijk}\}$
yes
$y_k - t_{ijk_{\min}} \geqq t(m+1)$
no
process the assigned tasks for a quantum of time $\Delta t = t_{ijk_{\min}}$
process the assigned tasks for a quantum of time $\Delta t = y_k - t(m+1)$
decrease $y_k, t(i)$ $i=1, 2, \ldots, m$ and respective t_{ijk} by a quantum of time Δt
no
$y_k = 0$
yes
$k:=k+1$
no
$k > r$
yes
stop

Fig. 1
Finding starting points of x^*_{ijk}'s

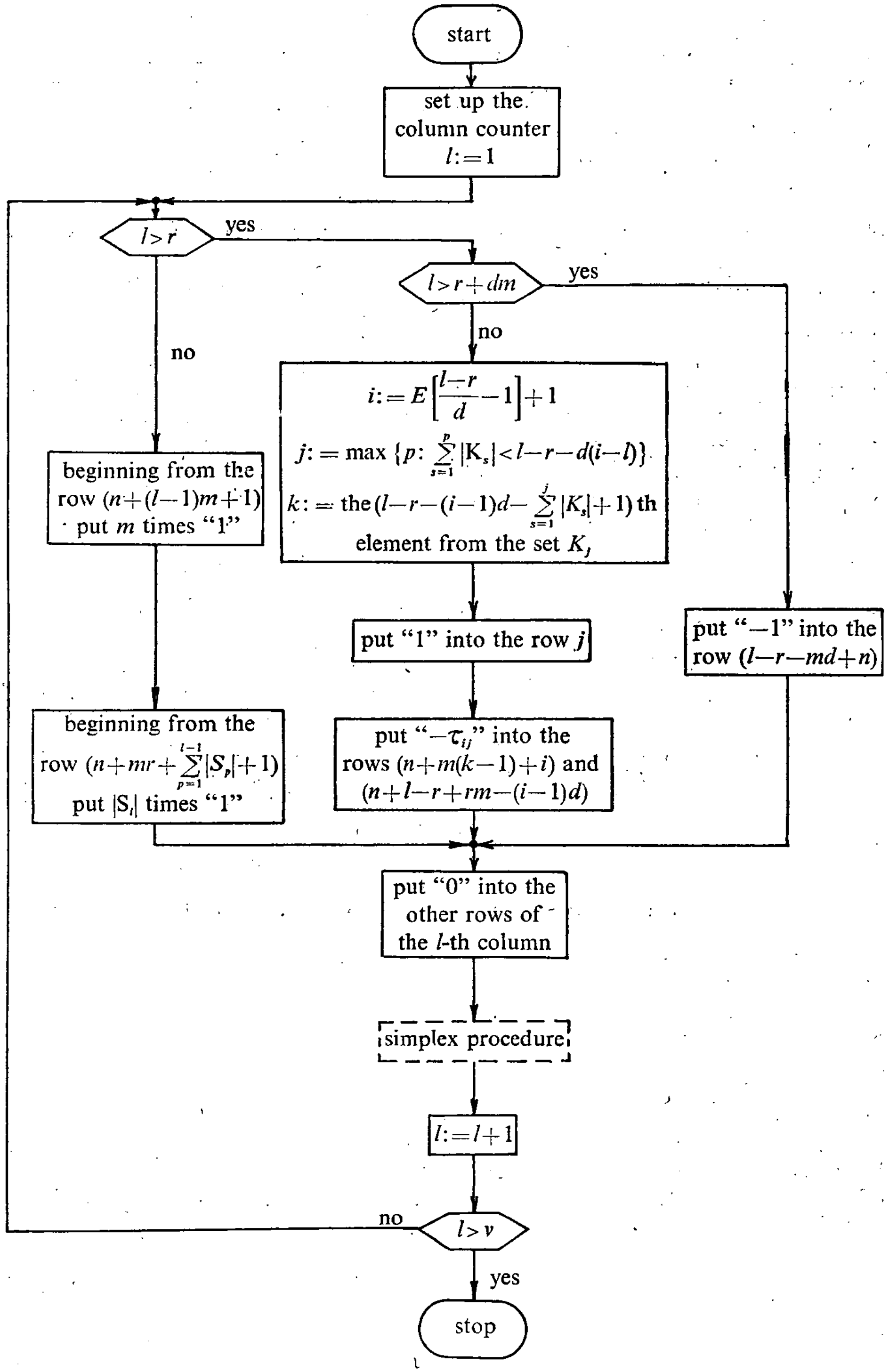

start
set up the column counter $l:=1$
$l>r$
yes
no
$l>r+dm$
yes
no
$i:=E\left[\frac{l-r}{d}-1\right]+1$
$j:=\max\{p:\sum_{s=1}^{p}|K_s|<l-r-d(i-l)\}$
$k:=$ the $(l-r-(i-1)d-\sum_{s=1}^{j}|K_s|+1)$ th element from the set K_j
beginning from the row $(n+(l-1)m+1)$ put m times "1"
put "1" into the row j
put "—1" into the row $(l-r-md+n)$
beginning from the row $(n+mr+\sum_{p=1}^{l-1}|S_p|+1)$ put $|S_l|$ times "1"
put "$-\tau_{ij}$" into the rows $(n+m(k-1)+i)$ and $(n+l-r+rm-(i-1)d)$
put "0" into the other rows of the l-th column
simplex procedure
$l:=l+1$
no
$l>v$
yes
stop

Fig. 2

Generation of consecutive columns of the matrix of coeffcients in one simplex iteration

The idea of this approach is based on a generation of consecutive columns of the matrix of coefficients in every simplex iteration. Such generation is possible, because in the Revised Simplex Method the elements of the matrix of coefficients are constant during computation. As a result only one column of this matrix has to be stored at any moment, and so storage requirements are significantly reduced.

For the purpose of describing the technical aspects of generation let us distinguish among columns of the matrix of coefficients three sets of columns. The first set contains r columns corresponding to variables y_k; the second set — $m \cdot \sum_{j=1}^{n} |K_j|$ columns corresponding to variables x_{ijk}; the third set — $mr + \sum_{j=1}^{n} |K_j|$ columns corresponding to artificial variables. After identification of the actually generated column to which the set does belong, appriopriate values are put into the rows corresponding to constraints (1), (2) and (3) on the base of the minimum information about the structure of the problem. This information includes n, m, matrix of execution times $[\tau_{ij}]$ and the vector describing the structure of the network, containing arcs as ordered pairs of nodes. After generation a single column, one check the benefit of introducing this column into the solution of the LP problem in accordance with the simplex procedure. The number of constraints (1), (2), (3) are equal respectively to n, $r \cdot m$ and $\sum_{j=1}^{n} |K_j|$. The block diagram of the generation of consecutive columns of the matrix of coefficients is one simplex iteration is shown in Fig. 2. In Fig. 2. $d = \sum_{j=1}^{n} |K_j|$.

The fact must be stressed that the computer time used by the algorithm in comparison with the time used by the algorithm in which the procedure of generation is not used, is reduced, except for small problems which do not require mass storage. Of course, if the network-node ordering is not given, the obtained schedule is in general a suboptimal one. The optimal schedule may be obtained by choosing the best one from among optimal solutions for all possible orders.

4. Scheduling on homogeneous processors — a comparison of two algorithms

In the case of homogeneous processors, tasks may be scheduled in accordance with the algorithm described in Sections 2 and 3. Let us call it the *A-algorithm*. However, for this case, a special algorithm has been elaborated [2] which will be called the *B-algorithm*. In this Section we present the conceptual basic of this algorithm in comparison with the *A*-algorithm.

In the *B*-algorithm we also use the concept of the main sets S_k, $k = 1, 2, \ldots, r$, which was introduced in Section 2.

Let us number from 1 to N the feasible sets, i.e. those subsets of all main sets, in which the number of elements is not greater than m. Now let Q_j denote the set of all numbers of the feasible sets in which task T_j may be processed and t_i the duration of set i. Thus one obtains the LP problem:

Minimize

$$y = \sum_{i=1}^{N} t_i.$$

Subject to

$$\sum_{i \in Q_j} t_i = \tau_j \quad j = 1, 2, \ldots, n \tag{4}$$

or in matrix notation

$$At = \tau$$

where A is the matrix of coefficients:

$$a_{ij} = \begin{cases} 1 & \text{if } i \in Q_j \\ 0 & \text{otherwise.} \end{cases}$$

Obviously, the columns of matrix A correspond to the resource feasible sets. The number of variables in this problem is much greater than in A-algorithm, for example: for 5 processors and 10 tasks it is about 50, 30 tasks — $2 \cdot 10^3$, 60 tasks — $3 \cdot 10^4$ and 100 tasks — $2 \cdot 10^5$. On the other hand, the number of constraints is much smaller than in A-algorithm, because it is equal to the number of tasks (see (4)).

In order to avoid the storage of matrix A, the method of automatic generation of columns for B-algorithm, for this matrix was also elaborated [2].

Comparing these two algorithms one should pay attention to core store requirements and computer time.

Core store requirement for both algorithms is equal 16c. So in this respect, it is more worthwhile to use the B-algorithm, because the number of constraints c in it is much smaller. The number of variables as well as the number of constraints influence computer time. In Table 1 computer times for A- and B-algorithms are compared for not very complicated networks and 5 processors. These results were produced using programs written in FORTRAN IV and processed on an ODRA 1305.

Table 1.

Number of tasks	Iterations to optimum		Computer time of single iteration [S]	
	Algorithm A	Algorithm B	Algorithm A	Algorithm B
10	65	12	1.5	0.9
30	250	39	3.5	3.1
60	500	88	6.2	7.3
100	1000	151	10.1	18.2

It proceeds from Table 1 that using the B-algorithm one reaches the optimum faster within the scope of studied examples. However, it seems that as the size (number of tasks) of the problem increases, the performance of A-algorithm relatively improves, but for both algorithms, the time used to reach the optimum permits their practical application to problems of up to 100 tasks.

Concluding, one should state that the B-algorithm is better for the case of homogeneous processors and may be used in parctice.

Abstract

This paper deals with deterministic problems of scheduling n preemptable tasks on m parallel processors. The structure of the set of tasks is given in the form of an activity network (i.e. a directed, acyclic graph with only one origin and only one terminal) and the minimizing of the schedule length is the performance measure. The cases of identical as well as heterogeneous processors are considered. The problem of obtaining the minimal schedule lenght is reduced to a linear programm ing problem. In order to provide facilities for solving problems of a practical size, the special procedure proposed here considerably reduces computer storage requirements. For the case of identical processors two approaches for solving the problem have been compared.

INSTITUTE OF CONTROL ENGINEERING,
TECHNICAL UNIVERSITY OF POZNAŃ
POZNAŃ, POLAND

References

[1] BŁAŻEWICZ, J., W. CELLARY, J. WĘGLARZ, Scheduling preemptable tasks on heterogeneous processors (submitted for publication).

[2] BŁAŻEWICZ, J., W. CELLARY, J. WĘGLARZ, Some computational problems of scheduling dependent and preemptable tasks to minimize schedule length, *Found. Control Engrg.*, v. 1, 1975, pp. 75—83.

[3] COFFMAN, E. G., JR. & P. J. DENNING, *Operating systems theory*, Prentice Hall, Englewood Cliffs, N. J., 1973.

[4] COFFMAN, E. G., JR. (ed), *Computer & job/shop scheduling theory*, Wiley-Interscience, 1976.

[5] COVO, A. A., Analysis of multiprocessor control organizations with partial program memory replication, *IEEE Trans. Computers*, v. C—23, 1974, pp. 113—120.

[6] GASS, S. J., *Linear programming*, McGraw—Hill, N. Y., 1969.

[7] HORVATH, E. C. & R. SETHI, Preemptive schedules for independent tasks, *Technical Report*, No 162, Computer Science Dep., Pennsylvania State Univ., 1975.

[8] MC NAUGHTON, R., Scheduling with deadlines and loss functions, *Management Sci.*, v. 6, 1959, pp. 1—12.

[9] MUNTZ, R. R. & E. G. COFFMAN, JR., Optimal preemptive scheduling on two-processor systems, *IEEE Trans. Computers*, v. C—18, 1969, pp. 1014—1020.

[10] MUNTZ, R. R. & E. G. COFFMAN, JR., Preemptive scheduling of real-time tasks on multiprocessor systems, *J. Assoc. Comput. Mach.*, v. 17, 1970, pp. 324—338.

[11] TORRES, J., Test and evaluation of computer traffic control system, *Diamond Interchange Traffic Control*, PB—224160, v. 9, 1973.

(Received Feb. 16, 1976)

A maximum-selector design in Codd-ICRA cellular automaton

By D. V. Takács

I. The Codd-ICRA cellular automaton

Introduction

The idea of the cellular automaton is created by John von Neumann [4], in 1948.

The cellular automaton is a set of identical, synchronized, finite, deterministic automata — called cells — connected by a homogeneous pattern. The cells directly connected with a certain cell, called neighbours of the latter one, and the state of a cell in a certain time step depends only on its own state and its neighbours' state in the last time step. The function, which governs this dependence, is called transition function.

In 1968 E. F. Codd published in his book *Cellular Automata* [1] a transition function in explicite form. This and his construction's techniques were our starting points in the Hungarian ICRA team since 1971.

We modified in some measure the Codd's transition function and techniques, keeping the main principles of the Codd-type cellular automaton. This new version is named Codd-ICRA cellular automaton.

The work of the ICRA team was motivated by practical purposes. Accordingly with the present technologies, it's not hopeless anymore the industrial producing such microelectronical circuits, which are the implementation of the Codd-ICRA cells. Therefore, one gets the possibility for constructing a new type of computer, constituted by cells only, in which the following advantages would be joined

— absolutely parallel working,
— hierarchiless,
— production homogeneity,
— flexible connection,
— flexible transformation,
— high speed,
— simple extensibility,
— absolute invertability of the software with the hardware,
— the manufacturing use of the selfreproducing ability.

Now let us introduce a bit formally the basic concepts of cellular automata. After having these, we show some Codd-ICRA techniques. Finally, we present a possible maximal number selector designed as a Codd-ICRA automaton.

Cellular automata and Codd-ICRA automaton

A *cellular automaton* consists of a collection of identical cells occupying a certain set of "places" I, and at each time step they change their states belonging to a finite set of possible states S, and this change takes place as a consequence of their mutual influences.

A cellular automaton, by definition, is a five-tuple $\mathfrak{A}=(I, L, v, S, \perp)$ where

$I\subseteq \mathbf{Z}^k$ is a subset of the points having integer coordinates in a k-dimensional Euklidian space. We shall associate a cell to each point i of I, and we shall identify it with i.

$L\subseteq \mathbf{Z}^k \ominus I$ is the set of dummy cells.

$v\colon I\longrightarrow (I\cup L)^n$ is the *neighbourhood function*, where n is the number of all neighbours of a single cell.

(I, L, v) is the socalled *cellular space*, which is homogeneous in the following sense; for every $i\in I$, $v(i)=\langle i+a_1, \dots, i+a_n\rangle$, where $a_1, \dots, a_n$ are fixed vectors. The cells, having dummy cell neighbour, namely the elements of the set P

$$P = \{i\in I | v(i)\notin I^n\},$$

we call *boundary cells*.

S is the finite set of the possible *states*.

$\perp\colon S\times S^n\longrightarrow S$ is called *local transition function*. If $i\in I$ and $s\in S$, then the cell, located at i, being in state s, with the neighbourhood $v(i)_1$ in state $s_1', \dots, v(i)_n$ in state s_n', will have the next state $\perp(s, \langle s_1', \dots, s_n'\rangle)$ denoted as

$$s\perp\langle s_1', \dots, s_n'\rangle.$$

If $s_0\in S$, and $s_0\perp\langle\underbrace{s_0, \dots, s_0}_{n\text{-times}}\rangle=s_0$, s_0 is called *quiescent state*.

$q\colon I\longrightarrow S$ is the *global state*, or *configuration* of the cellular automaton, which associates a state to each cell. The state of the cell $i\in I$, will be denoted by $q(i)$ or q_i, and the set of the global states by S^I.

$\lambda\colon L\longrightarrow S$. The mapping λ is the *input* of the cellular automata, which comes from the state of the dummy cells.

$\pi\colon P\longrightarrow S$ is the *output* of of the cellular automaton, which is a restriction of $q(P\subseteq I)$.

S^L: is the set of the inputs.

S^P: is the set of the outputs.

The generalization of the local transition function is the global *transition function*, which is the following mapping:

$\perp\!\!\!\perp\colon S^I\times S^L\longrightarrow S^I$ such that $q\perp\!\!\!\perp\lambda=q'$ iff

$$(\forall i\in I), q'(i) = q(i)\perp\!\!\!\perp\langle r(v(i)_1), \dots, r(v(i)_n)\rangle, \text{ where}$$

$$r(j) = \begin{cases} q(j), & \text{if } j\in I, \\ \lambda(j), & \text{if } j\in L. \end{cases}$$

The *global quiescent state* is q_0 for which

$$(\exists \lambda_0) q_0 \perp\!\!\!\perp \lambda_0 = q_0.$$

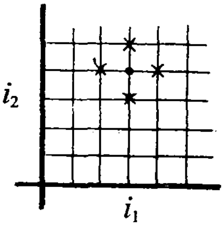


Fig. 1

The *Codd-ICRA automaton* $\mathfrak{A}=(I, L, v, S, \perp)$, is a special cellular automaton ([5], [6])[1], where $I \subset \mathbf{Z}^2$ ($\mathbf{Z}^2$ is the plane of integers), $S=\{0, 1, 2, 3, 4, 5, 6, 7\}$. For every $i=(i_1, i_2) \in I$, v is defined by $v(i_1, i_2)=\langle (i_1+1, i_2), (i_1, i_2-1), (i_1-1, i_2), (i_1, i_2+1) \rangle$ (see fig. 1).

Let ν be the mapping

ν: $S^4 \longrightarrow S^4$ such that $\nu(\langle s_1, s_2, s_3, s_4 \rangle)$ is the smallest word under lexicographic ordering, what one can get from $\langle s_1, s_2, s_3, s_4 \rangle$ by cyclic permutation: min lex $\{\langle s_1, s_2, s_3, s_4 \rangle, \langle s_2, s_3, s_4, s_1 \rangle, \langle s_3, s_4, s_1, s_2 \rangle, \langle s_4, s_1, s_2, s_3 \rangle\}$.

$\perp$ is defined for all $s \in S$ and $\langle s_1', s_2', s_3', s_4' \rangle \in S^4$ by

$$s \perp \langle s_1', s_2', s_3', s_4' \rangle =$$

$$= \begin{cases} s & \text{if } \langle s \rangle \nu(s_1', s_2', s_3', s_4') \text{ does not appear in the Codd-ICRA transition function table,} \\ s \perp \nu(s_1', s_2', s_3', s_4'), & \text{if } \langle s \rangle \nu(s_1', s_2', s_3', s_4') \text{ appears in the Codd-ICRA transition function table.} \end{cases}$$

How to use the transition function table? E.g. we are looking for the transition

$$0 \perp \langle 7212 \rangle$$

we have to look instead of this the term

$$0 \perp \langle 1272 \rangle$$

which appeares in the list and equal to 1, one can find it in the short form

$$0\ 1272\ 1$$

which means, that the next state of the central cell will be 1. Obviously, the $0 \perp \langle 4131 \rangle$, instead of which we have to look $0 \perp \langle 1314 \rangle$, and it doesn't occur, it means, the central cell in this environment has stable state. By regular notations the first example gives

$$0 \perp \langle 7212 \rangle = 1,$$

the second one

$$0 \perp \langle 4131 \rangle = 0.*$$

Codd-ICRA techniques

Basic operations. Let us make ourself a bit familiar with some Codd-ICRA techniques ([1], [3]).

The cells themselves are denoted by squares, their states with numbers written into them respectively.

* See [1] [7].

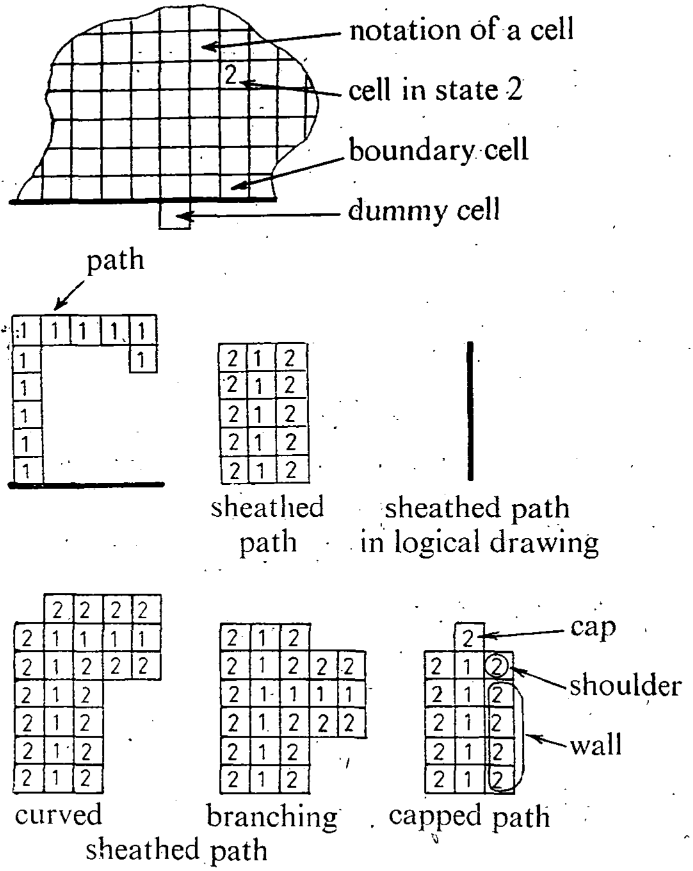


Fig. 2

We use very often phase figures from which one can see the situations in time steps $t=1, \ldots$. For instance we have the following configuration at time step one. The blank place means that the corresponding cell is in 0 state. We have to look in the transition table all the instances which appear on the figure. In the upper left corner of fig. 3 the cell is in state 2 and its neighbourhood is $2\perp\langle 0021\rangle$, which is equal to 2 (200212 doesn't appear in the table). And so on, we find three terms only, which show state changing, namely

$$012621,$$

$$602120,$$

$$112626,$$

so we have the following phase figures.

Along the sheathed paths the states' pair is propagating if this is in form OS, where $S=4, 5, 6, 7$. We call this OS couple of the cells signal. The signal propagation is basic behavior of cellular automaton. Since the signal is propagating cell by cell, so the time required for the propagation is strictly proportional to the distance which is made by it expressed with the number of the cells. In a fixed automaton this factor is constant. Rather often, with inaccurate speaking we say that the distance, which is made is equal to the necessary time.

2	2	2	2	2
1	0	6	1	1
2	2	2	2	2

Fig. 3

	0	
0	2	2
	1	

Fig. 4

2	2	2	2	2
1	0	6	1	1
2	2	2	2	2

$t=1$

2	2	2	2	2
1	1	0	6	1
2	2	2	2	2

$t=2$

Fig. 5

2	2	2	2	2	2	2	2	2
1	0	7	1	1	1	6	0	1
2	2	2	2	2	2	2	2	2

$t=1$

2	2	2	2	2	2	2	2	2
1	1	0	7	1	6	0	1	1
2	2	2	2	2	2	2	2	2

$t=2$

2	2	2	2	2	2	2	2	2
1	1	1	0	1	0	1	1	1
2	2	2	2	2	2	2	2	2

$t=3$

2	2	2	2	2	2	2	2	2
1	1	1	1	1	1	1	1	1
2	2	2	2	2	2	2	2	2

$t=4$

Fig. 6

Along a path collision of signals can happen. Fig. 6 shows such a case. The effect of several collisions depends on the parity of the cells' number is state 1, between two signals, and depends on the signal's content ($S=4$ or 5 or 6 or 7). The result is always the annihilation of the signals, except with the odd parity, homogenous case, where

$$04\times04=05,$$

$$05\times05=06,$$

$$06\times06=06,$$

$$07\times07=04.$$

Fig. 6 gives an example for odd parity, heterogenous collision, fig. 7 an odd, homogenous one.

In the next part we will show, that the original Codd's operations are valid for the Codd-ICRA constructions too. We connect three dummy cells, to three neighbouring boundary cells of the empty cellular automaton. (Empty: it contains cells in state 0 only.) The two extremal cells have state 2 constantly, and we introduce into the middle one the sequences of the Codd-ICRA programmes.

	2	2	2	2	2	2
2	1	1	1	1	1	1
2	1	2	2	2	1	2
2	1	2	2	2	1	2
0	7	1	1	1	1	2
2	2	2	2	2	2	

$t=1$

	2	2	2	2	2	2
2	1	1	1	1	1	1
2	1	2	2	2	1	2
2	7	2	2	2	1	2
1	0	7	1	1	1	2
2	2	2	2	2	2	

$t=2$

	2	2	2	2	2	2
2	1	1	1	1	1	1
2	7	2	2	2	1	2
2	0	2	2	2	1	2
1	1	0	7	1	1	2
2	2	2	2	2	2	

$t=3$

	2	2	2	2	2	2
2	7	1	1	1	1	1
2	0	2	2	2	1	2
2	1	2	2	2	1	2
1	1	1	0	7	1	2
2	2	2	2	2	2	

$t=4$

	2	2	2	2	2	2
2	0	7	1	1	1	1
2	1	2	2	2	1	2
2	1	2	2	2	1	2
1	1	1	1	0	7	2
2	2	2	2	2	2	

$t=5$

	2	2	2	2	2	2
2	1	0	7	1	1	1
2	1	2	2	2	1	2
2	1	2	2	2	7	2
1	1	1	1	1	0	2
2	2	2	2	2	2	

$t=6$

	2	2	2	2	2	2
2	1	1	0	7	1	1
2	1	2	2	2	7	2
2	1	2	2	2	0	2
1	1	1	1	1	1	2
2	2	2	2	2	2	

$t=7$

	2	2	2	2	2	2
2	1	1	1	0	7	1
2	1	2	2	2	0	2
2	1	2	2	2	1	2
1	1	1	1	1	1	2
2	2	2	2	2	2	

$t=8$

	2	2	2	2	2	2
2	1	1	1	1	0	7
2	1	2	2	2	1	2
2	1	2	2	2	1	2
1	1	1	1	1	1	2
2	2	2	2	2	2	

$t=9$

Fig. 7

The *Codd-ICRA programme* is a word constituted by cells' states: 0, 1, 4, 5 and 7, as elements, where each 1 is followed by 1, 4, 5, 6 or 7; the 4, 5, 6 or 7 are followed by one 0, and after a 0 there is at least one 1. With other words, the set of the programmes is identical to an arbitrary walking on the following graph on fig. 8. E.g. the word ⟨1, 1, 1, 5, 0, 1, 1, 6, 0, 1, 6, 0, 1⟩ is a programme. Firstly, we show a *pathway's embryo producing* programme, in order to be able to enter into the empty automaton.

It's the following: ⟨6, 0, 1, 7, 0, 1, 6, 0, 1⟩.

The effect of the programme is shown by fig. 9.

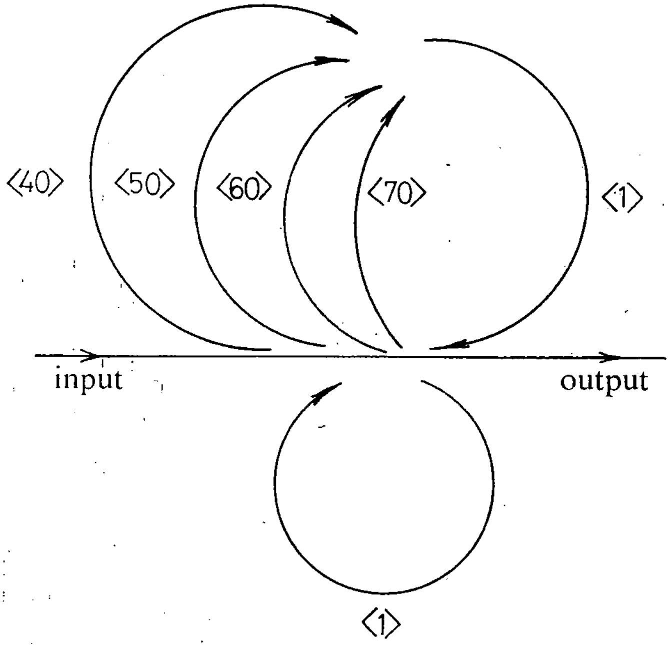


Fig. 8

The three cells under the double line are dummy cells

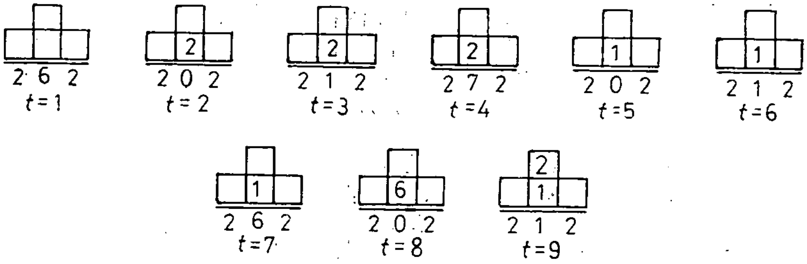


Fig. 9

The effect of the pathway embryo producing programme

The main functions of the pathway which are the Codd's operations.

Extend ⟨7, 0, 1, 6, 0, 1⟩.

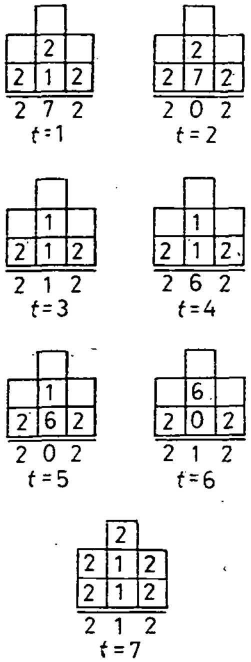


Fig. 10

Extend left ⟨4, 0, 1, 4, 0, 1, 5, 0, 1, 6, 0, 1⟩.

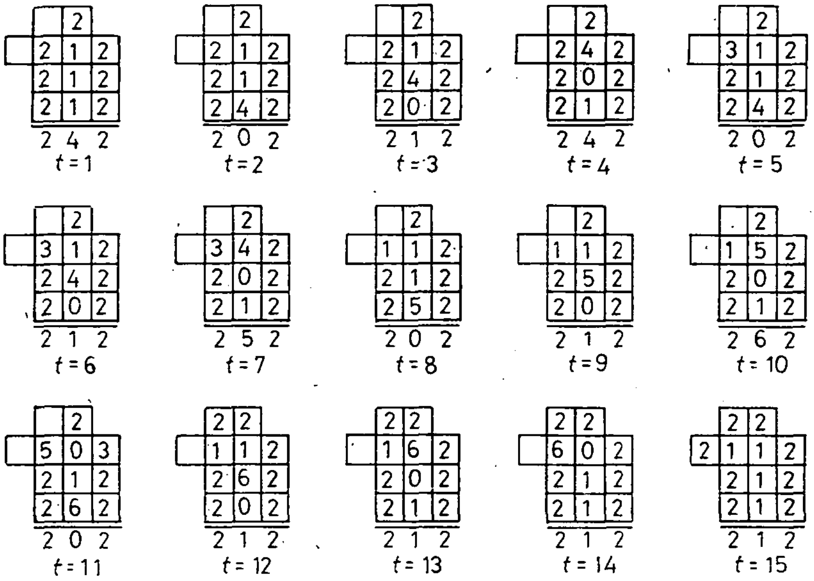


Fig. 11

Extend right ⟨5, 0, 1, 5, 0, 1, 4, 0, 1, 6, 0, 1⟩.

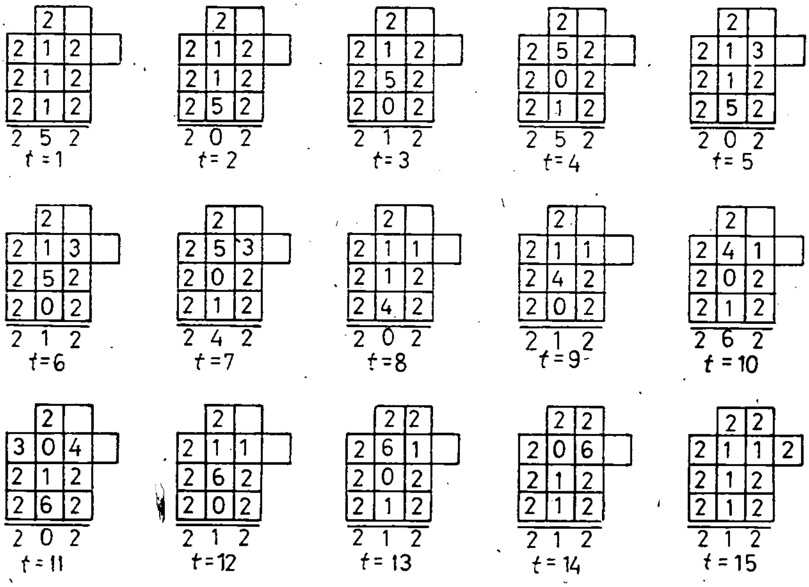


Fig. 12

Retract ⟨4, 0, 1, 5, 0, 1, 6, 0, 1, 6, 0, 1⟩.

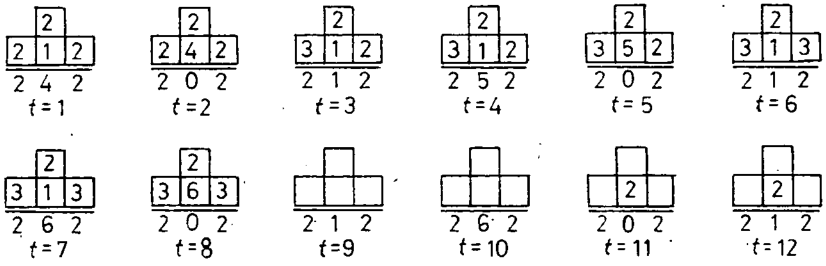


Fig. 13

Retract left ⟨5, 0, 1, 6, 0, 1, 6, 0, 1, 6, 0, 1⟩.

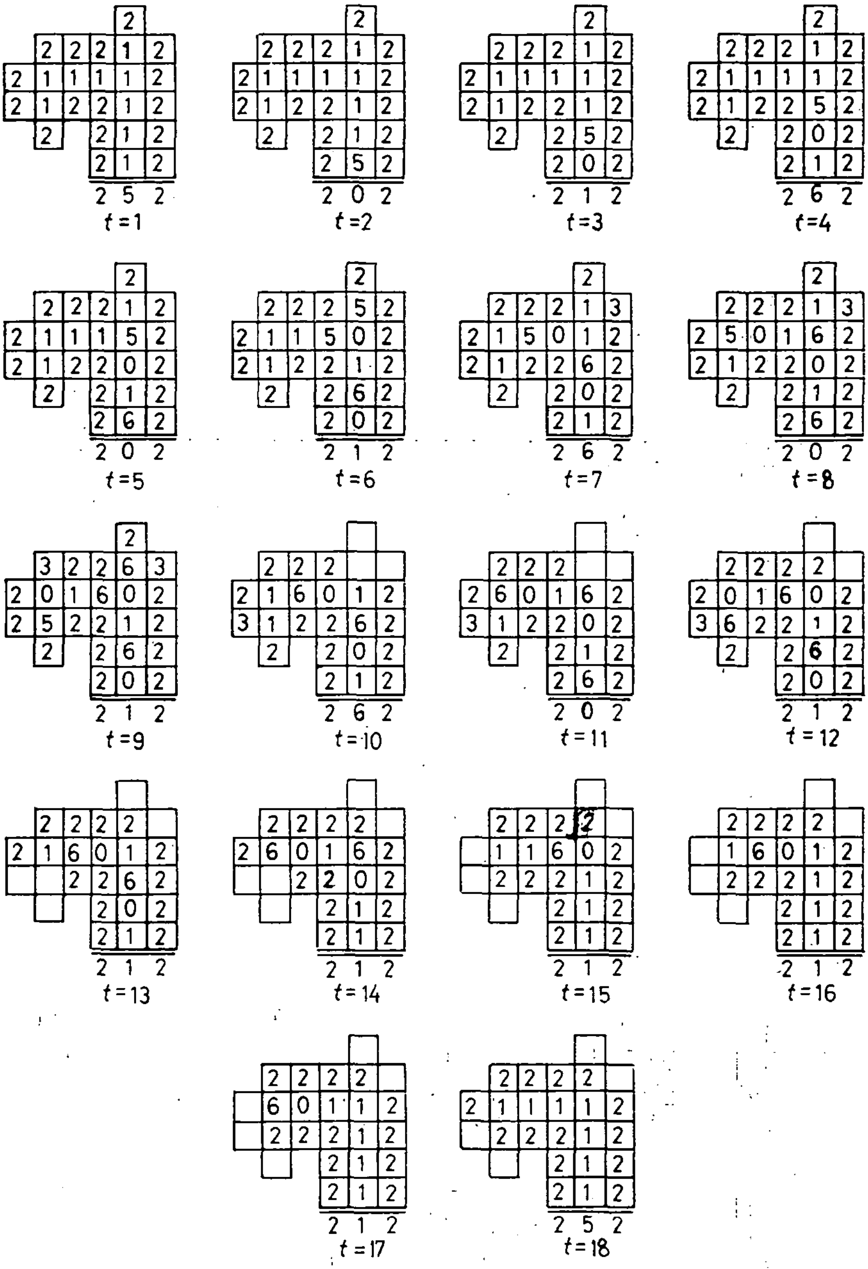


Fig. 14

Retract right ⟨4, 0, 1, 6, 0, 1, 6, 0, 1, 6, 0, 1⟩.

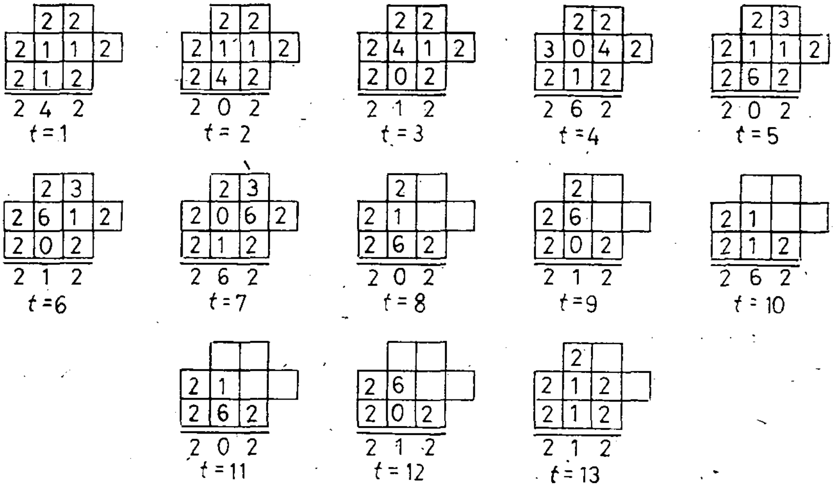


Fig. 15

Sheathing ⟨6, 0, 1⟩.

Sheathed, capped path made by the sheathing signal

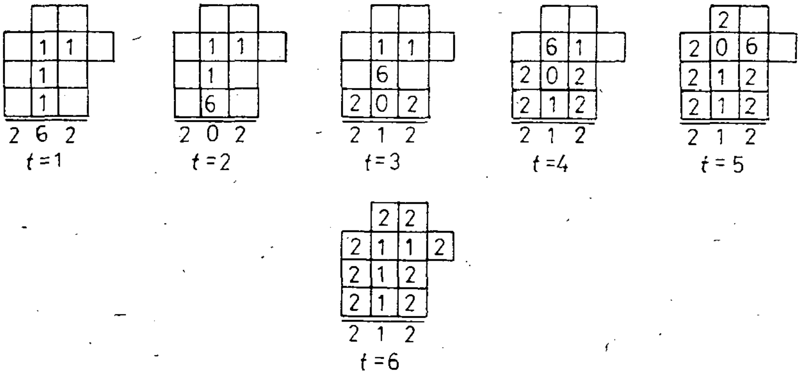


Fig. 16

Activation ⟨7, 0, 1⟩.

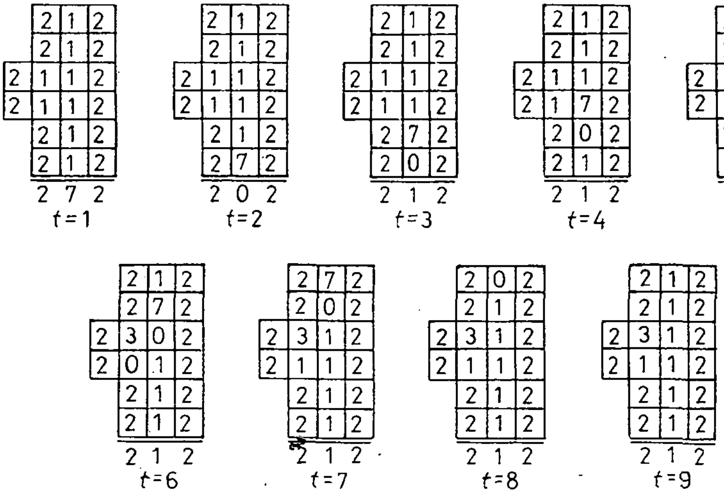


Fig. 17

The activating signal changes the gates' and transformer' (see later) suitable cells being in state 0 or 1 to state 3. These tools become ready for working by this way.

Mark ⟨7, 0, 1, 6, 0, 1, 4, 0, 1, 5, 0, 1, 7, 0, 1, 6, 0, 1⟩.

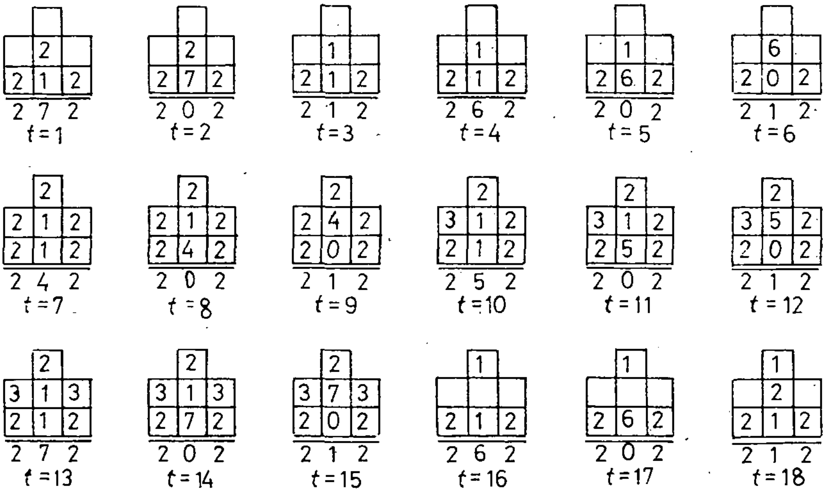


Fig. 18

Erase ⟨6, 0, 1, 7, 0, 1, 4, 0, 1, 5, 0, 1, 6, 0, 1, 6, 0, 1⟩.

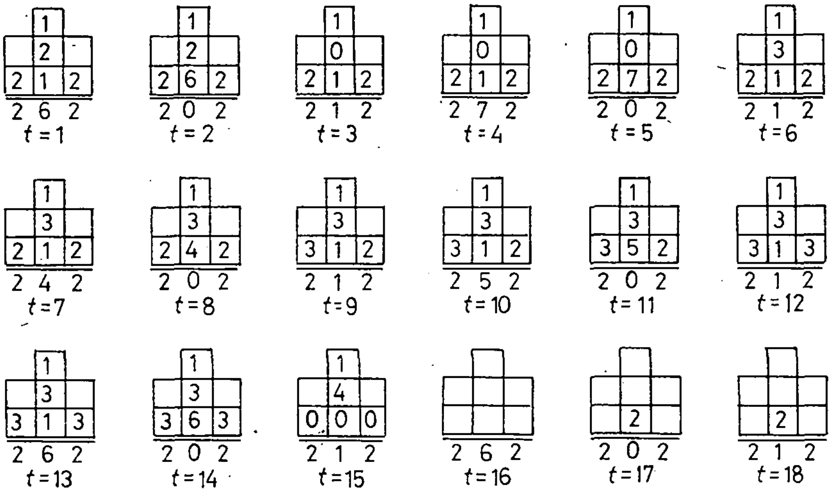

1 / 2 / 2 1 2 / 2 6 2 t=1
1 / 2 / 2 6 2 / 2 0 2 t=2
1 / 0 / 2 1 2 / 2 1 2 t=3
1 / 0 / 2 1 2 / 2 7 2 t=4
1 / 0 / 2 7 2 / 2 0 2 t=5
1 / 3 / 2 1 2 / 2 1 2 t=6
1 / 3 / 2 1 2 / 2 4 2 t=7
1 / 3 / 2 4 2 / 2 0 2 t=8
1 / 3 / 3 1 2 / 2 1 2 t=9
1 / 3 / 3 1 2 / 2 5 2 t=10
1 / 3 / 3 5 2 / 2 0 2 t=11
1 / 3 / 3 1 3 / 2 1 2 t=12
1 / 3 / 3 1 3 / 2 6 2 t=13
1 / 3 / 3 6 3 / 2 0 2 t=14
1 / 4 / 0 0 0 / 2 1 2 t=15
2 6 2 t=16
2 / 2 0 2 t=17
2 / 2 1 2 t=18

Fig. 19

Sense ⟨7, 0, 1, 7, 0, 1⟩

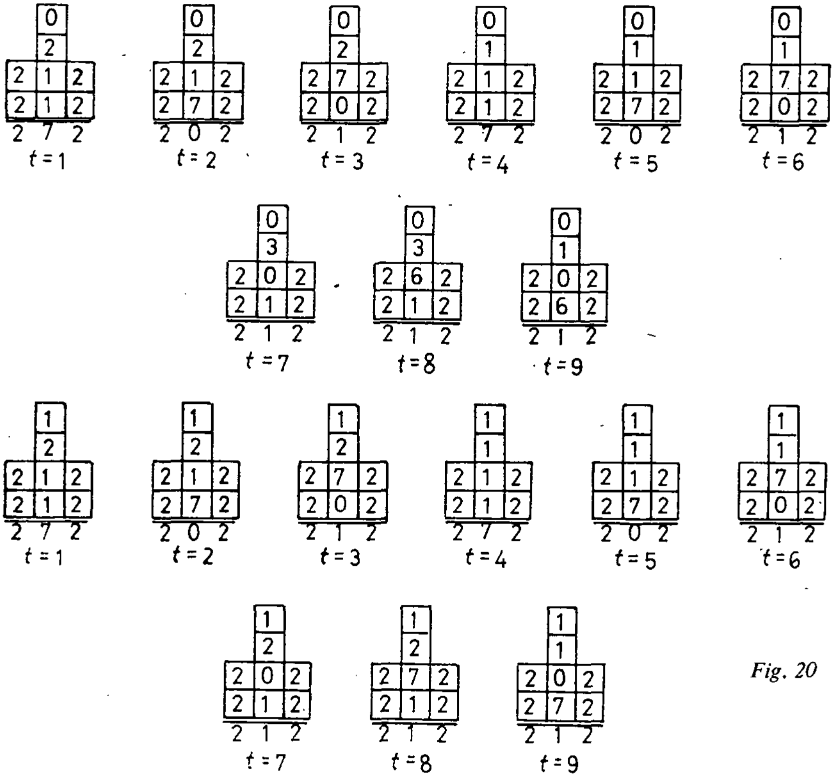

0 / 2 / 2 1 2 / 2 1 2 / 2 7 2 t=1
0 / 2 / 2 1 2 / 2 7 2 / 2 0 2 t=2
0 / 2 / 2 7 2 / 2 0 2 / 2 1 2 t=3
0 / 1 / 2 1 2 / 2 1 2 / 2 7 2 t=4
0 / 1 / 2 1 2 / 2 7 2 / 2 0 2 t=5
0 / 1 / 2 7 2 / 2 0 2 / 2 1 2 t=6
0 / 3 / 2 0 2 / 2 1 2 / 2 1 2 t=7
0 / 3 / 2 6 2 / 2 1 2 / 2 1 2 t=8
0 / 1 / 2 0 2 / 2 6 2 / 2 1 2 t=9
1 / 2 / 2 1 2 / 2 1 2 / 2 7 2 t=1
1 / 2 / 2 1 2 / 2 7 2 / 2 0 2 t=2
1 / 2 / 2 7 2 / 2 0 2 / 2 1 2 t=3
1 / 1 / 2 1 2 / 2 1 2 / 2 7 2 t=4
1 / 1 / 2 1 2 / 2 7 2 / 2 0 2 t=5
1 / 1 / 2 7 2 / 2 0 2 / 2 1 2 t=6
1 / 2 / 2 0 2 / 2 1 2 / 2 1 2 t=7
1 / 2 / 2 7 2 / 2 1 2 / 2 1 2 t=8
1 / 1 / 2 0 2 / 2 7 2 / 2 1 2 t=9

Fig. 20

Cap after sense ⟨4, 0, 1, 6, 0, 1⟩.

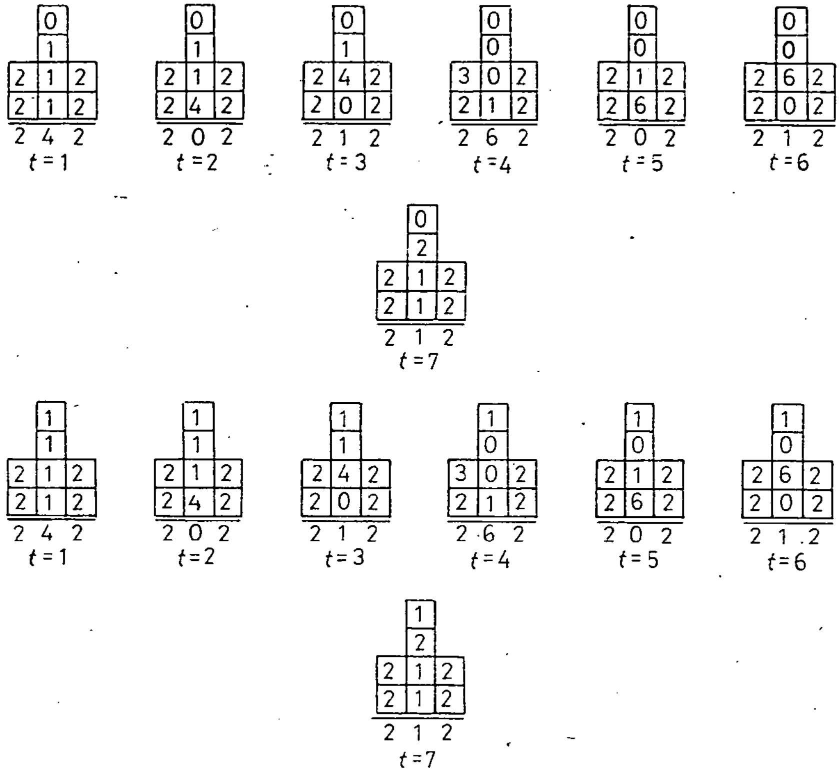
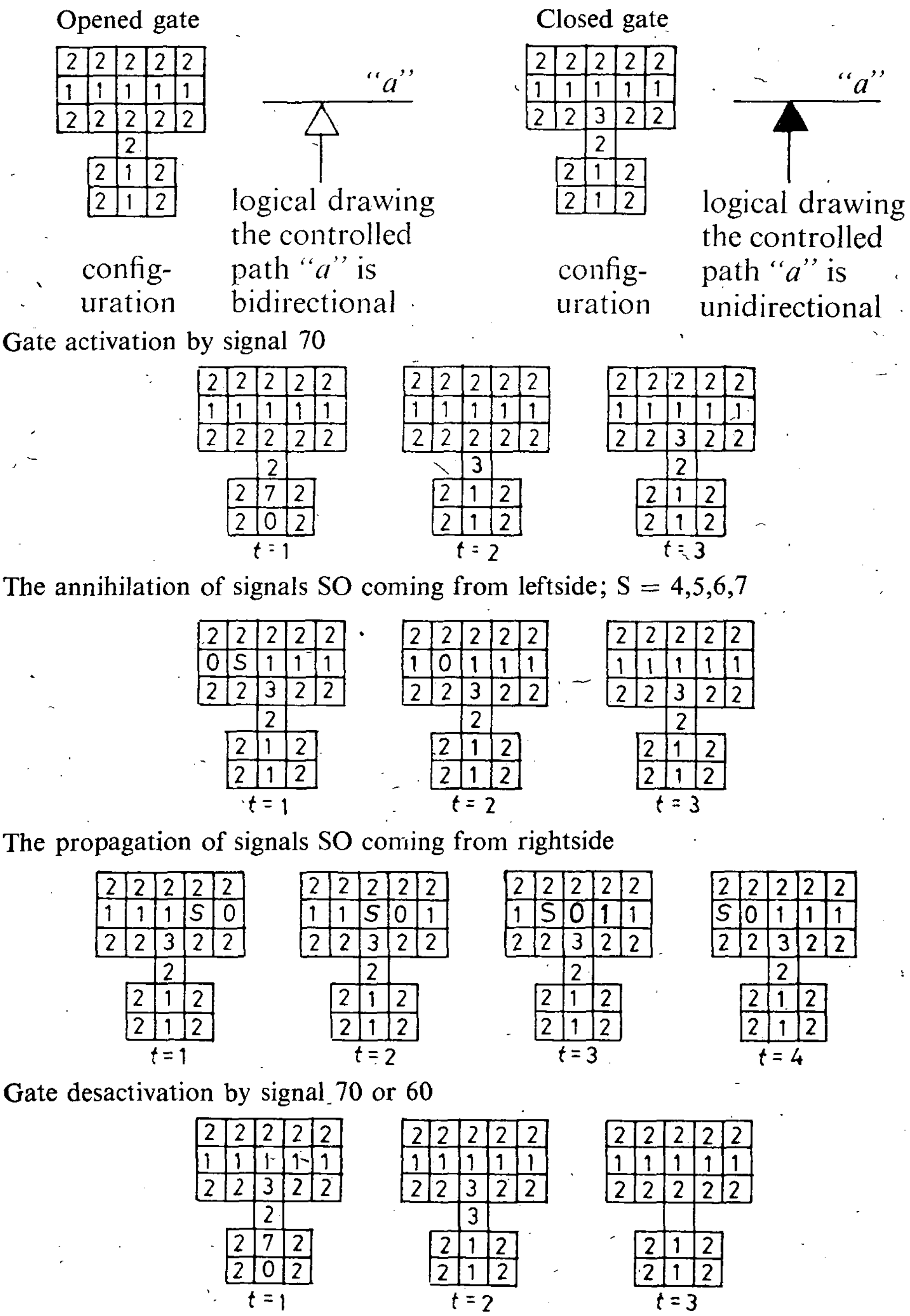


Fig. 21

We have seen, that it's possible to develop directly sheathed paths in the empty cellular space by means of programmes, introduced into the boundary cells from the dummy cells. Why we develop in this case, hovewer, unsheathed paths, making the sheating afterwards? One has to do this, because branching and looping sheathed paths are not programmable directly. So, one has to prepare firstly a simple formed sheathed path, and with this, used it as a writing arm, is possible to develop the required network by the repeated application the mark and retract operation. Finally the stacking of the network is followed by the sheathing and activation.

Some components. Configurations constituted from a few cells which are able to make some prescribed effect on signals propagating along pathways, called components. These effects: to close in one direction the way, the signals' transformation, protection of the paths in the case crossover, and creating signal sources.

One of the most important properties in our cell space is gating. We speak about gating if the signal can propagate in one direction along its path but in the other direction the signals are annihilated at a point. The component doing this annihilation is called gate.



Opened gate

```
2 2 2 2 2
1 1 1 1 1
2 2 2 2 2
    2
  2 1 2
  2 1 2
```

configuration

logical drawing the controlled path "a" is bidirectional

Closed gate

```
2 2 2 2 2
1 1 1 1 1
2 2 3 2 2
    2
  2 1 2
  2 1 2
```

configuration

logical drawing the controlled path "a" is unidirectional

Gate activation by signal 70

```
t = 1        t = 2        t = 3
2 2 2 2 2    2 2 2 2 2    2 2 2 2 2
1 1 1 1 1    1 1 1 1 1    1 1 1 1 1
2 2 2 2 2    2 2 2 2 2    2 2 3 2 2
    2            3            2
  2 7 2        2 1 2        2 1 2
  2 0 2        2 1 2        2 1 2
```

The annihilation of signals SO coming from leftside; S = 4,5,6,7

```
t = 1        t = 2        t = 3
2 2 2 2 2    2 2 2 2 2    2 2 2 2 2
0 S 1 1 1    1 0 1 1 1    1 1 1 1 1
2 2 3 2 2    2 2 3 2 2    2 2 3 2 2
    2            2            2
  2 1 2        2 1 2        2 1 2
  2 1 2        2 1 2        2 1 2
```

The propagation of signals SO coming from rightside

```
t = 1        t = 2        t = 3        t = 4
2 2 2 2 2    2 2 2 2 2    2 2 2 2 2    2 2 2 2 2
1 1 1 S 0    1 1 S 0 1    1 S 0 1 1    S 0 1 1 1
2 2 3 2 2    2 2 3 2 2    2 2 3 2 2    2 2 3 2 2
    2            2            2            2
  2 1 2        2 1 2        2 1 2        2 1 2
  2 1 2        2 1 2        2 1 2        2 1 2
```

Gate desactivation by signal 70 or 60

```
t = 1        t = 2        t = 3
2 2 2 2 2    2 2 2 2 2    2 2 2 2 2
1 1 1 1 1    1 1 1 1 1    1 1 1 1 1
2 2 3 2 2    2 2 3 2 2    2 2 2 2 2
    2            3             
  2 7 2        2 1 2        2 1 2
  2 0 2        2 1 2        2 1 2
```

Fig. 22

Codd type remote controlled gate

Codd-type remote controlled gate. On fig. 22 one can see the activation of this gate, as well as the phenomenon, which in closed state annihilates the signal coming from leftside, but permits from rightside. The opening or dezactivation of the gate can happen by signals 60 or 70. This gate remembers. Its state depends not only on the control signal but on its last own state too. Thus, one can call it bistable gate.

Local gate or valve. On fig. 23 one can see how the local gate is functioning. This is built into the sheathing wall during the construction of the controlled pathway, and works at any time.

The location of the local gate can be along straight or branched way. The latter case is important in the control of sophisticated networks [2].

Gates combinations. By means of gates' combinations, one can build a lot of tools for pathway controls. Such tools are shown by fig. 24. Let's see only one example

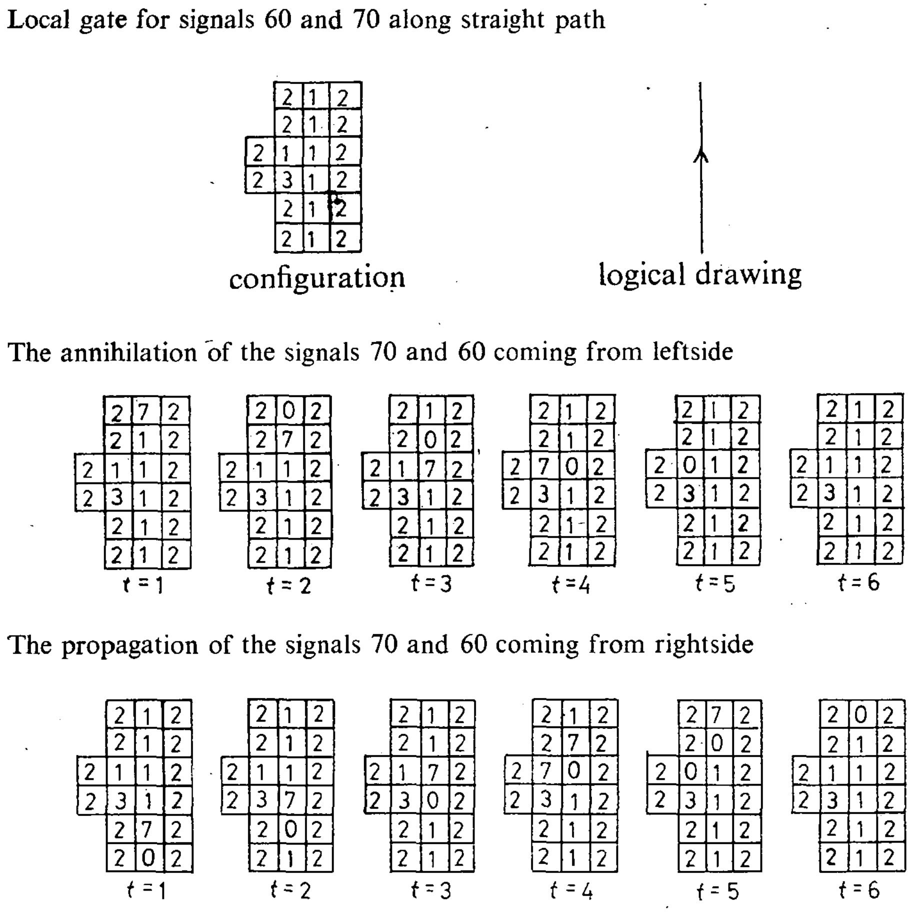
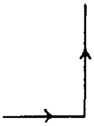

Local gate for signals 60 and 70 along straight path
configuration
logical drawing
The annihilation of the signals 70 and 60 coming from leftside
t = 1
t = 2
t = 3
t = 4
t = 5
t = 6
The propagation of the signals 70 and 60 coming from rightside
t = 1
t = 2
t = 3
t = 4
t = 5
t = 6

Fig. 23a

Rectifier gate in corner and junction

		2	1	2
		2	1	2
2	2	3	1	2
1	1	1	1	2
2	2	2	2	

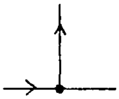
configuration



logical drawing

		2	1	2		
		2	1	2		
2	2	3	1	2	2	2
1	1	1	1	1	1	1
2	2	2	2	2	2	2

configuration



logical drawing

The annihilation of signals SO coming from leftside; S = 4,5,6,7

		2	1	2		
		2	1	2		
2	2	3	1	2	2	2
1	1	1	1	S	0	1
2	2	2	2	2	2	2

$t = 1$

		2	1	2		
		2	1	2		
2	2	3	1	2	2	2
1	1	1	S	0	1	1
2	2	2	2	2	2	2

$t = 2$

		2	1	2		
		2	1	2		
2	2	3	S	2	2	2
1	1	1	0	1	1	1
2	2	2	2	2	2	2

$t = 3$

		2		2		
		2	S	2		
2	2	3	0	2	2	2
1	1	1	1	1	1	1
2	2	2	2	2	2	2

$t=4$

		2	S	2		
		2	0	2		
2	2	3	1	2	2	2
1	1	1	1	1	1	1
2	2	2	2	2	2	2

$t = 5$

		2	0	2		
		2	1	2		
2	2	3	1	2	2	2
1	1	1	1	1	1	1
2	2	2	2	2	2	2

$t = 6$

The propagation of signals coming from rightside

		2	1	2		
		2	1	2		
2	2	3	1	2	2	2
0	**S**	1	1	1	1	1
2	2	2	2	2	2	2

$t = 1$

		2	1	2		
		2	1	2		
2	2	3	1	2	2	2
1	0	**S**	1	1	1	1
2	2	2	2	2	2	2

$t = 2$

		2	1	2		
		2	1	2		
2	2	3	1	2	2	2
1	1	0	**S**	1	1	1
2	2	2	2	2	2	2

$t = 3$.

		2	1	2		
		2	1	2		
2	2	3	**S**	2	2	2
1	1	1	0	**S**	1	1
2	2	2	2	2	2	2

$t = 4$

		2	1	2		
		2	**S**	2		
2	2	3	0	2	2	2
1	1	1	1	0	**S**	1
2	2	2	2	2	2	2

$t = 5$

		2	S	2		
		2	0	2		
2	2	3	1	2	2	2
1	1	1	1	1	0	**S**
2	2	2	2	2	2	2

$t = 6$

Fig. 23/b
Local gates or valves

in details, the socalled monostable gate. This is a Codd-type remote controlled gate which has in its control path a duplicating loop. The first sample of the duplicated signal closes the normally opened gate for 4 tacts only because the second sample of the signal opens it again [3] (see fig. 25).

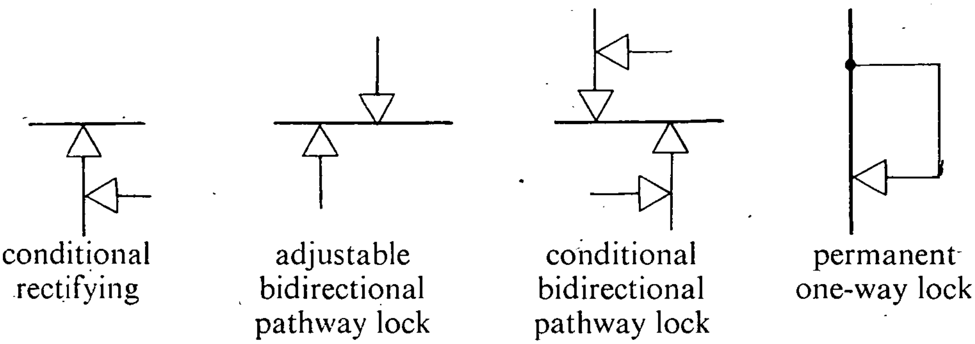
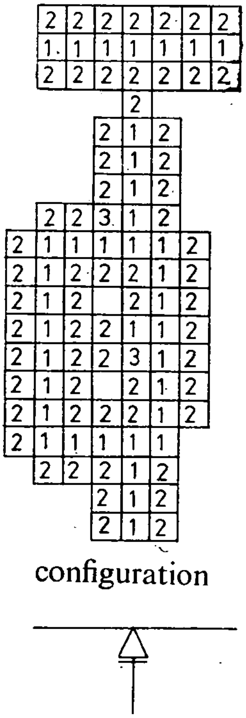

conditional rectifying
adjustable bidirectional pathway lock
conditional bidirectional pathway lock
permanent one-way lock

Fig. 24
Some gate combinations



	2	2	2	2	2	2	2
	1	1	1	1	1	1	1
	2	2	2	2	2	2	2
				2			
			2	1	2		
			2	1	2		
			2	1	2		
	2	2	3	1	2		
2	1	1	1	1	1	2	
2	1	2	2	2	1	2	
2	1	2		2	1	2	
2	1	2	2	1	1	2	
2	1	2	2	3	1	2	
2	1	2		2	1	2	
2	1	2	2	2	1	2	
2	1	1	1	1	1		
	2	2	2	1	2		
			2	1	2		
			2	1	2		

Fig. 25
Monostable gate

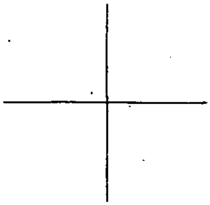
Cross-over. It is a hard problem how to cross the paths in the plane (in two dimensions). Without any protection obviously false signals propagate from one way into the other at the cross point. Since this is one of the favourite subject in the cellular automata litterature the publication of new crossover solutions, usually designed with a let of roundabout ways. E.g. Codd nimself, in his book made a solution, concerning two one-ways, which requires about 1500 cells. Therefore we suppose the result by Dettai [2] is significant giving a crossover which requires 9 cells only for two bidirectional ways, but only for 60 and 70 signals (see fig. 26). The minimal following distance between two signals is 6 [see 2].

6—7 transformer. On fig. 27 one can see a transformer which is usable in one direction only for 60 and 70 signals. For both inputs 60 and 70 the output is 70 [see 3].

Crossover for signals 60 and 70

		2	1	2		
		2	1	2		
2	2	1	1	1	2	2
1	1	1	1	1	1	1
2	2	1	1	1	2	2
		2	1	2		
		2	1	2		

configuration



logical drawing

Functioning

		2	1	2		
		2	1	2		
2	2	1	1	1	2	2
1	1	1	1	1	1	1
2	2	1	1	1	2	2
		2	7	2		
		2	0	2		

$t=1$

		2	1	2		
		2	1	2		
2	2	1	1	1	2	2
1	1	1	1	1	1	1
2	2	1	7	1	2	2
		2	0	2		
		2	1	2		

$t=2$

		2	1	2		
		2	1	2		
2	2	1	1	1	2	2
1	1	1	7	1	1	1
2	2	7	0	7	2	2
		2	1	2		
		2	1	2		

$t=3$

		2	1	2		
		2	1	2		
2	2	1	7	1	2	2
1	1	3	0	3	1	1
2	2	1	1	1	2	2
		2	1	2		
		2	1	2		

$t=4$

		2	1	2		
		2	7	2		
2	2	7	0	7	2	2
1	1	3	1	3	1	1
2	2	1	1	1	2	2
		2	1	2		
		2	1	2		

$t=5$

		2	7	2		
		2	0	2		
2	2	0	1	0	2	2
1	1	3	1	3	1	1
2	2	1	1	1	2	2
		2	1	2		
		2	1	2		

$t=6$

The functioning of the crossover. The minimal following distance for the signals is 6.

		2	0	2		
		2	1	2		
2	2	1	1	1	2	2
1	1	1	1	1	1	1
2	2	1	1	1	2	2
		2	1	2		
		2	1	2		

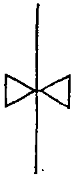
$t=7$

Fig. 26
Bidirectional pathway crossing

The 60, 70 transformer

	2	1	2	
	2	1	2	
2	3	1	3	2
2	1	1	1	2
	2	1	2	
	2	1	2	

configuration



logical drawing

The transformation of signal 60 to signal 70

	2	0	2	
	2	6	2	
2	3	1	3	2
2	1	1	1	2
	2	1	2	
	2	1	2	

$t = 1$

	2	1	2	
	2	0	2	
2	3	7	3	2
2	1	1	1	2
	2	1	2	
	2	1	2	

$t = 2$

	2	1	2	
	2	1	2	
2	3	0	3	2
2	1	7	1	2
	2	1	2	
	2	1	2	

$t = 3$

	2	1	2	
	2	1	2	
2	3	1	3	2
2	7	0	7	2
	2	7	2	
	2	1	2	

$t = 4$

	2	1	2	
	2	1	2	
2	3	1	3	2
2	0	1	0	2
	2	0	2	
	2	7	2	

$t = 5$

	2	1	2	
	2	1	2	
2	3	1	3	2
2	1	1	1	2
	2	1	2	
	2	0	2	

$t = 6$

The propagation of signal 70

	2	0	2	
	2	7	2	
2	3	1	3	2
2	1	1	1	2
	2	1	2	
	2	1	2	

$t = 1$

	2	1	2	
	2	0	2	
2	3	7	3	2
2	1	1	1	2
	2	1	2	
	2	1	2	

$t = 2$

	2	1	2	
	2	1	2	
2	3	0	3	2
2	1	7	1	2
	2	1	2	
	2	1	2	

$t = 3$

	2	1	2	
	2	1	2	
2	3	1	3	2
2	7	0	7	2
	2	7	2	
	2	1	2	

$t = 4$

	2	1	2	
	2	1	2	
2	3	1	3	2
2	0	1	0	2
	2	0	2	
	2	7	2	

$t = 5$

	2	1	2	
	2	1	2	
2	3	1	3	2
2	1	1	1	2
	2	1	2	
	2	0	2	

$t = 6$

The transformer does not works correctly from output

	2	1	2	
	2	1	2	
2	3	1	3	2
2	1	1	1	2
	2	6	2	
	2	0	2	

$t = 1$

	2	1	2	
	2	1	2	
2	3	1	3	2
2	1	6	1	2
	2	0	2	
	2	1	2	

$t = 2$

	2	1	2	
	2	1	2	
2	3	7	3	2
2	6	0	6	2
	2	1	2	
	2	1	2	

$t = 3$

	2	1	2	
	2	7	2	
2	3	0	3	2
2	0	0	0	2
	2	1	2	
	2	1	2	

$t = 4$

Fig. 27

Gene. The configuration called gene is constituted by four cells in state 1 located in square. It is usable only connected to pathway. The gene has three functions: rectifying, signal transformation and signal source. We already have shown now the gene rectifies as a local gate. As a transformer the gene runs as a minimal sized loop. Fig. 28 shows its two possible situations. The multiplying table is identic with the arbitrary sized loop.

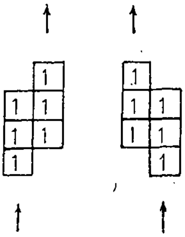


Fig. 28

The signal transformer gene wich is a minimal sized loop

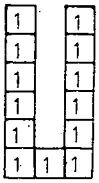


Fig. 29

Sharp turned pathway

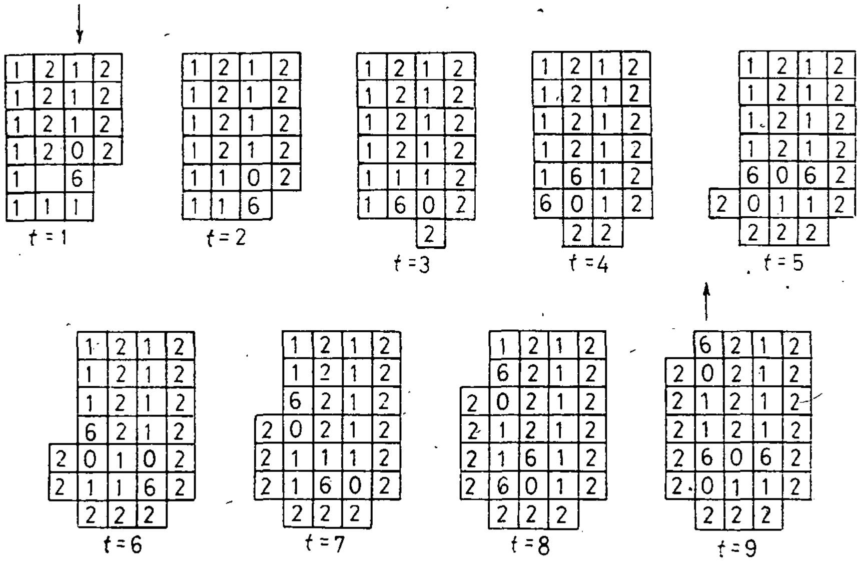
That is also a possibility, to use the gene as a signal source. Let us make a sharp turned way as in fig. 29.

A single 60 signal makes the sheating, but is is enough to create simoultaneously a high intensity signal source giving the sequence

$$\langle 6, 0, 1, 1, 6, 0, 1, 1, \ldots\rangle$$

as in fig. 30.

When constructing genes one has to be careful for the activation which must be done from the opposite direction where the later input will be.



The gene in the sharp turned pathway made by the sheathing signal 60 giving the sequence ⟨6,0,1,1,6,0,1,1, ...⟩

Fig. 30

Signal source

II. The maximum-selector*

The problem

Given a word of length l

$$x = \langle x^1, \ldots, x^l \rangle$$

whose elements $x^1, \ldots, x^l$ are binary words of length k from B^k, where $B = \{0, 1\}$. The binary numbers given by these words are denoted by

$$n^1, n^2, \ldots, n^l,$$

i.e.,

$$n^j = \sum_{i=1}^{k} 2^{k-i} x_i^j, \quad j = 1, \ldots, l, \quad \text{where} \quad x^j = x_1^j, \ldots, x_k^j \quad \text{and} \quad x_i^j = \{0, 1\}.$$

The problem is to find the binary word n representing the maximal one among $n^1, \ldots, n^l$.

The solution

The algorithm. Let us consider an alphabet $C = \{0, 1, S\}$ which is an extension of the binary Boolean alphabet with a "stop signal" $S \cdot C = B \cup \{S\}$. Define the Boolean sum function σ as the following mapping $\sigma \colon S^l \longrightarrow C$ such that for a word $u = \langle u^1, \ldots, u^l \rangle$,

$$\sigma(u) = \begin{cases} S & \text{if} \quad u^1 = \ldots = u^l = S, \\ 1 & \text{if} \quad \exists j \, u^j = 1, \\ 0 - \text{otherwise.} \end{cases}$$

Let us define also the mapping $\tau \colon C^2 \longrightarrow C$ such that for any $(y, y') \in C^2$,

$$\tau(y, y') = \begin{cases} 0 & \text{if} \quad y = y' = 0, \\ 1 & \text{if} \quad y = y' = 1, \\ S - \text{otherwise.} \end{cases}$$

Define also the mapping $C^l \longrightarrow C^l$ such that for the any $u \in C^l$,

$$\theta(u) = \langle \tau(u^1, \sigma(u)), \ldots, \tau(u^l, \sigma(u)) \rangle.$$

The function θ serves as a "stop function". If the relating word contains 1 then θ keeps the element of the word having the value 1, giving stop signal otherwise. If the relating word does not contain 1, leaves the elements unchanged.

Let μ be the mapping $\mu \colon B \times C \longrightarrow C$ for which

$$\mu(b, c) = \begin{cases} S & \text{if} \quad c = S, \\ b & \text{if} \quad c \neq S \end{cases}$$

for any $b \in B$ and $c \in C$.

[*] See [8].

Finally, if $x=\langle x^1, \ldots, x^l\rangle$ is given, define the words $\psi_1(x), \ldots, \psi_k(x)$ of length l by

$$\psi_i(x) = \begin{cases} x_1 & \text{if} \quad i=1, \\ \langle \mu(x_i^1, \theta\psi_{i-1}(x)^1), \ldots, \mu(x_i^l, \theta\psi_{i-1}(x)^l)\rangle & \text{if} \quad i=2,3,\ldots,k. \end{cases}$$

By means of μ, we get a stop signal keeping function, which keeps in the stop signal at each the place in $\psi_i(x)$, where it already appeared for $i'<i$. The algorithm is running in the following way.

First step: Calculate the values

$$\sigma\psi_1 = \sigma(x_1)$$

and

$$\theta\psi_1(x) = \theta(x_1) = \langle \tau(x_1^1, \sigma(x_1)), \ldots, \tau(x_1^l, \sigma(x_1))\rangle$$

and

$$\psi_2(x) = \langle \mu(x_2^1, \theta(x_1)^1), \ldots, \mu(x_2^l, \theta(x_1)^l)\rangle.$$

Second step: Calculate the values

$$\sigma\psi_2(x)$$

and

$$\theta\psi_2(x) = \langle \tau(\psi_2(x)^1, \sigma\psi_2(x)), \ldots, \tau(\psi_2(x)^l, \sigma\psi_2(x))\rangle$$

and

$$\psi_3(x) = \langle \mu(x_3^1, \theta\psi_2(x)^1), \ldots, \mu(x_3^l, \theta\psi_2(x)^l)\rangle.$$

i-th *step:* Calculate the values

$$\sigma\psi_1(x)$$

and

$$\theta\psi_i(x) = \langle \tau(\psi_i(x)^1, \sigma\psi_i(x)), \ldots, \tau(\psi_i(x)^l, \sigma\psi_i(x))\rangle$$

and

$$\psi_{i+1}(x) = \langle \mu(x_{i+1}^1, \theta\psi_i(x)^1), \ldots, \mu(x_{i+1}^l, \theta\psi_i(x)^l)\rangle.$$

After the k-th step the result in the case

$$(\exists j')(\forall i)(\tau^{j'} \neq S)$$

is

$$n = \langle \tau_1^{j'}, \ldots, \tau_k^{j'}\rangle.$$

Example: $x=\langle\langle 1,1,0,1,0,1\rangle, \langle 1,1,0,0,1,1\rangle, \langle 1,1,0,1,0,0\rangle\rangle$, $(l=3,\ k=6)$.

First step:

$$x_1 = \langle 1,1,1\rangle, \quad \psi_1(x) = x_1, \quad \sigma\psi_1(x) = 1,$$
$$\theta(x_1) = \langle \tau(1,1), \tau(1,1), \tau(1,1)\rangle = \langle 1,1,1\rangle;$$
$$x_2 = \langle 1,1,1,\rangle, \quad \psi_2(x) = \langle \mu(1,1), \mu(1,1), \mu(1,1\rangle = \langle 1,1,1\rangle.$$

Second step:

$$\sigma\psi_2(x) = 1,$$
$$\theta(\psi_2(x)) = \langle \tau(1,1), \tau(1,1), \tau(1,1)\rangle = \langle 1,1,1\rangle,$$
$$x_3 = \langle 0,0,0\rangle, \quad \psi_3(x) = \langle \mu(0,1), \mu(0,1)\, \mu(0,1)\rangle = \langle 0,0,0\rangle.$$

Third step:

$$\sigma\psi_3(x) = 0,$$

$$\theta\psi_3(x) = \langle\tau(0,0), \tau(0,0), \tau(0,0)\rangle = \langle 0,0,0\rangle,$$

$$x_4 = \langle 1,0,1\rangle, \quad \psi_4(x) = \langle\mu(1,0), \mu(0,0), \mu(1,0)\rangle = \langle 1,0,1\rangle.$$

Fourth step:

$$\sigma\psi_4(x) = 1,$$

$$\theta\psi_4(x) = \langle\tau(1,1), \tau(0,1), \tau(1,1)\rangle = \langle 1,S,1\rangle,$$

$$x_5 = \langle 0,1,0\rangle, \quad \psi_5(x) = \langle\mu(0,1), \mu(1,S), \mu(0,1)\rangle = \langle 0,S,0\rangle.$$

Fifth step:

$$\sigma\psi_5(x) = 0,$$

$$\theta\psi_5(x) = \langle\tau(0,0), \tau(S,0), \tau(0,0)\rangle = \langle 0,S,0\rangle,$$

$$x_6 = \langle 1,1,0\rangle, \quad \psi_6(x) = \langle\mu(1,0), \mu(1,S), \mu(0,0)\rangle = \langle 1,S,0\rangle.$$

Sixth step:

$$\sigma\psi_6(x) = 1,$$

$$\theta\psi_6(x) = \langle\tau(1,1), \tau(S,1), \tau(0,1)\rangle = \langle 1,S,S\rangle.$$

Since $j'=1$ thus

$$n = x^1 = \langle 1,1,0,1,0,1\rangle.$$

The solution principle in Codd-ICRA automaton. The selection of the maximal number happens in Codd-ICRA automaton. The solution principle is the following: the numbers are introduced into the cellular space simoultaneously; they are propagating along parallel pathways. After the inputs each number is duplicated and one of their sample arrives into a gate system as information. The second samples — after delay — go further in their channels as data. In these channels they annihilate or move on, according to the situation of the already opened or closed gate system. At all those places where the figures with highest local value are 1 the numbers go further, while at places where they are 0 the numbers are annihilated. In the second step, where the figures with the second highest local value are 1, among the remained numbers, those numbers go further, and the numbers are annihilated if they are 0, and so an, until the k-th figures, which have the lawest local value (the coefficients of 2^0). Exception is if at the actual step among the remained numbers 0-s are everywhere. In this case, all the remained numbers go further. The outputs of the channels are joined, so at the final, single output one gets the maximal number (even if the same one appeared at several channels).

The maximal number selector is called MAXEL.

The principal plan. The principal plan shows the parallel pathways, the duplication, the delay and the selector units, forming the functions τ and μ. At last we signed the feedback for the Boolean sum, and the output giving the maximal number (see fig. 31).

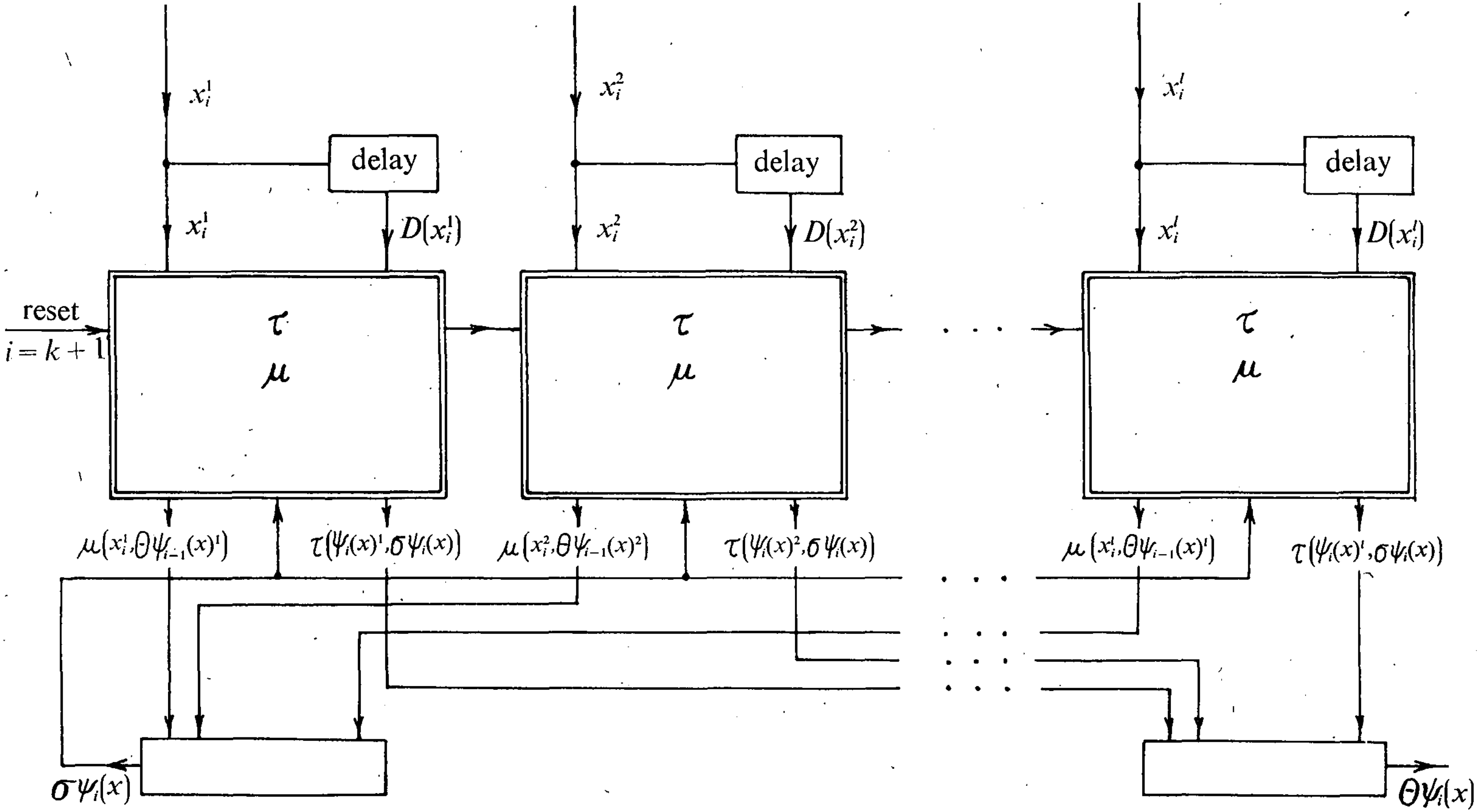

x_i^1
x_i^2
x_i^l
delay
delay
delay
x_i^1
$D(x_i^1)$
x_i^2
$D(x_i^2)$
x_i^l
$D(x_i^l)$
reset
$i=k+1$
τ
μ
τ
μ
τ
μ
$\mu(x_i^1, \Theta\psi_{i-1}(x)^1)$
$\tau(\psi_i(x)^1, \sigma\psi_i(x))$
$\mu(x_i^2, \Theta\psi_{i-1}(x)^2)$
$\tau(\psi_i(x)^2, \sigma\psi_i(x))$
$\mu(x_i^l, \Theta\psi_{i-1}(x)^l)$
$\tau(\psi_i(x)^l, \sigma\psi_i(x))$
$\sigma\psi_i(x)$
$\Theta\psi_i(x)$

Fig. 31
l-MAXEL
Principal plan

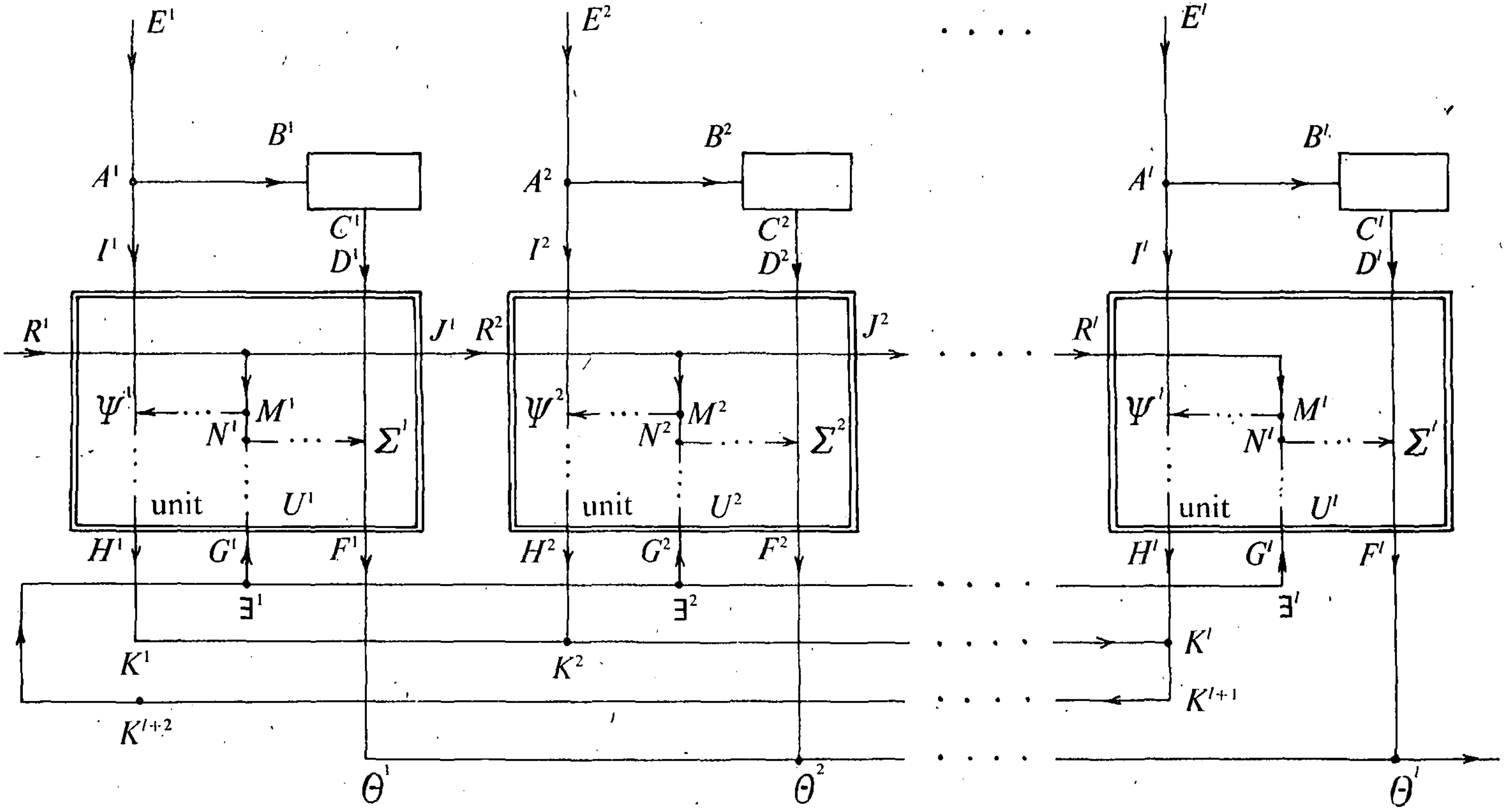


Fig. 32
l-MAXEL
System scheme

System scheme. The structure of the system scheme is identical with the principal plan, apart from the mapping of the input and output points. The selecting unit here is also almost a black box only, just showing the pathway control role of the gates located in the unit (see fig. 32).

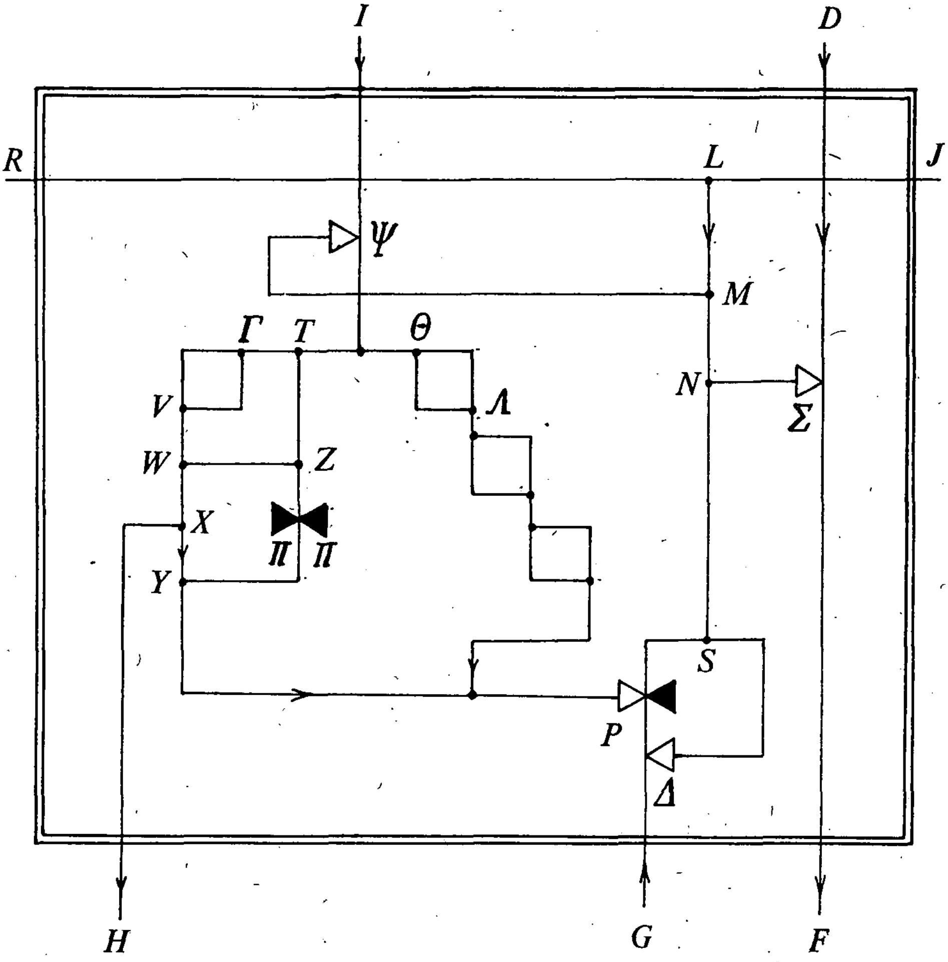


Fig. 33
The U^j selecting unit of *l*-MAXEL
Logical drawing

Logical drawing. Figure 33 shows the complete logical structure of a selecting unit.

The several cases of the functioning are the following.

α) $\sigma(x_i) = 1$

α′) $x_1^j = 1$ (the signal content is 60).

Inputs The name of the cell	Mode of application	Signal content
I	x_i^j information	60 11 70
D	$D(x_i^j)$ data	60 11 70
G	$\sigma(x_i)$	70 11
R	reset; active once at each $(k+1)$-th time step	60 11
Outputs The name of the cell	Mode of application	Signal content
H	x_i^j information	60 11
F	$D(x_i^j)$ data	60 11 70
J	reset; active once at each $(k+1)$-th time step	60 11

In this case four events happen.

I. The regeneration of gate P

$$x_i^j: \quad \text{goes via} \quad I \to \Psi \to \theta \to \Lambda \to P.$$

The loops $\theta\Lambda$ and the two next assure, that in each case one gets signal 60 at cell P, thus if it was in state 2 then it remains there, and if it was in state 3, it changes to state 2.

II. Information

$$\begin{tikzcd} & & & \Gamma \arrow[r] & V \arrow[dr] & & & \\ x_i^j\text{: goes via} \quad I \arrow[r] & \Psi \arrow[r] & T \arrow[ur] \arrow[dr] & & & W \arrow[r] & X \arrow[r] & H. \\ & & & Z \arrow[urr] & & & & \end{tikzcd}$$

By this way the signal can not arrive at cell P since the transformer Π transforms both 60 and 70 to 70, and this joins at cell Y with 60, where it annihilates.

III. The sum $\sigma(x_i)$ enters into the unit at cell G and annihilates at gate P.

IV. Data

$$D(x_i^j): \quad \text{goes via} \quad D \to \Sigma \to F$$

α'') $\qquad x_i^j=0$ (the signal content is 70).

In this case three events happen.

I. The regeneration of gate P

$$x_i^j: \quad \text{goes via} \quad I \to \Psi \to \theta \to \Lambda \to P.$$

The signal 60 does not change the state of P cell, if it is 2, but it will be changed to 2, if it is 3.

II. Information

$$x_i^j: \quad \text{goes via} \quad I \to \Psi \to T \to Z \to \Pi \to Y \to P,$$

and changes by the signal 70, the state of P cell from 2 to 3. The signal does not arrive at point X, thus nor at output H, since 40 arising from the end of the loop ΓV joins with 70 coming from TZ at point W, and because of $70 \times 40 = 11$ annihilates here.

III. The sum $\sigma(x_i)$ enters into the unit at cell G and goes via

$$\begin{tikzcd} & & & \Delta & & \\ G \arrow[r] & P \arrow[r] & S \arrow[ur] \arrow[dr] & & \Sigma & \\ & & & N \arrow[ur] \arrow[r] & M \arrow[r] & \Psi. \end{tikzcd}$$

This $\sigma(x_i)$ places to state 3 in the gate cells Δ, Σ and Ψ respectively, which have had previous state 2. Afterwards, the inputs of the selecting unit are already closed for the signals

$$x_{i+1}^j, \quad D(x_i^j) \quad \text{and} \quad \sigma(x_{i+1}),$$

β)

$$\sigma(x_i) = 0,$$

$$x_i^j = 0 \quad \text{(the signal context is 70).}$$

In this case three events happen.

I. The regeneration of gate P

$$x_i^j: \quad \text{goes via} \quad I \to \Psi \to \theta \to \Lambda \to P.$$

The signal 60 doesn't change the state of P cell, if it's 2, but it will be changed to 2, if it is 3.

II. Information

$$x_i^j: \quad \text{goes via} \quad I \to \Psi \to T \to Z \to \Pi \to Y \to P,$$

and changes the state of P cell to 2 from 3. At points X and H no signal arrives because it was annaihilated at W.

III. Data

$$D(x_i^j): \quad \text{goes via} \quad D \to \Sigma \to F.$$

The 0—1 *configuration of the selector unit's.* Figure 34 shows the 0—1 configuration of the unit described in the previous point.

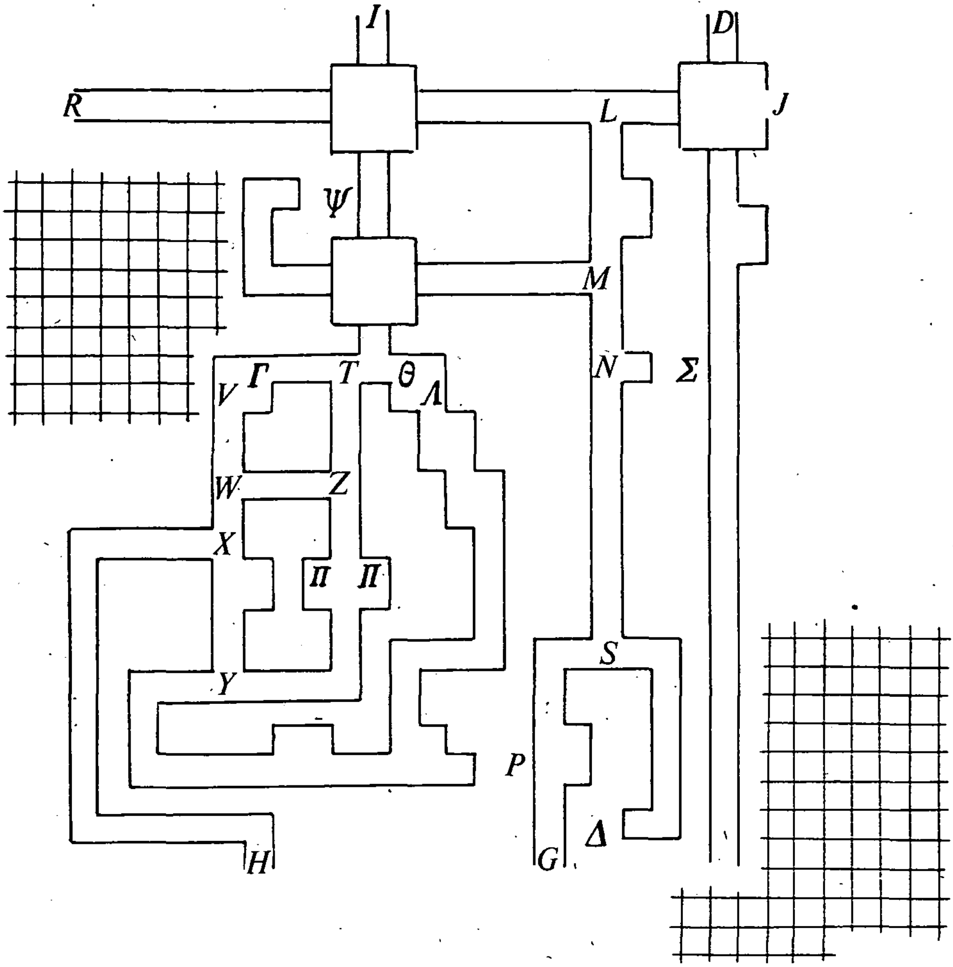


Fig. 34
The U^j selecting unit of *l*-MAXEL
0-1 configuration

The logical drawing of l-MAXEL's. $l=3$ is the smallest value of l for which one can show the slight assimmetries of the extremal selector units. Apart from these extremities, the construction of the MAXEL is modular (building bricks system).

The 0—1 *configuration of l-MAXEL's.* Figure 36 shows for $l=3$ and arbitrary k the 0—1 configuration. On the upper part of the figure one can see the input synchronization.

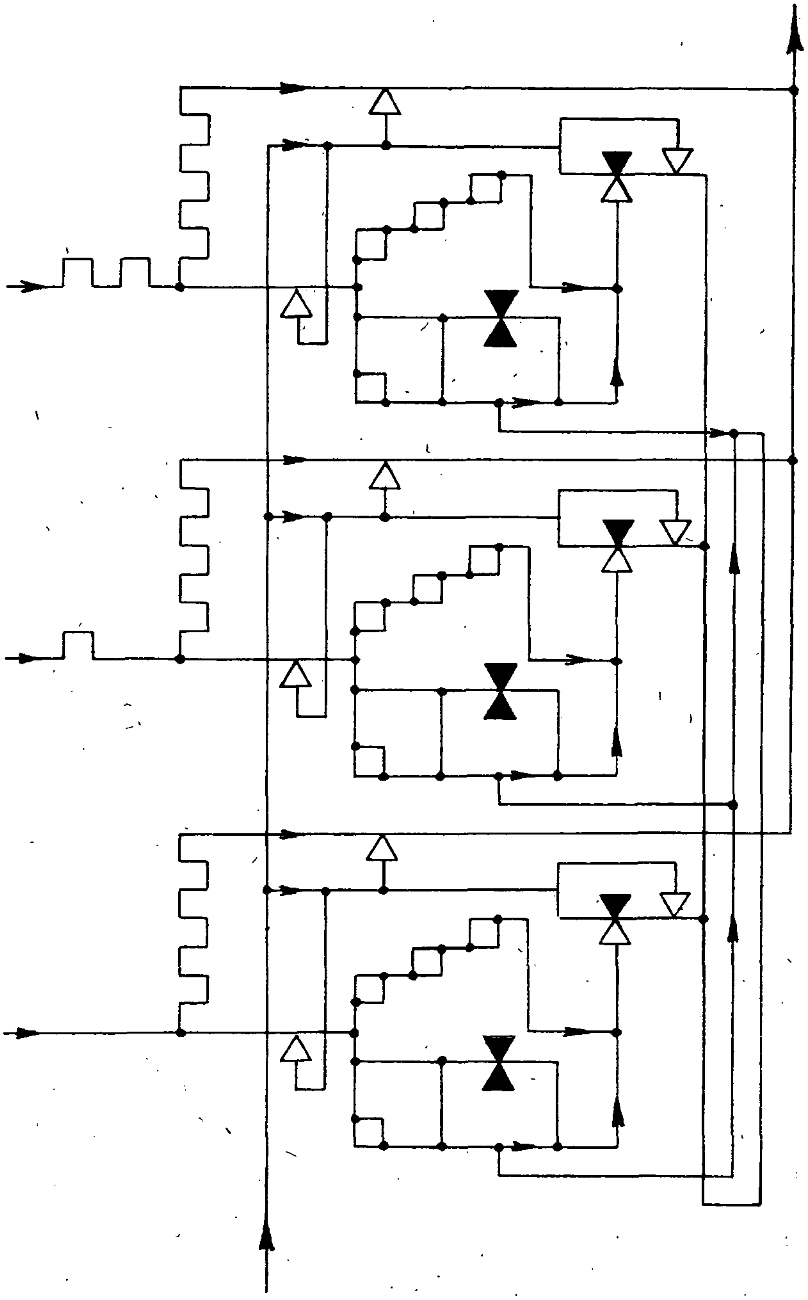


Fig. 35

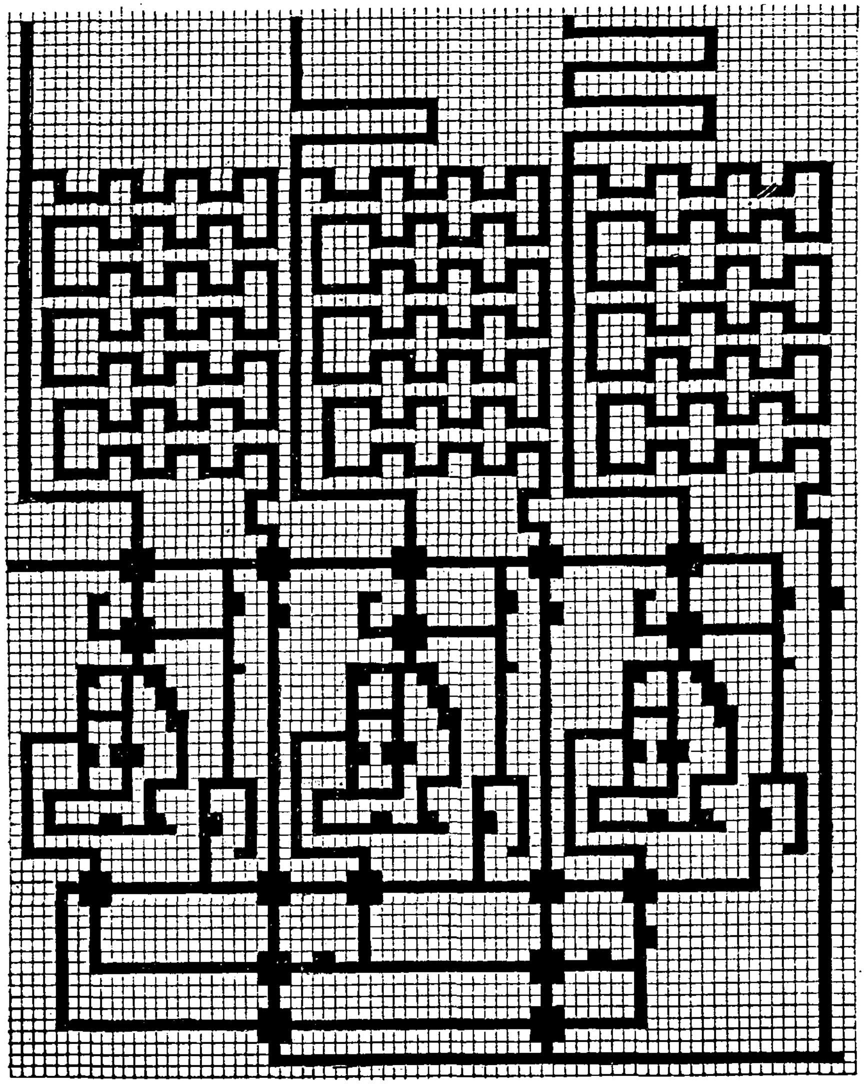


Fig. 36

l-MAXEL for $l = 3$
0-1 configuration

Coding

Inputs.

The Boolean 0 is coded by 70

The Boolean 1 is coded by 60

Outputs from the point of junctions $0^1, \dots, 0^l$.

The Boolean 1 is coded by 60

The Boolean 0 is coded by $\left\{\begin{array}{l}70\\40\\11\end{array}\right.$

If $N_j(x)=\{j|x_i^j=n\}$,

$\# N_j(x) = 1 \pmod 3$ the Boolean 0 is coded by 70,

$\# N_j(x) = 2 \pmod 3$ the Boolean 0 is coded by 40,

$\# N_j(x) = 3 \pmod 3$ the Boolean 0 is coded by 11.

Working times. The distance betwen cells A and B expressed by the number of cells between them is denoted $\overline{AB}$, when the shortest possible walking on is chosen from A to B. The retard between points I^j, I^{j+1} and D^j, D^{j+1} consists always 24 tacts. The retard T_{ID} between points I^j, D^j is the following

$$T_{ID} = \overline{I^1H^1K^1K^2\dots K^lK^{l+1}K^{l+2}\exists^1G^1\Sigma^1} - \overline{D^1\Sigma^1} - 1.$$

$$\overline{I^1H^1K^1} = 55.$$

$$\overline{K^1K^2\dots K^l} = (l-1)24.$$

$$\overline{K^lK^{l+1}} = 4.$$

$$\overline{K^{l+1}K^{l+2}} = (l-1)24.$$

$$\overline{K^{l+2}\exists^1} = 29.$$

$$\overline{\exists^1G^1\Sigma^1} = 24.$$

$$\overline{D^1\Sigma^1} = 12.$$

$$T_{ID} = 48(l-1)+99.$$

The periodicity p, in other words the distance between signals x_i^j and x_{i+1}^j, is

$$p = \overline{I^1H^1K^1K^2\dots K^lK^{l+1}K^{l+2}\exists^1G^1\Psi^1} - \overline{I^1\Psi^1} - 1.$$

$$\overline{\exists^1G^1\Psi^1} = 42.$$

$$\overline{I^1\Psi^1} = 6.$$

$$p = 48(l-1)+123.$$

The $T(n)$ working time for selecting the m maximal number is the running time for the first figure — n_1 — plus the time of the remained figures.

$$T(n) = T(n_1) + (k-1)p.$$

$$T(n_1) = T_{ID} + \overline{D^1F^1\Theta^1} + \overline{\Theta^1\Theta^2 \dots \Theta^l}.$$

$$\overline{D^1F^1\Theta^1} = 47.$$

$$\overline{\Theta^1\Theta^2} \dots \Theta^l = (l-1)24.$$

$$T(n_1) = 72(l-1) + 146.$$

$$T(n) = 24\,l(1+2k) + 75\,k + 1.$$

For example in the case

$$k=4, \quad l=3: \quad T_{ID} = 195, \quad p = 219, \quad T(n) = 733;$$

$$k=10, \quad l=10: \quad T_{ID} = 531, \quad p = 555, \quad T(n) = 5791.$$

One has to calculate the necessary time for the reactivization, by this way. The effect of putting in the gate Ψ last figure of the first number — x_k^1 — is

$$kp + \overline{I^1\Psi^1}.$$

By substracting from this the propagating time of the reactivating signal, one gets its starting time tact t_{reac}: minimal following distance,

$$t_{\text{reac}} = kp + \overline{I^1\Psi^1} - \overline{R^1L^1M^1\Psi^1} + \text{ minimal following ditance}$$

minimal following distance = 6,

$$\overline{I^1\Psi^1} = 7,$$

$$\overline{R^1L^1M^1\Psi^1} = 44.$$

Since

$$t_{\text{reac}} = kp - 31$$

the necessary time interval between two selecting procedure T_{reac} is

$$T_{\text{reac}} = kp + \overline{I^1\Psi^1} + \text{ minimal following distance } - \overline{I^1\Psi^1} - kp,$$

$$T_{\text{reac}} = 6.$$

Size. Let S^l be the size of an l-MAXEL expressed in terms of the number of its constituting cells. Then

$$S^l = [l(\overline{R^lJ^l}) + 1](\overline{E^lI^l} + \overline{I^lH^l} + \overline{F^l\Theta^l}),$$

$$\overline{R^lJ^l} = 24,$$

$$\overline{E^lI^l} = 32 + (l-3)6, \quad l \geqq 3,$$

$$\overline{I^lH^l} = 30,$$

$$\overline{F^l\Theta^l} = 17,$$

$$S^l = (24\,l + 1)[79 + (l-3)6], \quad l \geqq 3.$$

For example in the case

$$l = 3, \quad k = 4: \quad S^3 = 73\times 79,$$
$$l = 10, \quad k = 10: \quad S^{10} = 241\times 121.$$

Conclusion concerning the design

The advantages of the solution. 1. The data according to x_i^j coefficients are propagating along parallel pathways in the l-MAXEL, which permits a high simoultaneity within the frames of the task, making the profit from the two-dimensionality of Codd-ICRA cellular space.

2. The size of the l-MAXEL depends on the number l of numbers only, it is completely independent of the length k of the numbers.

3. The dependence of the working time the l-MAXEL on both l and k is very simple — linear — function.

4. If $l' > l$, l-MAXEL is easily extend ableby the insertion of new selector units.

The conditions of the solution. I. Inputs

1. For the inputs x_i^j one has to assure a constant speed which is independent of i and j.

2. If the inputs are not synchronized, one has to apply 24 tacts retard between two neighbouring j-th and $(j+1)$-th inputs, or build in input synchronization blocks.

3. The p periodicity of the signals as a function of l is

$$p = 48(l-1)+123.$$

II. The output periodicity is p, too.

III. One has to give in advance the values of k and l.

IV. Apart from the l inputs the l-MAXEL needs an $(l+1)$-th input for the reactivization after a whole selecting procedure.

The required reactivization time is

$$t_{\text{reac}} = kp-31.$$

Summary

After becoming acquainted with the basic techniques, and concepts of cellular automaton and cellular space we present a machine, which is able to select the maximal one among the numbers simoultaneously introduced into it.

NATIONAL CENTRE FOR EDUCATIONAL TECHNOLOGY
VESZPRÉM, HUNGARY

[1] Sometimes differently from the usual way of speaking, we say cellular *space* for the five-tuple $\mathfrak{A}=(I, v, L, S, \perp)$ and in these cases by a cellular *automaton* we mean the ordered pair $(\mathfrak{A}, q_0)$, where q_0 is a quiescent state.

[2] This published solution is a corrected version by means of the simulation experiences on PDP 10.

TABLE *Codd-ICRA*

s	s'	$s\perp s'$	s	s'	$s\perp s'$	s	s'	$s\perp s'$	s	s'	$s\perp s'$	s	s'	$s\perp s'$	s	s'	$s\perp s'$	s	s'	$s\perp s'$
0	0006	2	0	1214	1	0	1662	1	1	0262	6	1	1242	4	1	2255	5	2	0306	3
0	0007	3	0	1215	1	0	1666	1	1	0263	6	1	1243	4	1	2263	6	2	0307	3
0	0015	3	0	1216	1	0	1717	1	1	0272	7	1	1244	5	1	2266	6	2	0711	1
0	0016	2	0	1217	1	0	1722	1	1	0273	7	1	1252	5	1	2273	7	2	1117	1
0	0025	2	0	1222	7	0	1727	1	1	0342	4	1	1253	5	1	2277	3	2	1232	3
0	0026	2	0	1223	1	0	1772	1	1	0343	4	1	1255	6	1	2324	4	2	2324	3
0	0042	2	0	1224	1	0	1777	1	1	0352	5	1	1262	6	1	2325	5	2	2325	3
0	0051	3	0	1225	1	0	2226	2	1	0353	5	1	1263	6	1	2326	6	2	2326	3
0	0061	2	0	1226	1	0	2266	2	1	0362	6	1	1266	6	1	2327	7	2	2327	3
0	0062	2	0	1227	1	1	0004	0	1	0363	6	1	1272	7	1	2334	4	3	0002	2
0	0066	2	0	1232	6	1	0006	6	1	0372	7	1	1273	7	1	2335	5	3	0006	1
0	0106	2	0	1235	1	1	0007	3	1	0373	7	1	1277	4	1	2336	6	3	0025	1
0	0107	3	0	1242	1	1	0014	0	1	0411	0	1	1343	7	1	2337	7	3	0026	0
0	0116	2	0	1244	1	1	0016	6	1	0606	6	1	1353	7	1	2343	4	3	0027	0
0	0126	2	0	1252	1	1	0017	2	1	0611	6	1	1363	7	1	2353	5	3	0042	1
0	0161	2	0	1253	1	1	0024	4	1	0616	6	1	1373	7	1	2363	6	3	0062	0
0	0162	2	0	1255	1	1	0026	6	1	0621	6	1	1422	4	1	2373	7	3	0072	0
0	0166	2	0	1262	1	1	0036	6	1	0622	6	1	1424	5	1	2424	4	3	0102	2
0	0206	2	0	1266	1	1	0041	0	1	0626	6	1	1442	5	1	2433	4	3	0103	0
0	0207	3	0	1272	1	1	0052	5	1	0661	6	1	1522	5	1	2525	5	3	0106	4
0	0213	1	0	1273	1	1	0061	6	1	0722	7	1	1525	6	1	2533	5	3	0107	7
0	0226	2	0	1277	1	1	0062	6	1	1114	0	1	1552	6	1	2626	6	3	0111	1
0	0227	1	0	1313	1	1	0063	6	1	1115	6	1	1616	6	1	2633	6	3	0162	0
0	0261	2	0	1322	1	1	0066	6	1	1116	6	1	1622	6	1	2727	7	3	0172	0
0	0262	2	0	1324	1	1	0071	7	1	1117	7	1	1626	6	1	2733	7	3	0261	0
0	0272	1	0	1342	1	1	0104	0	1	1124	4	1	1662	6	1	3334	4	3	0271	0
0	0363	1	0	1352	1	1	0105	5	1	1125	5	1	1666	3	1	3335	5	3	1111	1
0	0611	2	0	1362	1	1	0106	6	1	1126	6	1	1717	7	1	3336	6	3	1232	2
0	0621	2	0	1363	1	1	0107	2	1	1127	7	1	1722	7	1	3337	7	3	2324	2
0	0622	2	0	1372	1	1	0114	0	1	1142	4	1	1727	4	2	0006	0	3	2325	2
0	0626	2	0	1373	1	1	0116	6	1	1152	5	1	1772	4	2	0007	1	3	2326	2
0	0661	2	0	1422	1	1	0126	6	1	1162	6	1	1777	3	2	0017	1	3	2327	2
0	0722	1	0	1424	1	1	0141	0	1	1166	3	1	2224	4	2	0025	3	4	s'	1 (a)
0	1116	2	0	1432	1	1	0161	6	1	1172	7	1	2225	5	2	0042	3			0 (b)
0	1124	1	0	1442	1	1	0162	6	1	1177	3	1	2226	6	2	0071	1	5	s'	1 (a)
0	1125	1	0	1522	1	1	0166	6	1	1214	4	1	2227	7	2	0106	0			0 (b)
0	1126	1	0	1523	1	1	0226	6	1	1215	5	1	2234	4	2	0107	1	6	s'	1 (c)
0	1127	1	0	1525	1	1	0227	7	1	1216	6	1	2235	5	2	0117	1			0 (d)
0	1142	1	0	1532	1	1	0242	4	1	1217	7	1	2236	6	2	0142	3	7	s'	1 (c)
0	1152	1	0	1552	1	1	0243	4	1	1224	4	1	2237	7	2	0171	1			0 (d)
0	1162	1	0	1616	1	1	0252	5	1	1225	5	1	2243	4	2	0206	0			
0	1166	2	0	1622	1	1	0253	5	1	1226	6	1	2244	4	2	0207	3			
0	1212	1	0	1626	1	1	0261	6	1	1227	7	1	2253	5	2	0251	3			

s : the own state of the cell
$s'=\langle s'_1, s'_2, s'_3, s'_4\rangle$: the neighbourhood states
$s\perp s'$: the next state of the cell

(*a*) : if s' does not contain 1
(*b*) : if s' contains 1
(*c*) : if s' does not contain any even number
(*d*) : if s' contains an even number

References

[1] Codd, E. F., *Cellular automata,* ACM monograph series, Academic Press, 1968.

[2] Dettai, I., *Effectivity problems and suggestions concerning Codd-ICRA cellular space,* KGM ISZSZI Technical Report, 1974.

[3] Fay, G., *A basic course on cellular automata,* University Lecture Notes, Budapest University of Eötvös Loránd (in Hungarian) 1975.

[4] Neumann von, J., *Theory of self-reproducing automata,* University of Illinois Press, Urbana Edited and completed by A. W. Burks, 1966.

[5] Riguet, J., Automates cellulaires à bord et automates Codd-ICRA, I. *Comptes rendus de l'academis les sciences de Paris,* t 282, Serie A., 1976, pp. 167—170.

[6] Riguet, J., Automates cellulaires à bord et automates Codd-ICRA, II. *Comptes rendus de l'academie les sciences de Paris,* t 282 Serie A, 1976, pp. 239—242.

[7] Szőke, M. (Mrs), *Destruction in Codd-ICRA,* Graduate thesis in mathematics, Budapest University of Eötvös Loránd (in Hungarian), 1975.

[8] Takacs, D. V. (Mrs), *Un automate cellulaire Codd-ICRA pour la recherche du maximum de l nombres,* These présentée a Université René Descartes de Paris Sorbonne, Doctorat d'Université, 1975.

(Received Oct. 13, 1976)

Characteristically free quasi-automata

By I. Babcsányi

In [2], [3] and [4] we dealt with the cyclic state-independent, well-generated group-type and reversible state-independent quasi-automata, respectively. In this paper we investigate a more general class of quasi-automata: the characteristically free quasi-automata. For the notions and notations which are not defined here we refer the reader to [3] and [7].

1. General preliminaries

The A-sub-quasi-automaton $\mathbf{A}_1=(A_1, F, \delta_1)$ of the quasi-automaton $\mathbf{A}=(A, F, \delta)$ is the *kernel* of $\mathbf{A}$ if

$$A_1 = \langle \delta(a,f) | a \in A, f \in F \rangle. \tag{1}$$

$\mathbf{A}$ is *well-generated* if $A=A_1$. In [3] and [4] the well-generated quasi-automaton is called simply generated quasi-automaton. $\bar{F}^A$ (or simply $\bar{F}$) denotes the characteristic semigroup of $\mathbf{A}$, and $\bar{f}^A$ (or $\bar{f}$) is the element of $\bar{F}^A$ represented by $f(\in F)$.

The well-generated quasi-automaton $\mathbf{A}=(A, F, \delta)$ is said to be *characteristically free* if there exists a generating system G of $\mathbf{A}$ such that

$$\delta(a,f) = \delta(b,g) \Rightarrow a = b, \quad \bar{f} = \bar{g} (a, b \in G; f, g \in F). \tag{2}$$

G is called a *characteristically free generating system* of $\mathbf{A}$, and its elements are called *characteristically free generating elements* of $\mathbf{A}$.

We note that every characteristically free generating system is minimal.

Theorem 1. *The quasi-automaton* $\mathbf{A}=(A, F, \delta)$ *is characteristically free if and only if* $\mathbf{A}$ *is a direct sum of isomorphic characteristically free cyclic quasi-automata.*

Proof. It can easily be seen that the subsets $A_b=\langle \delta(b,f) | f \in F \rangle$ $(b \in G)$ of A form a partition on A, where G is a characteristically free generating system of $\mathbf{A}$. Quasi-automata $\mathbf{A}_b=(A_b, F, \delta_b)$ $(b \in G)$ are characteristically free cyclic quasi-automata. Let b_1, b_2 $(\in G)$ be arbitrary generating elements. The mapping

$$\varphi_{b_1, b_2} : \delta(b_1, f) \to \delta(b_2, f) \quad (f \in F) \tag{3}$$

is an isomorphism of $\mathbf{A}_{b_1}$ onto $\mathbf{A}_{b_2}$.

Conversely, it is clear that the direct sum of isomorphic characteristically free cyclic quasi-automata is characteristically free.

Theorems 1. and 2. are equivalent for A-finite well-generated quasi-automata.

Theorem 2. *The A-finite well-generated quasi-automaton $\mathbf{A}=(A, F, \delta)$ is characteristically free if and only if there exists a generating system G of $\mathbf{A}$ such that*

$$|A| = |G|\cdot O(\bar{F}).$$

(In this case G is a characteristically free generating system.)

Proof. Let G be a generating system of the A-finite well-generated quasi-automaton $\mathbf{A}=(A, F, \delta)$ such that

$$|A| = |G|\cdot O(\bar{F}).$$

Since $|A_b|\leqq O(\bar{F})$ $(b\in G)$ and $A=\bigcup_{b\in G} A_b$ therefore

$$|A| \leqq \sum_{b\in G} |A_b| \leqq |G|\cdot O(\bar{F}) = |A|,$$

thus

$$|A| = \sum_{b\in G} |A_b|.$$

This means that A_b $(b\in G)$ form a partition on A and $|A_b|=O(\bar{F})$. It is evident that the mapping $\bar{f}\to\delta(b,f)$ $(f\in F)$ is one-to-one. Therefore, the quasi-automata $\mathbf{A}_b$ $(b\in G)$ are characteristically free, cyclic, and for every pair b_1, b_2 $(\in G)$, $\mathbf{A}_{b_1}\cong\mathbf{A}_{b_2}$. By Theorem 1, the quasi-automaton $\mathbf{A}=(A, F, \delta)$ is characteristically free, and G is a characteristically free generating system of $\mathbf{A}$.

The necessity of this theorem follows from Theorem 1.

Lemma 1. (I. Babcsányi [3].) *Arbitrary two minimal generating systems of a well-generated quasi-automaton have the same cardinality.*

Corollary 1 and 2 follow immadeately from Theorem 2 and Lemma 1.

Corollary 1. *The A-finite cyclic quasi-automaton $\mathbf{A}=(A, F, \delta)$ is characteristically free if and only if $|A|=O(\bar{F})$.*

The necessity of Corollary 1 is true for infinite quasi-automata; thus we get the following result:

Theorem 3. *If the cyclic quasi-automaton $\mathbf{A}=(A, F, \delta)$ is characteristically free then $|A|=O(\bar{F})$.*

It should be noted that the converse of Theorem 3 does not hold. Indeed, in Example 1 for the quasi-automaton $\mathbf{A}=(A, F_1(X), \delta)$ we show that $|A|=O\big(\overline{F_1(X)}\big)$, but $\mathbf{A}$ is not characteristically free.

Example 1. $A=\langle 1; 2; 3; \dots\rangle$, $X=\langle x, y\rangle$,

$$\delta(1,x)=2,\quad \delta(1,y)=1,\quad \delta(i,x)=\delta(i,y)=i+1 \quad (i=2,3,\dots).$$

It can be seen that $\overline{F_1(X)}=\langle\overline{y^i x^j}\mid i,j=0,1,2,\dots\rangle$. (We note that $x^0=y^0$ is the empty word.)

Corollary 2. *Every minimal generating system of an A-finite characteristically free quasi-automaton is characteristically free.*

In the following example it is shown that Corollary 2 does not hold for infinite quasi-automata.

Example 2. Let N be the set of natural numbers, $A = N \times N$ and $X = \langle x, y \rangle$. The definition of next state function δ is the following:

$$\delta((i, 1), x) = (i, 2),$$

$$\delta((i, 2), x) = \delta((i, 4), x) = (i, 3),$$

$$\delta((i, 2j+1), x) = \delta((i, 2j+4), x) = (i, 2j+3),$$

$$\delta((i, 1), y) = (i+1, 1),$$

$$\delta((i, 2), y) = \delta((i, 4), y) = (i, 1),$$

$$\delta((i, 2j+1), y) = \delta((i, 2j+4), y) = (i, 2j+2) \quad (i, j = 1, 2, 3, \ldots).$$

The quasi-automaton $\mathbf{A} = (A, F(X), \delta)$ is cyclic. $\langle (1, j) \rangle$ $(j = 1, 2, 3, \ldots)$ are minimal generating systems, but only $\langle (1, 1) \rangle$ is characteristically free.

Lemma 2. *The characteristic semigroup of every characteristically free quasi-automaton has a left identity element.*

Proof. Let G be a characteristically free generating system of the quasi-automaton $\mathbf{A} = (A, F, \delta)$ and $b \in G$. There exists an $e \in F$ such that $\delta(b, e) = b$. Thus

$$\underset{f \in F}{\forall} f[\delta(b, f) = \delta(\delta(b, e), f) = \delta(b, ef)],$$

that is,

$$\underset{f \in F}{\forall} f[\bar{f} = \bar{e}\bar{f}].$$

Theorem 4. *Let a_0 be a characteristically free generating element of the cyclic quasi-automaton $\mathbf{A} = (A, F, \delta)$. $\delta(a_0, h)$ $(h \in F)$ is a characteristically free generating element of $\mathbf{A}$ if and only if there exists a $k \in F$ such that $\delta(a_0, hk) = a_0$ and $\bar{k}\bar{h}$ is a left identity element of $\bar{F}$.*

Proof. Let a_0 be a characteristically free generating element of $\mathbf{A}$, $\delta(a_0, hk) = a_0$ $(h, k \in F)$ and $\bar{k}\bar{h}$ a left identity element of $\bar{F}$. Furthermore, for $f, g\, (\in F)$, let,

$$\delta(a_0, hf) = \delta(\delta(a_0, h), f) = \delta(\delta(a_0, h), g) = \delta(a_0, hg).$$

Since a_0 is a characteristically free generating element, thus,

$$\bar{h}\bar{f} = \bar{h}\bar{g},$$

that is,

$$\bar{f} = \bar{k}\bar{h}\bar{f} = \bar{k}\bar{h}\bar{g} = \bar{g}.$$

This means that $\delta(a_0, h)$ is a characteristically free generating element of $\mathbf{A}$. Conversely, let $\delta(a_0, h)$ $(h \in F)$ be a characteristically free generating element of $\mathbf{A}$. There exists a $k \in F$ such that $a_0 = \delta(a_0, hk)$. Now let $f \in F$ be arbitrary. By Lemma 2,

$\bar{h}\bar{k}$ is a left identity element of $\bar{F}$. Therefore

$$\delta(\delta(a_0, h), f) = \delta(a_0, hf) = \delta(a_0, hkhf) = \delta(\delta(a_0, h), khf),$$

that is,

$$\bar{f} = \bar{k}\bar{h}\bar{f}.$$

It is clear that every well-generated state-independent quasi-automaton is characteristically free. The converse of this statement does not hold (see Example 2). However, by Corollary 2, every A-finite strongly connected characteristically free quasi-automaton is state-independent.

Lemma 3. *The characteristic semigroup of a state-independent quasi-automaton is left cancellative.*

Proof. Let the quasi-automaton $\mathbf{A}=(A, F, \delta)$ be state-independent and $\bar{h}\bar{f}=\bar{h}\bar{g}$ $(h, f, g \in F)$. Then for an arbitrary state $a(\in A)$.

$$\delta(a, hf) = \delta(\delta(a, h), f) = \delta(\delta(a, h), g) = \delta(a, hg).$$

Since $\mathbf{A}$ is state-independent thus $\bar{f}=\bar{g}$, i.e., the characteristic semigroup $\bar{F}$ of $\mathbf{A}$ is left cancellative.

The converse of Lemma 3 does not hold. Indeed, in Example 3 the characteristic semigroup $\overline{F(X)}$ of the quasi-automaton $\mathbf{A}=(A, F(X), \delta)$ is left cancellative, but $\mathbf{A}$ is obviously not state-independent.

Example 3. $A=\langle 1, 2, 3\rangle$, $X=\langle x, y\rangle$

δ	1	2	3
x	2	1	2
y	2	3	2

$\bar{F}$	$\bar{x}$	$\bar{x}^2$	$\bar{y}$	$\bar{y}^2$
$\bar{x}$	$\bar{x}^2$	$\bar{x}$	$\bar{y}^2$	$\bar{y}$
$\bar{x}^2$	$\bar{x}$	$\bar{x}^2$	$\bar{y}$	$\bar{y}^2$
$\bar{y}$	$\bar{x}^2$	$\bar{x}$	$\bar{y}^2$	$\bar{y}$
$\bar{y}^2$	$\bar{x}$	$\bar{x}^2$	$\bar{y}$	$\bar{y}^2$

$\mathbf{A}$ is not a characteristically free quasi-automaton.

Theorem 5. *A characteristically free quasi-automaton is state-independent if and only if its characteristic semigroup is left cancellative.*

Proof. The necessity obviously follows from Lemma 3. For the proof of sufficiency, let the characteristic semigroup $\bar{F}$ of the characteristically free quasi-automaton $\mathbf{A}=(A, F, \delta)$ be left cancellative. Take the elements $a(\in A)$ and $f, g(\in F)$ such that $\delta(a, f)=\delta(a, g)$. Let G be a characteristically free generating system of $\mathbf{A}$. There are $b(\in G)$ and $h(\in F)$ such that $\delta(b, h)=a$, thus,

$$\delta(b, hf) = \delta(b, hg).$$

Since $\mathbf{A}$ is characteristically free thus $\bar{h}\bar{f}=\bar{h}\bar{g}$. But $\bar{F}$ is left cancellative Therefore, $\bar{f}=\bar{g}$. This means that $\mathbf{A}$ is state-independent.

We note that if a characteristically free quasi-automaton is state-independent, then each of its minimal generating systems is characteristically free.

In the following two paragraphs we generalise some results of papers [2] and [4], concerning cyclic state-independent and reversible state-independent quasi-automata for characteristically free quasi-automata.

2. Endomorphism semigroup

Theorem 6. *Let a_0 be a characteristically free generating element of the characteristically free cyclic quasi-automaton $\mathbf{A}=(A, F, \delta)$ and $\delta(a_0, e)=a_0$ $(e\in F)$. Then*

$$E(A) \cong \bar{F}\bar{e}.$$

Proof. Define the following mappings $\alpha_{a_0,h}\colon A\to A$

$$\alpha_{a_0,h}(\delta(a_0,f)) = \delta(a_0, hf) \quad (f\in F). \tag{4}$$

If $\delta(a_0,f)=\delta(a_0,g)$ $(f, g\in F)$ then, by (2), $\bar{f}=\bar{g}$, thus,

$$\delta(a_0, hf) = \delta(\delta(a_0,h),f) = \delta(\delta(a_0,h), g) = \delta(a_0, hg),$$

i.e., $\alpha_{a_0,h}$ is well-defined. Let $a(\in A)$ and $f(\in F)$ be arbitrary elements. Then there exists a $g(\in F)$ such that $\delta(a_0, g)=a$ Therefore,

$$\alpha_{a_0,h}(\delta(a,f)) = \alpha_{a_0,h}(\delta(a_0, gf)) = \delta(a_0, hgf) =$$
$$= \delta(\delta(a_0, hg),f) = \delta(\alpha_{a_0,h}(\delta(a_0, g)), f) = \delta(\alpha_{a_0,h}(a),f),$$

i.e., $\alpha_{a_0,h}$ is an endomorphism of $\mathbf{A}$. Let α be arbitrary endomorphism of $\mathbf{A}$. There exists an $h\in F$ such that $\delta(a_0, h)=\alpha(a_0)$. Then for every $a=\delta(a_0, g)\in A$,

$$\alpha(a) = \alpha(\delta(a_0, g)) = \delta(\alpha(a_0), g) = \delta(\delta(a_0, h), g) = \delta(a_0, hg) =$$
$$= \alpha_{a_0,h}(\delta(a_0, g)) = \alpha_{a_0,h}(a),$$

that is, $\alpha=\alpha_{a_0,h}$. Therefore, every endomorphism of $\mathbf{A}$ is of type (4).

From Lemma 2 it follows that $\bar{e}$ is a left identity element of $\bar{F}$. It can easily be seen that the mapping

$$\alpha_{a_0,h} \to \bar{h}\bar{e} \quad (h\in F)$$

is an isomorphism of $E(A)$ onto $\bar{F}\bar{e}$.

Corollary 3. *The endomorphism semigroup of a characteristically free cyclic quasi-automaton is a homomorphic image of its characteristic semigroup.*

Proof. The mapping $\bar{f}\to\bar{f}\bar{e}$ $(f\in F)$ is an endomorphism of $\bar{F}$.

In Example 2 $\overline{xy}$ is a left identity element of $\overline{F(X)}$.

$$\overline{F(X)} = \langle \overline{x^k}; \overline{y^k}; \overline{x^k y}; \overline{y^l x^k}; \overline{y^l x^{j+1} y} \mid j, k, l = 1, 2, 3, \ldots\rangle,$$

$$\overline{F(X)}\,\overline{xy} = \langle \overline{y^k}; \overline{x^k y}; \overline{y^l x^{j+1} y} \mid j, k, l = 1, 2, 3, \ldots\rangle.$$

Let G be a characteristically free generating system of the characteristically free quasi-automaton $\mathbf{A}=(A, F, \delta)$. Furthermore, $\pi\colon G\to G$ and $\omega\colon G\to F$.

Theorem 7. *The mapping $\varphi_{\pi\omega}\colon A\to A$ for which*

$$\varphi_{\pi\omega}\big(\delta(b,f)\big)=\delta\big(\pi(b),\omega(b)f\big)\quad (b\in G; f\in F) \tag{5}$$

is an endomorphism of **A**. *Furthermore, every endomorphism of* **A** *is of type* (5) *and*

$$\varphi_{\pi\omega}=\bigcup_{b\in G}\varphi_{b,\pi(b)}\alpha_{b,\omega(b)},$$

where $\varphi_{b,\pi(b)}$ is a mapping of type (3) *and $\alpha_{b,\omega(b)}$ is a mapping of type* (4).

Proof. Let $\delta(b,f)=\delta(c,g)$ $(b, c\in G; f, g\in F)$. From (2) it follows that $b=c$ and $\bar f=\bar g$, that is, $\pi(b)=\pi(c)$ and $\overline{\omega(b)}\bar f=\overline{\omega(b)}\bar g$. Therefore, $\varphi_{\pi\omega}$ is well-defined. Let $a=\delta(b,h)$ be an arbitrary state of **A** and $f\in F$. Then

$$\varphi_{\pi\omega}\big(\delta(a,f)\big)=\varphi_{\pi\omega}\big(\delta(b,hf)\big)=\delta\big(\pi(b),\omega(b)hf\big)=$$
$$=\delta\big(\delta(\pi(b),\omega(b)h),f\big)=\delta\big(\varphi_{\pi\omega}(\delta(b,h)),f\big)=\delta\big(\varphi_{\pi\omega}(a),f\big).$$

Therefore, $\varphi_{\pi\omega}$ is an endomorphism of **A**. Let α be an arbitrary endomorphism of **A**, $\alpha(b)\in A_c$ $(b, c\in G)$ and $\alpha(b)=\delta(c,h)$ $(h\in F)$. Since the subsets A_c $(c\in G)$ of A form a partition on A, thus the mapping $\pi\colon b\to c$ is well-defined. Let $\omega\colon G\to F$ such that $\delta\big(c,\omega(b)\big)=\alpha(b)$. Then

$$\alpha\big(\delta(b,f)\big)=\delta\big(\alpha(b),f\big)=\delta\big(\delta(c,\omega(b)),f\big)=$$
$$=\delta\big(c,\omega(b)f\big)=\delta\big(\pi(b),\omega(b)f\big)=\varphi_{\pi\omega}\big(\delta(b,f)\big)\quad (b\in G, f\in F),$$

that is, $\alpha=\varphi_{\pi\omega}$. This means that α is a mapping of type (5). Furthermore,

$$\varphi_{\pi\omega}\big(\delta(b,f)\big)=\delta\big(\pi(b),\omega(b)f\big)=\varphi_{b,\pi(b)}\big(\delta(b,\omega(b)f)\big)=\varphi_{b,\pi(b)}\alpha_{b,\omega(b)}\big(\delta(b,f)\big),$$

that is,

$$\varphi_{\pi\omega|A_b}=\varphi_{b,\pi(b)}\alpha_{b,\omega(b)}.$$

Denote the set of mappings $\varphi_\pi:=\bigcup_{b\in G}\varphi_{b,\pi(b)}$ by T and the set of mappings $\alpha_\omega:=\bigcup_{b\in G}\alpha_{b,\omega(b)}$ by H. T and H are subsemigroups of $E(A)$ under the usual multiplication of mappings.

Corollary 4. *If the quasi-automaton $\mathbf{A}=(A, F, \delta)$ is characteristically free then*

$$E(A)=TH \quad\textit{and}\quad T\cap H=\{\iota\}.$$

Proof. It is evident that $\varphi_{\pi\omega}=\varphi_\pi\alpha_\omega$ and

$$\varphi_\pi=\alpha_\omega \Leftrightarrow \varphi_\pi=\alpha_\omega=\iota,$$

where ι is the identity element of $E(A)$.

Corollary 5. *If the A-finite quasi-automaton $\mathbf{A}=(A, F, \delta)$ is characteristically free and $\bar F$ is a monoid then*

$$O\big(E(A)\big)=|A|^{|G|}$$

where G is a characteristically free generating system of **A**.

Proof. By Theorem 1, $O(T)$ is equal to the number of the transformations of G, that is, $O(T)=|G|^{|G|}$. Since $\bar{F}$ is a monoid thus, by Theorem 6, $E(A_b)\cong\bar{F}(b\in G)$. By Theorem 2, $O(\bar{F})=\frac{|A|}{|G|}$. Therefore, by Theorem 7, $O(H)=\left(\frac{|A|}{|G|}\right)^{|G|}$. Thus, by Corollary 4,

$$O\big(E(A)\big) = O(T)\cdot O(H) = |G|^{|G|}\cdot\left(\frac{|A|}{|G|}\right)^{|G|} = |A|^{|G|}.$$

Theorem 8. *Let the quasi-automaton* $\mathbf{A}=(A, F, \delta)$ *be characteristically free. Then*

1) $\varphi_\pi\in G(A)$ *if and only if* π *is a permutation of* G, *where* G *is a characteristically free generating system of* $\mathbf{A}$.

2) $\alpha_\omega\in G(A)$ *if and only if* $G'=\langle\delta(b, \omega(b))|b\in G\rangle$ *is a characteristically free generating system of* $\mathbf{A}$.

Proof. 1) By Theorem 1, $\varphi_{\pi|A_b}$ $(b\in G)$ is an isomorphism. Thus $\varphi_\pi\in G(A)$ if and only if

$$\varphi_{\pi|A_b} = \varphi_{\pi|A_c}(b, c\in G) \Rightarrow A_b = A_c,$$

that is, $b=c$. This means that π is a permutation of G.

2) By (3), $\alpha_\omega\in G(A)$ if and only if for every $b\in G$,

$$\overline{\omega(b)f} = \overline{\omega(b)g} \quad (f, g\in F) \Rightarrow \bar{f} = \bar{g}$$

and $G'=\langle\delta(b, \omega(b))|b\in G\rangle$ is a generating system of $\mathbf{A}$, i.e., G' is a characteristically free generating system of $\mathbf{A}$.

The quasi-automaton $\mathbf{A}=(A, F, \delta)$ is called *reversible* if for every pair $a(\in A)$, $f(\in F)$ there exists a $g\,(\in F)$ such that $\delta(a, fg)=a$. (s. V. M. Gluskov [9].)

We note that if $\bar{F}$ is left cancellative (i.e., if the characteristically free quasi-automaton $\mathbf{A}$ is state-independent) then every mapping α_ω is one-to-one. If every $\mathbf{A}_b$ $(b\in G)$ is strongly connected (i.e., $\mathbf{A}$ is reversible) then α_ω is onto. If $\mathbf{A}$ is reversible and state-independent then H is a subgroup of $G(A)$ (see [3] and [4]).

If $\varphi_\pi, \alpha_\omega\in G(A)$ then

$$\varphi_{\pi\omega}\big(\delta(b,f)\big) = \delta\big(\pi(b), \omega(b)f\big) = \alpha_{\pi(b),\omega(b)}\big(\delta(\pi(b),f)\big) =$$
$$= \alpha_{\pi(b),\omega(b)}\,\varphi_{b,\pi(b)}\big(\delta(b,f)\big) \quad (f\in F, b\in G),$$

that is,

$$\varphi_\pi\alpha_\omega = \varphi_{\pi\omega} = \bigcup_{b\in G}\alpha_{\pi(b),\omega(b)}\,\varphi_{b,\pi(b)} = \bigcup_{b\in G}\alpha_{b,\omega(\pi^{-1}(b))}\,\varphi_{\pi^{-1}(b),b} = \alpha'_\omega\varphi_\pi$$

where $\alpha'_\omega := \bigcup_{b\in G}\alpha_{b,\omega(\pi^{-1}(b))}$.

We denote the set of mappings $\alpha_\omega\,(\in G(A))$ by H'. H' is a subgroup of H. Let us denote the set of mappings $\varphi_\pi(\in G(A))$ by P. P is a subgroup of T.

Corollary 6. *If the quasi-automaton* $\mathbf{A}=(A, F, \delta)$ *is characteristically free then*

$$G(A) = PH' = H'P \quad \textit{and} \quad P\cap H' = \{\iota\}.$$

Proof. It is evident that PH', $H'P\subseteq G(A)$. Let $\alpha\in G(A)$. Then there exist $\varphi_\pi\in T$ and $\alpha_\omega\in H$ such that $\alpha=\varphi_\pi\alpha_\omega$, by Corollary 4. We show that $\varphi_\pi\in P$ and $\alpha_\omega\in H'$. Using the proof of Theorem 7, we get that the mapping $\pi\colon b\to c$ $(b, c\in G)$, where $\alpha(b)\in A_c$, is a transformation of G. Assume that $\alpha(b_1)$, $\alpha(b_2)\in A_c$ $(b_1, b_2, c\in G)$.

Then there exist $h_1, h_2 \in F$ for which $\alpha(b_1)=\delta(c, h_1)$ and $\alpha(b_2)=\delta(c, h_2)$, that is, $b_1=\delta(\alpha^{-1}(c), h_1)$ and $b_2=\delta(\alpha^{-1}(c), h_2)$. By Theorem 1, $A_{b_1}=A_{b_2}$, that is, $b_1=b_2$. Thus π is one-to-one. Since $\alpha(b)\in A_c$ thus $\alpha(A_b)\subseteqq A_c$. Thus, for every $c(\in G)$ there exists a $b(\in G)$ such that $\alpha(b)\in A_c$, since α is an automorphism. Therefore, π is a permutation of G, that is, $\varphi_\pi \in G(A)$. This means that $\alpha_\omega=\varphi_\pi^{-1}\alpha\in G(A)$. Since $\varphi_\pi\alpha_\omega=\alpha'_\omega\varphi_\pi$, where $\alpha'_\omega\in H$, thus $\alpha'_\omega=\varphi_\pi\alpha_\omega\varphi_\pi^{-1}\in G(A)$.

Corollary 7. *If the quasi-automaton* $\mathbf{A}=(A, F, \delta)$ *is characteristically free, then* P *can be embedded homomorphically into the automorphism group of* H'.

Proof. It is clear that the mapping $\Theta_\pi\colon \alpha_\omega\to\alpha'_\omega$ is an automorphism of H' $(\varphi_\pi\in P, \alpha_\omega, \alpha'_\omega\in H')$. The mapping $\varphi_\pi\to\Theta_\pi$ $(\varphi_\pi\in P)$ is well-defined. Take arbitrary mappings $\varphi_{\pi_1}, \varphi_{\pi_2}$ $(\in P)$ and α_ω $(\in H')$. If

$$\varphi_{\pi_2}\alpha_\omega = \alpha_{\omega_1}\varphi_{\pi_2} \quad\text{and}\quad \varphi_{\pi_1}\alpha_{\omega_1} = \alpha_{\omega_2}\varphi_{\pi_1}\,(\alpha_{\omega_1}, \alpha_{\omega_2}\in H')$$

then

$$\varphi_{\pi_1\pi_2}\alpha_\omega = \varphi_{\pi_1}\varphi_{\pi_2}\alpha_\omega = \varphi_{\pi_1}\alpha_{\omega_1}\varphi_{\pi_2} = \alpha_{\omega_2}\varphi_{\pi_1}\varphi_{\pi_2} = \alpha_{\omega_2}\varphi_{\pi_1\pi_2},$$

thus,

$$\Theta_{\pi_1}\Theta_{\pi_2}(\alpha_\omega) = \Theta_{\pi_1}(\alpha_{\omega_1}) = \alpha_{\omega_2} = \Theta_{\pi_1\pi_2}(\alpha_\omega),$$

that is, $\Theta_{\pi_1}\Theta_{\pi_2} = \Theta_{\pi_1\pi_2}$.

We note that if the quasi-automaton $\mathbf{A}$ is reversible and state-independent then $H'=H$ (see I. BABCSÁNYI [4].)

Example 4.

$\mathbf{A}$	1	2	3	4	5	6
x	3	3	3	6	6	6
y	2	1	3	5	4	6

$\bar{F}$	$\bar{x}$	$\bar{y}$	$\overline{y^2}$
$\bar{x}$	$\bar{x}$	$\bar{x}$	$\bar{x}$
$\bar{y}$	$\bar{x}$	$\overline{y^2}$	$\bar{y}$
$\overline{y^2}$	$\bar{x}$	$\bar{y}$	$\overline{y^2}$

$G=\langle 1; 4\rangle$ is a characteristically free generating system of $\mathbf{A}$.

$$\pi_1 = \begin{pmatrix}1 & 4\\ 1 & 4\end{pmatrix},\quad \pi_2 = \begin{pmatrix}1 & 4\\ 1 & 1\end{pmatrix},\quad \pi_3 = \begin{pmatrix}1 & 4\\ 4 & 4\end{pmatrix},\quad \pi_4 = \begin{pmatrix}1 & 4\\ 4 & 1\end{pmatrix};$$

$$\omega_1 = \begin{pmatrix}1 & 4\\ \bar{x} & \bar{x}\end{pmatrix},\quad \omega_2 = \begin{pmatrix}1 & 4\\ \bar{x} & \bar{y}\end{pmatrix};\quad \omega_3 = \begin{pmatrix}1 & 4\\ \bar{x} & \overline{y^2}\end{pmatrix},\quad \omega_4 = \begin{pmatrix}1 & 4\\ \bar{y} & \bar{x}\end{pmatrix},\quad \omega_5 = \begin{pmatrix}1 & 4\\ \bar{y} & \bar{y}\end{pmatrix},$$

$$\omega_6 = \begin{pmatrix}1 & 4\\ \bar{y} & \overline{y^2}\end{pmatrix},\quad \omega_7 = \begin{pmatrix}1 & 4\\ \overline{y^2} & \bar{x}\end{pmatrix},\quad \omega_8 = \begin{pmatrix}1 & 4\\ \overline{y^2} & \bar{y}\end{pmatrix},\quad \omega_9 = \begin{pmatrix}1 & 4\\ \overline{y^2} & \overline{y^2}\end{pmatrix}.$$

$$T = \langle \iota = \varphi_{\pi_1}, \varphi_{\pi_2}, \varphi_{\pi_3}, \varphi_{\pi_4}\rangle,\quad H = \langle \alpha_{\omega_i} | i = 1, 2, \dots, 9\rangle,$$

$$O(T) = 4,\quad O(H) = 9,\quad O(E(A)) = O(TH) = |A|^{|G|} = 6^2 = 36,$$

$$T\cap H = \{\iota\},\quad P = \langle \iota = \varphi_{\pi_1}, \varphi_{\pi_4}\rangle,\quad H' = \langle \alpha_{\omega_5}, \alpha_{\omega_6}, \alpha_{\omega_8}, \alpha_{\omega_9} = \iota\rangle.$$

$$\varphi_{\pi_4}\alpha_{\omega_5} = \alpha_{\omega_5}\varphi_{\pi_4},\quad \varphi_{\pi_4}\alpha_{\omega_6} = \alpha_{\omega_8}\varphi_{\pi_4} \quad\text{and}\quad \varphi_{\pi_4}\alpha_{\omega_8} = \alpha_{\omega_6}\varphi_{\pi_4},$$

that is, $G(A)=PH'=H'P$, $P\cap H'=\{\iota\}$.

$$|HT| = 24. \quad\text{Therefore,}\quad E(A) = TH \neq HT.$$

3. Reduced quasi-automata

In the paper [2] we introduced on the state set A of the quasi-automaton $\mathbf{A}=(A, F, \delta)$ the following congruence relation ϱ:

$$a \varrho b \Leftrightarrow \forall_{f \in F} f[\delta(a,f) = \delta(b,f)]. \tag{6}$$

The factor quasi-automaton $\bar{\mathbf{A}} := \mathbf{A}/\varrho$ is said to be the *reduced quasi-automaton belonging to* **A**. The quasi-automaton $\mathbf{A}=(A, F, \delta)$ is called *reduced* if for arbitrary $a, b\ (\in A)$:

$$a \varrho b \Rightarrow a = b.$$

We note that if $\bar{e}$ is a left identity element of $\bar{F}$ then

$$a \varrho b (a, b \in A) \Leftrightarrow \delta(a, e) = \delta(b, e).$$

If the characteristic semigroup $\bar{F}$ of a well-generated quasi-automaton **A** is a monoid, then **A** is reduced. The proof is obvious; we only note that **A** is well-generated if and only if

$$\forall_{a \in A} a[\delta(a, e) = a],$$

where $\bar{e}$ is a right identity element of $\bar{F}$ (see I. BABCSÁNYI [4]).

Denote the characteristic semigroup of $\bar{\mathbf{A}}=(\bar{A}, F, \bar{\delta})$ by $\bar{\bar{F}}$. Let $\bar{\bar{f}}$ be the element of $\bar{\bar{F}}$ represented by $f\,(\in F)$. Furthermore $\bar{a}$ is the element of $\bar{A}$ represented by $a(\in A)$.

Lemma 4. *If the quasi-automaton* $\mathbf{A}=(A, F, \delta)$ *is characteristically free then the quasi-automaton* $\bar{\mathbf{A}}=(\bar{A}, F, \bar{\delta})$ *is characteristically free as well.*

Proof. Let G be a characteristically free generating system of **A**. It is clear that the set $\bar{G}=\langle \bar{a}_0 | a_0 \in G \rangle$ is a generating system of $\bar{A}$. Let

$$\bar{\delta}(\bar{a}_0, f) = \bar{\delta}(\bar{b}_0, g) \quad (a_0, b_0 \in G;\ f, g \in F),$$

that is,

$$\forall_{h \in F} h[\delta(a_0, fh) = \delta(b_0, gh)].$$

Since G is characteristically free thus

$$a_0 = b_0 \quad \text{and} \quad \forall_{h \in F} h[\bar{f}\bar{h} = \bar{g}\bar{h}],$$

thus, $\bar{a}_0 = \bar{b}_0$ and $\bar{\bar{f}} = \bar{\bar{g}}$. This means that $\bar{G}$ is characteristically free.

Theorem 9. *If the quasi-automaton* $\mathbf{A}=(A, F, \delta)$ *is characteristically free then* $E(A) \cong E(\bar{A})$.

Proof. Let G be a characteristically free generating system of **A**. It is evident that all mappings $\varphi_{\bar{\pi}\bar{\omega}}$ of type (5) are endomorphisms of $\bar{\mathbf{A}}$ ($\bar{\pi}: \bar{G} \to \bar{G}$; $\bar{\omega}: \bar{G} \to F$).

Take the mapping $\Psi\colon E(A)\to E(\bar{A})$ for which

$$\Psi(\varphi_{\pi\omega})=\varphi_{\bar{\pi}\bar{\omega}}\Leftrightarrow \underset{a_0\in G}{\forall}\, a_0[\bar{\pi}(\bar{a}_0)=\overline{\pi(a_0)} \quad \text{and} \quad \bar{\omega}(\bar{a}_0)=\omega(a_0)].$$

Since the mapping $a_0\to\bar{a}_0$ $(a_0\in G)$ is one-to-one, thus the $\bar{\pi}$ and $\bar{\omega}$ are well-defined.

$$\varphi_{\pi\omega}=\varphi_{\pi'\omega'}\big(\in E(A)\big)\Rightarrow \underset{a_0\in G}{\forall}\, a_0\big[\underset{f\in F}{\forall}\, f\big[\delta\big(\pi(a_0),\omega(a_0)f\big)=\delta\big(\pi'(a_0),\omega'(a_0)f\big)\big]\big]\Rightarrow$$

$$\Rightarrow \underset{a_0\in G}{\forall}\, a_0\big[\overline{\delta\big(\pi(a_0),\omega(a_0)\big)}=\overline{\delta\big(\pi'(a_0),\omega'(a_0)\big)\big]}\Rightarrow$$

$$\Rightarrow \underset{\bar{a}_0\in\bar{G}}{\forall}\, \bar{a}_0\big[\bar{\delta}\big(\bar{\pi}(\bar{a}_0),\bar{\omega}(\bar{a}_0)\big)\big]=\bar{\delta}\big(\bar{\pi}'(\bar{a}_0),\bar{\omega}'(\bar{a}_0)\big)\big]\Rightarrow\varphi_{\bar{\pi}\bar{\omega}}=\varphi_{\bar{\pi}'\bar{\omega}'}.$$

Conversely,

$$\varphi_{\bar{\pi}\bar{\omega}}=\varphi_{\bar{\pi}'\bar{\omega}'}\Rightarrow \underset{\bar{a}_0\in\bar{G}}{\forall}\, \bar{a}_0\big[\underset{f\in F}{\forall}\, f\big[\bar{\delta}\big(\bar{\pi}(\bar{a}_0),\bar{\omega}(\bar{a}_0)f\big)=\bar{\delta}\big(\bar{\pi}'(\bar{a}_0),\bar{\omega}'(\bar{a}_0)f\big)\big]\big]\Rightarrow$$

$$\Rightarrow \underset{\bar{a}_0\in G}{\forall}\, \bar{a}_0\big[\underset{f\in F}{\forall}\, f\big[\bar{\delta}\big(\overline{\pi(a_0)},\bar{\omega}(\bar{a}_0)f\big)=\bar{\delta}\big(\overline{\pi'(a_0)},\bar{\omega}'(\bar{a}_0)f\big)\big]\big].$$

Since $\overline{\pi(a_0)}$, $\overline{\pi'(a_0)}\in\bar{\mathrm{G}}$ and $\bar{\mathrm{G}}$ is a characteristically free generating system of $\bar{\mathbf{A}}$ thus

$$\underset{a_0\in G}{\forall}\, a_0[\overline{\pi(a_0)}=\overline{\pi'(a_0)}],$$

that is,

$$\underset{a_0\in G}{\forall}\, a_0\big[\underset{f\in F}{\forall}\, f\big[\delta\big(\pi(a_0),f\big)=\delta\big(\pi'(a_0),f\big)\big]\big].$$

But $\pi(a_0)$, $\pi'(a_0)\in G$ and G is a characteristically free generating system of $\mathbf{A}$. Thus

$$\underset{a_0\in G}{\forall}\, a_0[\pi(a_0)=\pi'(a_0)],$$

that is, $\pi=\pi'$. From this, using $\bar{\omega}(\bar{a}_0)=\omega(a_0)$ and $\bar{\omega}'(\bar{a}_0)=\omega'(a_0)$, we get that $\varphi_{\pi\omega}=$ $=\varphi_{\pi'\omega'}$. This means that Ψ is one-to-one. It is clear that Ψ is onto.

Let $\varphi_{\pi_1\omega_1}$, $\varphi_{\pi_2\omega_2}\in E(A)$ and $\delta(a_0,f)$ $(a_0\in G, f\in F)$ an arbitrary state of $\mathbf{A}$. If $\pi:=\pi_1\pi_2$ and $\omega(a_0):=\omega_1\big(\pi_2(a_0)\big)\omega_2(a_0)$ then

$$\varphi_{\pi_1\omega_1}\varphi_{\pi_2\omega_2}\big(\delta(a_0,f)\big)=\varphi_{\pi_1\omega_1}\big(\delta\big(\pi_2(a_0),\omega_2(a_0)f\big)\big)=$$

$$=\delta\big(\pi_1\pi_2(a_0),\omega_1(\pi_2(a_0))\,\omega_2(a_0)f\big)=\varphi_{\pi\omega}\big(\delta(a_0,f)\big),$$

that is, $\varphi_{\pi_1\omega_1}\varphi_{\pi_2\omega_2}=\varphi_{\pi\omega}$. But $\bar{\pi}_1\bar{\pi}_2(\bar{a}_0)=\bar{\pi}_1\big(\overline{\pi_2(a_0)}\big)=\overline{\pi_1\pi_2(a_0)}=\bar{\pi}(\bar{a}_0)$ and $\bar{\omega}_1\big(\bar{\pi}_2(\bar{a}_0)\big)\cdot$ $\cdot\bar{\omega}_2(\bar{a}_0)=\bar{\omega}_1\big(\overline{\pi_2(a_0)}\big)\bar{\omega}_2(\bar{a}_0)=\omega_1\big(\pi_2(a_0)\big)\omega_2(a_0)$. Therefore,

$$\varphi_{\bar{\pi}_1\bar{\omega}_1}\varphi_{\bar{\pi}_2\bar{\omega}_2}\big(\bar{\delta}(\bar{a}_0,f)\big)=\varphi_{\bar{\pi}_1\bar{\omega}_1}\big(\bar{\delta}\big(\bar{\pi}_2(\bar{a}_0),\bar{\omega}_2(\bar{a}_0)f\big)\big)=$$

$$=\bar{\delta}\big(\bar{\pi}_1\bar{\pi}_2(\bar{a}_0),\bar{\omega}_1\big(\bar{\pi}_2(\bar{a}_0)\big)\bar{\omega}_2(\bar{a}_0)f\big)=\varphi_{\bar{\pi}\bar{\omega}}\big(\bar{\delta}(\bar{a}_0,f)\big),$$

that is, $\varphi_{\bar{\pi}_1\bar{\omega}_1}\varphi_{\bar{\pi}_2\bar{\omega}_2}=\varphi_{\bar{\pi}\bar{\omega}}$. Thus Ψ is an isomorphism of $E(A)$ onto $E(\bar{A})$.

We note that if $\pi\neq\pi'$ then $\varphi_{\pi\omega}\neq\varphi_{\pi'\omega'}$. Furthermore,

$$\varphi_{\pi\omega}=\varphi_{\pi\omega'}\Leftrightarrow \underset{a_0\in G}{\forall}\, a_0[\overline{\overline{\omega(a_0)}}=\overline{\overline{\omega'(a_0)}}].$$

Corollary 8. *If the quasi-automaton* $\mathbf{A}=(A, F, \delta)$ *is characteristically free, then the characteristic semigroup* $\bar{\bar{\mathbf{F}}}$ *of* $\bar{\mathbf{A}}$ *can be embedded isomorphically into the endomorphism semigroup* $E(A)$ *of* $\mathbf{A}$.

Proof. Let G be a characteristically free generating system of $\mathbf{A}$ and π the identity mapping on G. Denote the mapping $\varphi_{\pi\omega}$ by φ_h if

$$\underset{a_0 \in G}{\forall} a_0[\omega(a_0) = h].$$

It can clearly be seen that the mapping $\bar{\bar{h}} \to \varphi_h$ $(h \in F)$ is one-to-one. Let $h, k, f \in F$ and $a_0 \in G$. Then

$$\varphi_h \varphi_k(\delta(a_0, f)) = \varphi_h(\delta(a_0, kf)) = \delta(a_0, hkf) = \varphi_{hk}(\delta(a_0, f)),$$

that is, $\varphi_h \varphi_k = \varphi_{hk}$. Thus the mapping $\bar{\bar{h}} \to \varphi_h$ $(h \in F)$ is an isomorphism of $\bar{\bar{F}}$ into $E(A)$.

We note that the characteristic semigroup $\bar{F}$ of the characteristically free quasi-automaton $\mathbf{A}=(A, F, \delta)$ can be embedded homomorphically into $E(A)$. If $O(\bar{\bar{F}})=1$ then every element of $\bar{F}$ is its left identity element. In this case $H=\{\iota\}$.

Corollary 9. *If the cyclic quasi-automaton* $\mathbf{A}=(A, F, \delta)$ *is characteristically free then* $E(A) \cong \bar{\bar{F}}$.

Proof. By Theorem 6, $E(A) \cong \bar{F}\bar{e}$. Since $\bar{e}$ is a left identity element of $\bar{F}$, thus the mapping $\bar{f}\bar{e} \to \bar{\bar{f}}$ $(f \in F)$ is an isomorphism of $\bar{F}\bar{e}$ onto $\bar{\bar{F}}$.

Corollary 10. *The characteristically free quasi-automaton* $\mathbf{A}=(A, F, \delta)$ *is reduced if and only if its characteristic semigroup is a monoid.*

Proof. By Lemma 2, there exists a left identity element $\bar{e}$ of $\bar{F}$, that is,

$$\underset{a \in A}{\forall} a\big[\underset{f \in F}{\forall} f[\delta(a, f) = \delta(a, ef) = \delta(\delta(a, e), f)]\big].$$

If $\mathbf{A}$ is reduced then

$$\underset{a \in A}{\forall} a[a = \delta(a, e)],$$

i.e. $\bar{e}$ is the identity element of $\bar{F}$. It is evident that if $\bar{F}$ is a monoid then $\mathbf{A}$ reduced.

The next result follows from Theorem 6 and Corollary 10.

Corollary 11. *The characteristically free cyclic quasi-automaton* $\mathbf{A}$ *is reduced if and only if* $\bar{F} \cong E(A)$.

Lemma 5. *Let the quasi-automaton* $\mathbf{A}=(A, F, \delta)$ *be characteristically free and* L *the set of left identity elements of* $\bar{\mathbf{F}}$. *Then*

$$\underset{a_0 \in G}{\forall} a_0[\bar{a}_0 = \langle \delta(a_0, e) | \bar{e} \in L \rangle],$$

and for arbitrary pair a_0, b_0 $(\in G)$, $|\bar{a}_0| = |\bar{b}_0|$, *where* G *is a characteristically free generating system of* $\mathbf{A}$.

Proof. Let $\bar{a}_0 = \bar{b}$ $(a_0 \in G, b \in A)$. Then there exist $h \in F$ and $b_0 \in G$ for which $\delta(b_0, h) = b$, thus,

$$\underset{f \in F}{\forall} f[\delta(a_0, f) = \delta(b, f) = \delta(b_0, hf)],$$

that is, $a_0 = b_0$ and $\underset{f \in F}{\forall} f[\bar{f} = \bar{h}\bar{f}]$. Therefore, $\bar{h} \in L$. It is evident that if $\bar{e} \in L$ then $\delta(a_0, e) \in \bar{a}_0$. If $\delta(a_0, e_1) = \delta(a_0, e_2)$ $(a_0 \in G;\ \bar{e}_1, \bar{e}_2 \in L)$ then $\bar{e}_1 = \bar{e}_2$, thus the mapping $\delta(a_0; e) \to \bar{e}$ $(\bar{e} \in L)$ is one-to-one; therefore, $|\bar{a}_0| = O(L)$ $(a_0 \in G)$.

We note that for every state $a(\in A)$:

$$\bar{a} \supseteqq \langle \delta(a, e) | \bar{e} \in L \rangle$$

and $\bar{a} \subseteqq A_{a_0}$, where $a_0 \in G$ and $a = \delta(a_0, h)$ $(h \in F)$.

Corollary 12. (I. Babcsányi [4].) *If the quasi-automaton* $\mathbf{A} = (A, F, \delta)$ *is reversible and state-independent then* $\bar{a} = \langle \delta(a, e) | \bar{e} \in L \rangle$ $(a \in A)$ *and for every pair* $a, b\ (\in A)$, $|\bar{a}| = |\bar{b}|$.

Corollary 13. (I. Babcsányi [4].) *If the reversible state-independent quasi-automaton* $\mathbf{A} = (A, F, \delta)$ *is A-finite and there exists an* $a\ (\in A)$ *such that* $|A_a|$ *is a prime number, then the characteristic semigroup* $\bar{F}$ *of* $\mathbf{A}$ *is a group or every element of* $\bar{F}$ *is its left identity element.*

Proof. By Corollary 12, $|\bar{a}|$ is a divisor of $|A_a|$ $(a \in A)$. If $|A_a|$ is a prime number then $|\bar{a}| = 1$ or $|\bar{a}| = |A_a|$. If $|\bar{a}| = 1$ then, also by Corollary 12, $|\bar{b}| = 1$ for every $b(\in A)$. This implies that $\bar{F}$ is a group. If $|\bar{a}| = |A_a|$ then for every state $b(\in A_a)$,

$$\underset{f \in F}{\forall} f[\delta(a, f) = \delta(b, f)].$$

Since for every $h(\in F)$, $\delta(a, h) \in A_a$ thus

$$\underset{f \in F}{\forall} f[\delta(a, f) = \delta(a, hf)],$$

that is,

$$\underset{f \in F}{\forall} f[\bar{f} = \bar{h}\bar{f}].$$

Therefore, $\bar{h}$ is a left identity element of $\bar{F}$.

Let the characteristically free quasi-automaton $\mathbf{A} = (A, F, \delta)$ be cyclic and a_0 a characteristically free generating element of $\mathbf{A}$. $\delta(a_0, h)$ $(h \in F)$ is a characteristically free generating element of $\mathbf{A}$ if and only if the mapping $\alpha_{a_0, h}$ (see (3)) is an automorphism of $\mathbf{A}$. This means that the cardinal number of the set of characteristically free generating elements equals $O(G(A))$.

In Example 2 $\overline{(i, 1)} = \langle (i, 1) \rangle$; $\overline{(i, 2)} = \langle (i, 2);\ (i, 4) \rangle$; $\overline{(i, 2j+1)} = \langle (i, 2j+1);\ (i, 2j+4) \rangle$ $(i, j = 1, 2, 3, \ldots)$. $\bar{\bar{F}} = \langle \overline{\overline{x^k}};\ \overline{\overline{y^k}};\ \overline{\overline{xy}};\ \overline{\overline{y^l x^k}} | k, l = 1, 2, 3, \ldots \rangle$. $E(A) \cong \bar{\bar{F}}$ and $G(A) = \{\iota\}$.

Theorem 10. *If the characteristically free quasi-automaton* $\mathbf{A} = (A, F, \delta)$ *is cyclic then the quasi-automaton* $\mathbf{E}(\mathbf{A}) = (E(A), F, \delta')$ *is well-defined, where*

$$\delta'(\alpha_{a_0, h}, f) = \alpha_{a_0, hf} \quad (f \in F)$$

and $\mathbf{E}(\mathbf{A}) \cong \bar{\mathbf{A}}$.

Proof. Since

$$\alpha_{a_0,h} = \alpha_{a_0,k} \Leftrightarrow \underset{f\in F}{\forall} f[\delta(a_0, hf) = \delta(a_0, kf)],$$

thus

$$\alpha_{a_0,h} = \alpha_{a_0,k} \Rightarrow \underset{f\in F}{\forall} f[\alpha_{a_0,hf} = \alpha_{a_0,kf}].$$

Furthermore,

$$\delta'(\alpha_{a_0,h}, fg) = \alpha_{a_0,hfg} = \delta'(\alpha_{a_0,hf}, g) = \delta'(\delta'(\alpha_{a_0,h}, f), g)$$

($h, k, f, g\in F$; a_0 is a characteristically free generating element of **A**.), that is, **E**(**A**) is well-defined. The mapping $\Psi\colon E(A)\to\bar{A}$ for which

$$\Psi: \alpha_{a_0,h} \to \overline{\delta(a_0, h)} \quad (h\in F)$$

is one-to-one and onto. Finally, we shall show that Ψ is a homomorphism. Take arbitrary elements $\alpha_{a_0,h}\in E(A)$ and $f\in F$. Then

$$\Psi(\delta'(\alpha_{a_0,h}, f)) = \Psi(\alpha_{a_0,hf}) = \overline{\delta(a_0, hf)} =$$
$$= \overline{\delta(\delta(a_0, h), f)} = \bar{\delta}(\overline{\delta(a_0, h)}, f) = \bar{\delta}(\Psi(\alpha_{a_0,h}), f).$$

Theorem 11. *If the characteristically free quasi-automaton* $\mathbf{A}=(A, F, \delta)$ *is cyclic, then* $E(E(A))$ *is the semigroup of left translations of* $E(A)$ *and* $E(E(A))\cong E(A)$.

Proof. Note that $E(E(A))$ denote the endomorphism semigroup of **E**(**A**).
Let $\alpha_{a_0,h}, \alpha_{a_0,k}(\in E(A))$ be arbitrary endomorphisms and $\mu\in E(E(A))$. Then

$$\mu(\alpha_{a_0,h}\alpha_{a_0,k}) = \mu(\alpha_{a_0,hk}) = \mu(\delta'(\alpha_{a_0,h}, k)) = \delta'(\mu(\alpha_{a_0,h}), k) =$$
$$= \delta'(\alpha_{a_0,g}, k) = \alpha_{a_0,gk} = \alpha_{a_0,g}\alpha_{a_0,k} = \mu(\alpha_{a_0,h})\alpha_{a_0,k},$$

where $h, k, g\in F$ and $\mu(\alpha_{a_0,h})=\alpha_{a_0,g}$. This means that μ is a left translation of $E(A)$. Conversely, if μ is a left translation of $E(A)$, then

$$\mu(\delta'(\alpha_{a_0,h}, f)) = \mu(\alpha_{a_0,hf}) = \mu(\alpha_{a_0,h}\alpha_{a_0,f}) = \mu(\alpha_{a_0,h})\alpha_{a_0,f} =$$
$$= \alpha_{a_0,g}\alpha_{a_0,f} = \alpha_{a_0,gf} = \delta'(\alpha_{a_0,g}, f) = \delta'(\mu(\alpha_{a_0,h}), f),$$

where $f\in F$ and $\mu(\alpha_{a_0,h})=\alpha_{a_0,g}$, i.e. μ is an endomorphism of **E**(**A**). It is well-known that every monoid is isomorphic to the semigroup of its left translations.

We note that if the quasi-automaton $\mathbf{A}=(A, F, \delta)$ is cyclic and characteristically free, a_0 is a characteristically free generating element of **A**, $\delta(a_0, e)=a_0 (e\in F)$ and $A_e:=\langle\delta(a_0, fe)|f\in F\rangle$, then the quasi-automaton $\mathbf{A}_e=(A_e, Fe, \delta_e)$ is well-defined. $\mathbf{A}_e$ is a reduced sub-quasi-automaton of **A** and $\overline{Fe^{A_e}}\cong\bar{\bar{F}}$.

Theorem 12. *If the endomorphism semigroup* $E(A)$ *of the characteristically free cyclic quasi-automaton* $\mathbf{A}=(A, F, \delta)$ *is isomorphic to the direct product of semigroups* E_i $(i=1, 2, \dots, n)$ *then* $\bar{\mathbf{A}}$ *is isomorphic to the A-direct product of reduced characteristically free cyclic quasi-automata* $\mathbf{A}_i=(A_i, F, \delta_i)$ *and* $E(A_i)\cong E_i$.

Proof. It is sufficient to prove this theorem for $n=2$. Let $E(A)\cong E_1\otimes E_2$. We can assume that $E(A)=E_1\otimes E_2$. By Theorem 10, $\mathbf{E}(\mathbf{A})\cong\bar{\mathbf{A}}$. Let $\alpha_{a_0,h}:=(\alpha_{1,h}, \alpha_{2,h})$

($\alpha_{i,h} \in E_i$, $i=1, 2$). Since

$$(\alpha_{1,hf}, \alpha_{2,hf}) = \alpha_{a_0,hf} = \alpha_{a_0,h}\alpha_{a_0,f} = (\alpha_{1,h}, \alpha_{2,h})(\alpha_{1,f}, \alpha_{2,f}) = (\alpha_{1,h}\alpha_{1,f}, \alpha_{2,h}\alpha_{2,f})$$

thus $\alpha_{i,hf}=\alpha_{i,h}\alpha_{i,f}$. This means that the mappings δ_i: $E_i \times F \to E_i$ given by

$$\delta_i(\alpha_{i,h}, f) = \alpha_{i,hf}$$

are well-defined. Furthermore, the quasi-automata $\mathbf{E}_i=(E_i, F, \delta_i)$ are also well-defined.

$$\delta'\big((\alpha_{1,h}, \alpha_{2,h}), f\big) = \delta'(\alpha_{a_0,h}, f) = \alpha_{a_0,hf} =$$
$$= (\alpha_{1,hf}, \alpha_{2,hf}) = \big(\delta_1(\alpha_{1,h}, f), \delta_2(\alpha_{2,h}, f)\big),$$

that is, $\mathbf{E}(\mathbf{A})=\mathbf{E}_1 \otimes \mathbf{E}_2$. Thus $\overline{\mathbf{A}} \cong \mathbf{E}_1 \otimes \mathbf{E}_2$. It is evident that $\alpha_{a_0,e}$ is a characteristically free generating element of $\mathbf{E}(\mathbf{A})$, where a_0 is a characteristically free generating element of $\mathbf{A}$ and $\delta(a_0, e)=a_0$ ($e \in F$). Prove that $\alpha_{i,e}$ ($i=1, 2$) is a characteristically free generating element of $\mathbf{E}_i$. Let

$$\alpha_{i,f} = \delta_i(\alpha_{i,e}, f) = \delta_i(\alpha_{i,e}, g) = \alpha_{i,g} \quad (f, g \in F).$$

Then for every $h \in F$,

$$\delta_i(\alpha_{i,h}, f) = \alpha_{i,hf} = \alpha_{i,h}\alpha_{i,f} = \alpha_{i,h}\alpha_{i,g} = \alpha_{i,hg} = \delta_i(\alpha_{i,h}, g),$$

that is $\bar{f}^{E_i}=\bar{g}^{E_i}$. Therefore, the quasi-automata $\mathbf{E}_i$ are cyclic and characteristically free. From Theorem 6 it follows that β_h: $\alpha_{i,f} \to \alpha_{i,hf}$ ($f \in F$) is an endomorphism of $\mathbf{E}_i$, and for arbitrary endomorphism β of $\mathbf{E}_i$ there exists an $h \in F$ such that $\beta=\beta_h$.

$$\beta_h = \beta_k (h, k \in F) \Leftrightarrow \underset{f \in F}{\forall} f[\alpha_{i,hf} = \alpha_{i,kf}] \Leftrightarrow \alpha_{i,he} = \alpha_{i,ke}.$$

But $\alpha_{i,h}=\alpha_{i,he}$ and $\alpha_{i,k}=\alpha_{i,ke}$. Therefore, the mapping $\beta_h \to \alpha_{i,h}$ ($h \in F$) is a one-to-one mapping of $E(E_i)$ onto E_i. Since $\beta_f\beta_g=\beta_{fg}$ ($f, g \in F$), thus the mapping $\beta_h \to \alpha_{i,h}$ ($h \in F$) is an isomorphism. Let $\overline{\alpha_{i,h}}=\overline{\alpha_{i,k}}$, that is,

$$\underset{f \in F}{\forall} f[\delta_i(\alpha_{i,h}, f) = \delta_i(\alpha_{i,k}, f)].$$

Thus $\alpha_{i,h}=\alpha_{i,he}=\alpha_{i,ke}=\alpha_{i,k}$. Therefore, the quasi-automata $\mathbf{E}_i$ are reduced.

Corollary 14. *The reduced characteristically free cyclic quasi-automaton* $\mathbf{A}$ *is isomorphic to the A-direct product of reduced characteristically free cyclic quasi-automata* $\mathbf{A}_i$ ($i=1, 2, \ldots, n$) *if* $E(A) \cong E(A_1) \otimes E(A_2) \otimes \ldots \otimes E(A_n)$.

Example 5.

$\mathbf{A}_1$	1	2
x	1	2
y	2	2

$\mathbf{A}_2$	3	4
x	4	3
y	4	3

$\mathbf{A}_1 \otimes \mathbf{A}_2$	(1,3)	(1,4)	(2,3)	(2,4)
x	(1,4)	(1,3)	(2,4)	(2,3)
y	(2,4)	(2,3)	(2,4)	(2,3).

1 is characteristically free generating element of $\mathbf{A}_1$. 3 and 4 are characteristically free generating element of $\mathbf{A}_2$. $\mathbf{A}_1$ and $\mathbf{A}_2$ are reduced. $E(A_1)=\langle \alpha_1, \beta_1 \rangle$, where $\alpha_1=\begin{pmatrix} 1 & 2 \\ 1 & 2 \end{pmatrix}$ and $\beta_1=\begin{pmatrix} 1 & 2 \\ 2 & 2 \end{pmatrix}$ $E(A_2)=\langle \alpha_2, \beta_2 \rangle$, where $\alpha_2=\begin{pmatrix} 3 & 4 \\ 3 & 4 \end{pmatrix}$ and $\beta_2=\begin{pmatrix} 3 & 4 \\ 4 & 3 \end{pmatrix}$, $E(A_1 \times A_2)=E(A_1) \otimes E(A_2)$.

4. Homomorphism

Let $\mathbf{A}=(A, F, \delta)$ be a quasi-automaton and $I\,(\notin F)$ an arbitrary symbol. Define the semigroup F^I to be $F\cup\{I\}$, multiplication in F is unchanged and I acts as an identity for $F\cup\{I\}$. Furthermore, let φ be a mapping of A into itself and $\delta_\varphi\colon A\times F^I\to A$ such that

$$\delta_\varphi(a,f)=\begin{cases}\delta(a,f) & \text{if } f\in F\\ \varphi(a) & \text{if } f=I\end{cases}\quad (a\in A). \tag{7}$$

Lemma 6. (I. Babcsányi [4].) *The quasi-automaton* $\mathbf{A}_\varphi:=(A, F^I, \delta_\varphi)$ *is well-defined if and only if* φ *is an idempotent endomorphism of the quasi-automaton* $\mathbf{A}=(A, F, \delta)$ *and the restriction of* φ *to the kernel of* $\mathbf{A}$ *is the identity mapping. In this case* $\mathbf{A}$ *is sub-quasi-automaton of* $\mathbf{A}_\varphi$.

Proof. Necessity. Assume that the quasi-automaton $\mathbf{A}_\varphi$ is well-defined. Let $a(\in A)$ be an arbitrary state. Then

$$\varphi(a)=\delta_\varphi(a,I)=\delta_\varphi(a,I^2)=\delta_\varphi(\delta_\varphi(a,I),I)=\varphi^2(a),$$

that is, $\varphi^2=\varphi$. Furthermore, if $f\in F$ then

$$\delta_\varphi(a,If)=\delta_\varphi(a,fI)=\delta_\varphi(a,f)=\delta(a,f),$$

$$\delta_\varphi(\delta_\varphi(a,f),I)=\delta_\varphi(\delta(a,f),I)=\varphi(\delta(a,f)),$$

$$\delta_\varphi(\delta_\varphi(a,I),f)=\delta_\varphi(\varphi(a),f)=\delta(\varphi(a),f).$$

Since $\mathbf{A}_\varphi$ is well-defined, thus

$$\delta(a,f)=\varphi(\delta(a,f))=\delta(\varphi(a),f).$$

This means that φ is an idempotent endomorphism of $\mathbf{A}$ and $\varphi|A_1=\iota$ (A_1 is the state set of the kernel of $\mathbf{A}$ (see (1))). The proof of sufficiency is similar. Since F is a subsemigroup of F^I and δ coincides with the restriction of δ_φ to $A\times F$, thus $\mathbf{A}$ is sub-quasi-automaton of $\mathbf{A}_\varphi$.

Theorem 13. (I. Babcsányi [4].) *Every homomorphism of the quasi-automaton* $\mathbf{A}_\varphi=(A, F^I, \delta_\varphi)$ *is a homomorphism of the quasi-automaton* $\mathbf{A}=(A, F, \delta)$. *Conversely, if* Ψ *is a homomorphism of* $\mathbf{A}$ *onto the quasi-automaton* $\mathbf{B}=(B, F, \delta')$, *then* Ψ *is a homomorphism of* $\mathbf{A}_\varphi$ *onto* $\mathbf{B}_{\varphi'}$ *if and only if* $\Psi\varphi=\varphi'\Psi$.

Proof. Since A is the state set of $\mathbf{A}$ and $\mathbf{A}_\varphi$, furthermore, $\mathbf{A}$ is a sub-quasi-automaton of $\mathbf{A}_\varphi$, thus every homomorphism of $\mathbf{A}_\varphi$ is a homomorphism if $\mathbf{A}$. Conversely, let Ψ be a homomorphism of $\mathbf{A}$ onto $\mathbf{B}$. φ and φ' are mappings of type (7). It is clear that Ψ is a homomorphism of $\mathbf{A}_\varphi$ onto $\mathbf{B}_{\varphi'}$, if and only if

$$\forall_{a\in A}\, a[\Psi\varphi(a)=\Psi(\delta_\varphi(a,I))=\delta'_{\varphi'}(\Psi(a),I)=\varphi'\Psi(a)],$$

that is, $\Psi\varphi=\varphi'\Psi$.

We note that if φ is the identity mapping of A, then the homomorphisms of $\mathbf{A}$ and $\mathbf{A}_\varphi$ coincide. In this case denote $\mathbf{A}_\varphi$ by $\mathbf{A}_I=(A, F^I, \delta_I)$.

Theorem 14. *Let* $\mathbf{A}=(A, F, \delta)$ *be an arbitrary quasi-automaton. There exists a characteristically free quasi-automaton* $\mathbf{B}=(B, F, \delta')$ *such that* $\mathbf{A}_I$ *is the homomorphic image of* $\mathbf{B}$ *and the characteristic semigroups of* $\mathbf{A}_I$ *and* $\mathbf{B}$ *are equal.*

Proof. Take the quasi-automaton $\mathbf{A}_I=(A, F^I, \delta_I)$. Let G be a generating system of A_I. Define the following relation τ on $G\times F^I$:

$$(b,f)\tau(c, g) \Leftrightarrow b = c \quad \text{and} \quad \bar{f}^{A_I} = \bar{g}^{A_I}(b, c\in G;\ f, g\in F^I).$$

It is clear that τ is an equivalence relation. Let C_τ be the partition on $G\times F^I$ induced by τ. $C_\tau(A)$ is the set of the classes $C_\tau(b,f)$ $(b\in G, f\in F^I)$. Consider the mapping $\delta' : C_\tau(A)\times F^I\to C_\tau(A)$ for which

$$\delta'(C_\tau(b,f), h) = C_\tau(b,fh).$$

Let $g, h\in F^I$. Then

$$\delta'(C_\tau(b,f), gh) = C_\tau(b,fgh) = \delta'(C_\tau(b,fg), h) = \delta'(\delta'(C_\tau(b,f), g), h),$$

that is, the quasi-automaton $\mathbf{C}_\tau(A)=(C_\tau(A), F^I, \delta')$ is well-defined. We prove that $\overline{F^I}$ is the characteristic semigroup of $\mathbf{C}_\tau(A)$:

$$\bar{f}^{A_I} = \bar{g}^{A_I} \Leftrightarrow \underset{h\in F^I}{\forall} h[\bar{h}^{A_I}\bar{f}^{A_I} = \bar{h}^{A_I}\bar{g}^{A_I}] \Leftrightarrow$$

$$\Leftrightarrow \underset{h\in F^I}{\forall} h\big[\underset{b\in G}{\forall} b[C_\tau(b, hf) = C_\tau(b, hg)]\big] \Leftrightarrow$$

$$\Leftrightarrow \underset{C_\tau(b,\, h)\in C_\tau(A)}{\forall} C_\tau(b, h)\big[\delta'(C_\tau(b, h),f) = \delta'(C_\tau(b, h), g)\big] \Leftrightarrow \bar{f}^{C_\tau(A)} = \bar{g}^{C_\tau(A)}.$$

The set $G_I:=\langle C_\tau(b, I)\,|\,b\in G\rangle$ is a generating system of $C_\tau(A)$. Let

$$C_\tau(b,f) = \delta'(C_\tau(b, I),f) = \delta'(C_\tau(c, I), g) = C_\tau(c, g)$$

$(b, c\in G; f, g\in F^I)$. Then $b=c$ and $\bar{f}^{A_I}=\bar{g}^{A_I}$. Thus $C_\tau(b, I)=C_\tau(c, I)$ and $\bar{f}^{C_\tau(A)}=\bar{g}^{C_\tau(A)}$, i.e., $\mathbf{C}_\tau(A)$ is characteristically free. The mapping

$$\Psi\colon C_\tau(b,f) \to \delta_I(b,f)(b\in G, f\in F^I)$$

is a homomorphism of $\mathbf{C}_\tau(A)$ onto $\mathbf{A}_I$.

Example 6. Take again the quasi-automaton $\mathbf{A}$ given in the Example 3.

$\mathbf{A}_I$	1	2	3
I	1	2	3
x	2	1	2
y	2	3	2

$$G = \langle 2\rangle$$

$$\overline{F^I} = \langle \bar{x}, \overline{x^2}, \bar{y}, \overline{y^2}, \bar{I}\rangle$$

$\mathbf{C}_\tau(A)$	$C_\tau(2, I)$	$C_\tau(2, x)$	$C_\tau(2, x^2)$	$C_\tau(2, y)$	$C_\tau(2, y^2)$
I	$C_\tau(2, I)$	$C_\tau(2, x)$	$C_\tau(2, x^2)$	$C_\tau(2, y)$	$C_\tau(2, y^2)$
x	$C_\tau(2, x)$	$C_\tau(2, x^2)$	$C_\tau(2, x)$	$C_\tau(2, x^2)$	$C_\tau(2, x)$
y	$C_\tau(2, y)$	$C_\tau(2, y^2)$	$C_\tau(2, y)$	$C_\tau(2, y^2)$	$C_\tau(2, y)$

$$\Psi = \begin{pmatrix} C_\tau(2, I) & C_\tau(2, x) & C_\tau(2, x^2) & C_\tau(2, y) & C_\tau(2, y^2) \\ 2 & 1 & 2 & 3 & 2 \end{pmatrix}$$

Corollary 15. *Let the quasi-automaton* $\mathbf{A}=(A, F, \delta)$ *be well-generated and* $\bar{F}^A$ *a monoid. There exists a characteristically free quasi-automaton* $B=(B, F, \delta')$ *such that* $\mathbf{A}$ *is a homomorphic image of* $\mathbf{B}$ *and* $\bar{F}^B=\bar{F}^A$.

By Theorem 14 the proof is evident. (The identity element of $\bar{F}^A$ acts as I.)

Характеристично свободные квазиавтоматы

А-подквазиавтомат $\mathbf{A}_1=(A_1 F, \delta_1)$ квазиавтомата $\mathbf{A}=(A, F, \delta)$ называется *ягром* автомата $\mathbf{A}$, если $A_1=\langle\delta(a, f)|a\in A, f\in F\rangle$. $\mathbf{A}$ называется *верно-порождённым* если $A=A_1$. Верно-порождённый квазиавтомат $\mathbf{A}$ называется *характеристично свободным* если (2) выполняется. (G есть неприводимая система образующих в квазиавтомате A) $\bar{F}^A$ (или $\bar{F}$) является характеристической подгруппой квазиавтомата $\mathbf{A}$.

Квазиавтомат $\mathbf{A}=(A, F, \delta)$ характеристично свободный тогда и только тогда, когда он прямая сумма изоморфных характеристично свободных циклических квазиавтоматов (Теорема 1.). Если циклический квазиавтомат $\mathbf{A}$ характеристично свободный, тогда $|A|=0(\bar{F})$. (Теорема 3.). Если $\mathbf{A}$ ещё А-конечный, тогда теорема 3. можно повернуть. (Следствие 1.). Характеристично свободный $\mathbf{A}$ от состоянии независимый тогда и только тогда, когда его характеристическая полугруппа авляется с левым сокращением (Теорема 5.).

Во втором цункте получаем все ендоморфизмы характеристично свободных квазиавтоматов (Теорема 6. и 7.)

В третем пункте проводим отношение ϱ (в. ещё [2]) на множестве состояний А квазиавтомата $\mathbf{A}=(A, F, \delta)$. Отношение ϱ конгруенция. $\mathbf{A}$ называется *ограниченным*, если $a\varrho b$ $(a, b\in A\Rightarrow \Rightarrow a=b$. Если $\mathbf{A}$ характеристично свободный, тогда факторквазиавтомат $\bar{\mathbf{A}}:=\mathbf{A}/\varrho$ квазиавтомата $\mathbf{A}$ тоже характеристично свободный (Лемма 4.) и $E(A)\cong E(\bar{A})$ (Теорема 9). (Через $E(A)$ обозначаем полугруппу всех ендоморфизмых $\mathbf{A}$.)

Если $\mathbf{A}$ характеристично свободный циклический квазиавтомат и $E(A)\cong E_1\otimes E_2\otimes \otimes\ldots\otimes E_n$, тогда $\bar{\mathbf{A}}\cong\mathbf{A}_1\otimes\mathbf{A}_2\otimes\ldots\otimes\mathbf{A}_n$, где $\mathbf{A}_i$ $(i=1, 2, \ldots, n)$ характеристично свободные циклические ограниченные квазиавтоматы и $E(A_i)\cong E_i$ (Теорема 12.).

Если $\mathbf{A}=(A, F, \delta)$ верно — порождённый квазиавтомат и $\bar{F}^A$ обладает двусторонной единицей, тогда существует такой характеристично свободный квазиавтомат $\mathbf{B}=(B, F, \delta')$, что $\mathbf{A}$ есть гомоморфный образ квазиавтомата $\mathbf{B}$ и $\bar{F}^A=\bar{F}^B$ (Следствие 15.).

ENTZBRUDER VOCATIONAL SECONDARY SCHOOL
H—9700 SZOMBATHELY, HUNGARY

References

[1] Babcsányi, I., A félperfekt kváziautomatákról (On quasi-perfect quasi-automata), *Mat. Lapok*, v. 21, 1970, pp. 95—102.

[2] Babcsányi, I., Ciklikus állapot-független kváziautomaták (Cyclic state-independent quasi-automata), *Mat. Lapok*, v. 22, 1971, pp. 289—301.

[3] Babcsányi, I., Endomorphisms of group-type quasi-automata, *Acta Cybernet.*, v. 2, 1975, pp. 313—322.

[4] Babcsányi, I., *Generálható kváziautomaták* (Generated quasi-automata), Univ. Doct. Dissertation, Bolyai Institute of József Attila University, Szeged, 1974.

[5] Clifford, A. C. & G. B. Preston, *The algebraic theory of semigroups*, v. 1, 1961, v. 2, 1967.

[6] Fleck, A. C., Preservation of structure by certain classes of functions on automata and related group theoretic properties, Computer Laboratory Michigan State-University, 1961, (preprint).

[7] Gécseg, F. & I. Peák, *Algebraic theory of automata*, Budapest, 1972.

[8] Trauth, CH. A., Group-type automata, *J. Assoc. Comput. Mach.*, v. 13, 1966, pp. 170—175.

[9] Gluskov, V. M. (Глушков, В. М.), Абстрактная теория автоматов, *Uspehi Mat. Nauk*, v. 16:5 (101), 1961, pp. 3—62.

(Received April 16, 1976)

Verallgemeinerte Superposition bei binären Automaten

Von M. Gössel und H. D. Modrow

Das große Interesse, das die linearen Automaten innerhalb der Automatentheorie gefunden haben, beruht zu einem erheblichen Teil auf der Gültigkeit des Superpositionsprinzips bezüglich der Addition. Aufgrund der Gültigkeit dieses Prinzips ist z. B. das Input-Output-Verhalten eines linearen Automaten (für den Initialzustand 0) mit eindimensionalem Input schon durch die sog „Impulsantwort" des Automaten vollständig bestimmt, s. z. B. [1].

Ausgehend vom Superpositionsprinzip für lineare Automaten mit eindimensionalem Input und Output über GF(2) wird in der vorliegenden Arbeit untersucht, inwieweit sich dieses Prinzip von der Operation $+$ (Addition modulo 2, Antivalenz) auch auf andere Operationen (Boolesche Funktionen) und Automatenklassen übertragen läßt.

Ein Ansatz für die Übertragung des Superpositionsprinzips auf andere Boolesche Funktionen findet sich in [2]. Dort wird gezeigt, daß für sog. „disjunktive" Automaten, die in Analogie zu den linearen Automaten eingeführt werden, das Superpositionsprinzip für die Disjunktion gilt.

In der vorliegenden Arbeit wird nicht von gegebenen Automatenklassen ausgegangen, sondern es wird für jede zweistellige Boolesche Funktion die Klasse aller Automaten charakterisiert, die dem Superpositionsprinzip bzgl. dieser Booleschen Funktion genügen. Diese Charakterisierung ist nicht explizit zustandsabhängig. Eine derartige Beschreibung auf der Grundlage der hier gegebenen Charakterisierung findet sich in [3].

In dieser Note werden nur initiale vollständig definierte synchrone Automaten $\mathfrak{A}=[X, Y, Z, z_0, \delta, \lambda]$ mit $X=Y=\{0, 1\}$, $z_0 \in Z$ betrachtet. Diese Automaten werden *binäre Automaten* genannt. Ist $\mathfrak{A}=[X, Y, Z, z_0, \delta, \lambda]$ ein binärer Automat, so bezeichnen wir den letzten von $\mathfrak{A}$ ausgegebenen Buchstaben (aus $Y=\{0, 1\}$) nach Eingabe eines nicht leeren Wortes $p=x_1 \ldots x_t$ $(t \geqq 1)$ durch $g_t^{\mathfrak{A}}(x_1, \ldots, x_t)$,

$$g_t^{\mathfrak{A}}(x_1, \ldots, x_t) = {}_{Df}\, \lambda(\delta(z_0, x_1 \ldots x_{t-1}), x_t).$$

Damit is $\mathfrak{A}$ ein System $\{g_t^{\mathfrak{A}} \mid t=1, 2, \ldots\}$ von Booleschen Funktionen zugeordnet; $g_t^{\mathfrak{A}}\colon \{0, 1\}^t \to \{0, 1\}$.

Definition. Es bezeichne F eine zweistellige Boolesche Funktion, $F: \{0, 1\}^2 \to \{0, 1\}$. Dann nennen wir einen binären Automaten $\mathfrak{A}$ *F-superponierbar*, wenn für jedes $t \geqq 1$ gilt:

$$g_t^{\mathfrak{A}}(F(x_1, x_1'), \ldots, F(x_t, x_t')) = F(g_t^{\mathfrak{A}}(x_1, \ldots, x_t), g_t^{\mathfrak{A}}(x_1', \ldots, x_t')) \tag{1}$$
$$(x_1, \ldots, x_t, x_1', \ldots, x_t' \in \{0, 1\}).$$

Unmittelbar einsichtig ist, daß ein binärer Automat $\mathfrak{A}$ F-superponierbar für jede 2-stellige Boolesche Funktion F ist, wenn er die folgende Eigenschaft besitzt: Es gibt eine eindeutige Abbildung $\tau_{\mathfrak{A}}: Nz^+ \to Nz^+$ [1] mit $\tau_{\mathfrak{A}}(t) \leqq t$, so daß

$$g_t^{\mathfrak{A}}(x_1, \ldots, x_t) = x_{\tau_{\mathfrak{A}}(t)} \tag{2}$$

für jedes $t, t \geqq 1, x_1, \ldots, x_t \in \{0, 1\}$ gilt. Für einen solchen Automaten ist also in jedem Takt t die Funktion $g_t^{\mathfrak{A}}$ die Projektion auf eine (die $\tau_{\mathfrak{A}}(t)$-te) Variable. In jedem Takt t wird von $\mathfrak{A}$ einer der bisher eingegebenen Inputs ausgewählt und als Output direkt ausgegeben.

Automaten, für die (2) gilt, nennen wir *auswählende Automaten;* die Klasse aller dieser Automaten bezeichnen wir durch $\mathscr{A}_{\text{Ausw}}$.

Man zeigt nun umgekehrt auch leicht, daß nur Automaten aus $\mathscr{A}_{\text{Ausw}}$ F-superponierbar für jede 2-stellige Boolesche Funktion sind.

Dazu stellen wir jede Funktion $g_t^{\mathfrak{A}}$ $(t \geqq 1)$ in ihrer antivalenten Normalform dar:

$$g_t^{\mathfrak{A}}(x_1, \ldots, x_t) = a_0 + \sum_{1 \leqq i \leqq t} a_i \cdot x_i + \sum_{1 \leqq i < j \leqq t} a_{ij} \cdot x_i \cdot x_j + \ldots$$
$$\ldots + a_{1 \ldots t} \cdot x_1 \cdot \ldots \cdot x_t; \tag{3}$$

dabei bezeichnet $+$ die Antivalenz (Addition modulo 2), $\cdot$ die Konjunktion (Multiplikation) — im folgenden wird das Zeichen $\cdot$ nicht geschrieben —, und die $a_0, a_1, \ldots, a_t,$ $a_{12}, \ldots, a_{t-1,t}$ $a_{123}, \ldots, a_{1 \ldots t}$ sind feste Koeffizienten aus $\{0, 1\}$. Die Darstellung (3) einer t-stelligen Booleschen Funktion ist eindeutig; vgl. [5].

Verschwänden nun in (3) alle Koeffizienten, so widerspräche dies der vorausgesetzten $\equiv$-Superponierbarkeit (vgl. Tab. 1) von $\mathfrak{A}$; denn dann wäre

$$\begin{aligned} 0 &= g_t^{\mathfrak{A}}(0, \ldots, 0) \\ &= (g_t^{\mathfrak{A}}(0, \ldots, 0) \equiv g_t^{\mathfrak{A}}(1, \ldots, 1)) \\ &= (0 \equiv 0) \\ &= 1. \end{aligned}$$

Wäre in (3) $a_0 = 1$, so widerspräche dies der vorausgesetzten $+$-Superponierbarkeit von $\mathfrak{A}$; denn dann wäre

$$1 = g_t^{\mathfrak{A}}(0, \ldots, 0) = g_t^{\mathfrak{A}}(0, \ldots, 0) + g_t^{\mathfrak{A}}(0, \ldots, 0) = 1 + 1 = 0.$$

Wir nehmen nun an, es wäre $a_1 = a_2 = \ldots = a_t = 0$. Dann muß es einen nicht-verschwindenden Koeffizienten $a_{i_1 \ldots i_k}$ mit $1 < k \leqq t$ geben; o.B.d.A. sei k minimal

[1] $Nz^+ =_{\text{Df}} \{1, 2, 3, \ldots\}$

und $a_{1\ldots k}=1$. Wegen der $+$-Superponierbarkeit von $\mathfrak{A}$ gilt aber

$$g_t^{\mathfrak{A}}(\underbrace{1, \ldots, 1}_{k}, 0, \ldots, 0) = g_t^{\mathfrak{A}}(0, \underbrace{1, \ldots, 1}_{k-1}, 0, \ldots, 0) + g_t^{\mathfrak{A}}(1, 0, \ldots, 0).$$

Nach unserer Annahme wäre aber

$$g_t^{\mathfrak{A}}(\underbrace{1, \ldots, 1}_{k}, 0, \ldots, 0) = a_{1\ldots k} \cdot 1^k = 1, \quad g_t^{\mathfrak{A}}(0, \underbrace{1, \ldots, 1}_{k-1}, 0, \ldots, 0) = 0 = g_t^{\mathfrak{A}}(1, 0, \ldots, 0).$$

Aus diesem Widerspruch folgt, daß es mindestens ein τ $(1 \leqq \tau \leqq t)$ mit $a_\tau = 1$ geben muß.

Gäbe es nun zwei verschiedene nicht-verschwindende Koeffizienten mit je einem Index, etwa $a_1 = a_2 = 1$, so folgte wegen der vorausgesetzten $+$-Superponierbarkeit von $\mathfrak{A}$

$$g_t^{\mathfrak{A}}(1, 1, 0, \ldots, 0) = g_t^{\mathfrak{A}}(1, 0, \ldots, 0) + g_t^{\mathfrak{A}}(0, 1, 0, \ldots, 0) = 1 + 1 = 0$$

und damit wegen der $\vee$-Superponierbarkeit von $\mathfrak{A}$

$$0 = g_t^{\mathfrak{A}}(1, 1, 0, \ldots, 0) = g_t^{\mathfrak{A}}(1, 0, \ldots, 0) \vee g_t^{\mathfrak{A}}(0, 1, 0, \ldots, 0) = 1 \vee 1 = 1.$$

Es gibt also höchstens — und damit genau — einen nicht verschwindenden Koeffizienten a_τ mit $1 \leqq \tau \leqq t$ in (3).

Wir nehmen nun an, es gäbe einen nicht-verschwindenden Koeffizienten $a_{i_1 \ldots i_k}$ mit $k \geqq 2$; o.B.d.A. sei k minimal, $a_{1\ldots k} = 1$.

Fall 1. $\tau \leqq k$. O.B.d.A. sei $\tau = 1$, also $a_1 = 1$, $a_{1\ldots k} = 1$. Dann folgt der Widerspruch

$$\begin{aligned} 0 = a_1 + a_{1\ldots k} &= g_t^{\mathfrak{A}}(\underbrace{1, \ldots, 1}_{k}, 0, \ldots, 0) \\ &= g_t^{\mathfrak{A}}(1, 0, \ldots, 0) + g_t^{\mathfrak{A}}(0, \underbrace{1, \ldots, 1}_{k-1}, 0, \ldots, 0) \\ &= a_1 + 0 = 1. \end{aligned}$$

Fall 2. $\tau > k$. O.B.d.A. sei $\tau = t$. Dann folgt der Widerspruch

$$\begin{aligned} 0 = a_{1\ldots k} + a_t &= g_t^{\mathfrak{A}}(\underbrace{1, \ldots, 1}_{k}, 0, \ldots, 0, 1) \\ &= g_t^{\mathfrak{A}}(1, 0, \ldots, 0) + g_t^{\mathfrak{A}}(0, \underbrace{1, \ldots, 1}_{k-1}, 0, \ldots, 0, 1) \\ &= 0 + a_t = 1. \end{aligned}$$

Damit ist gezeigt:

Satz 1. *Ein binärer Automat $\mathfrak{A}$ ist für jede zweistellige Boolesche Funktion F F-superponierbar genau dann, wenn $\mathfrak{A} \in \mathscr{A}_{\text{Ausw}}$ ist.*

Aus dem Beweis folgt sogar:

Korollar. Ist ein binärer Automat $+$-, $\vee$- und $\equiv$-superponierbar, so ist er für jede zweistellige Boolesche Funktion F F-superponierbar.

(Dies ergibt sich auch unmittelbar aus dem unabhängig bewiesenen Satz 6.)

Tabelle 1

Die zweistelligen Booleschen Funktionen

Die Funktionen sind nach den Koeffizienten ihrer antivalenten Normalform geordnet: $F(x_1, x_2) = A_0 + A_1 x_1 + A_2 x_2 + A_{12} x_1 x_2$.

Im unteren Teil sind die Wertetabellen angegeben

	F_0	F_1	F_2	F_3	F_4	F_5	F_6	F_7	F_8	F_9	F_{10}	F_{11}	F_{12}	F_{13}	F_{14}	F_{15}
A_0	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
A_1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
A_2	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
A_{12}	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
Zeichen	0	1	x_1	$\bar{x}_1$	x_2	$\bar{x}_2$	$+$	$\equiv$	$\wedge$			$\rightarrow$			$\vee$	
x_1 x_2																
0 0	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
1 0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0
0 1	0	1	0	1	1	0	1	0	0	1	0	1	1	0	1	0
1 1	0	1	1	0	1	0	0	1	1	0	0	1	0	1	1	0

Im folgenden soll für jede der 16 zweistelligen Booleschen Funktionen F_i, $i=0, \ldots, 15$ (vgl. Tab. 1), die Klasse derjenigen binären Automaten angegeben werden, die F_i-superponierbar sind.

Zunächst ergibt sich für die nicht echt von zwei Variablen abhängigen Funktionen $F_0, \ldots, F_5$ (s. Tab. 1) der

Satz 2. *Es sei $\mathfrak{A}$ ein binärer Automat. Dann gilt:*

1. *Jeder binäre Automat ist F_2- und F_4-superponierbar.*
2. *$\mathfrak{A}$ ist F_0-superponierbar genau dann, wenn für jedes $t \geqq 1$ $g_t^{\mathfrak{A}}(0, \ldots, 0)=0$ gilt.* *$\mathfrak{A}$ ist F_1-superponierbar genau dann, wenn für jedes $t \geqq 1$ gilt: $g_t^{\mathfrak{A}}(1, \ldots, 1)=1$.*
3. *$\mathfrak{A}$ ist F_3- bzw. F_5-superponierbar genau dann, wenn für jedes $t \geqq 1$ die Funktion $g_t^{\mathfrak{A}}$ selbstdual ist, d.h. wenn für jedes $x_1, \ldots, x_t \in \{0, 1\}$ gilt:*

$$\overline{g_t^{\mathfrak{A}}(x_1, \ldots, x_t)} = g_t^{\mathfrak{A}}(\bar{x}_1, \ldots, \bar{x}_t).$$

Um zu Aussagen für die echt von 2 Variablen abhängigen Funktionen $F_6, \ldots, F_{15}$ zu kommen, ordnen wir jeder Funktion $g_t^{\mathfrak{A}}$, jeder (partiellen) Belegung b der t Variablen von $g_t^{\mathfrak{A}}$, $t \geqq 3$, mit $t-2$ fixierten Werten und jedem $F \in \{F_6, \ldots, F_{15}\}$ zwei zweistellige Boolesche Funktionen G^b und H_F^b, folgendermaßen zu.

Ist $b=[x_1, x_2, x_3^0, \ldots, x_t^0]$ mit $x_3^0, \ldots, x_t^0 \in \{0, 1\}$ (o.B.d.A sind hier gerade den letzten $t-2$ Variablen fixierte Werte zugeordnet), so sei

$$G^b(x_1, x_2) =_{\mathrm{Df}} g_t^{\mathfrak{A}}(x_1, x_2, x_3^0, \ldots, x_t^0), \tag{4}$$

$$H_F^b(x_1, x_2) =_{\mathrm{Df}} g_t^{\mathfrak{A}}(x_1, x_2, F(x_3^0, x_3^0), \ldots, F(x_t^0, x_t^0)). \tag{5}$$

Ist $\mathfrak{A}$ nun F-superponierbar, so muß nach (1) für jede derartige Belegung b und jedes $x_1, x_1', x_2, x_2' \in \{0, 1\}$ gelten:

$$H_F^b(F(x_1, x_1'), F(x_2, x_2')) = F(G^b(x_1, x_2), G^b(x_1', x_2')). \tag{6}$$

Ausgehend von (6) werden wir die Automatenklassen bestimmen, die F_i-superponierbar sind, $F_i \in \{F_6, \ldots, F_{15}\}$. Wir setzen $G^b = G$, $H_F^b = H$ und stellen, F, G, H in antivalenter Normalform dar.

$$\begin{aligned}
F(x_1, x_2) &= A_0 + A_1 x_1 + A_2 x_2 + A_{12} x_1 x_2,\\
G(x_1, x_2) &= B_0 + B_1 x_1 + B_2 x_2 + B_{12} x_1 x_2,\\
H(x_1, x_2) &= b_0 + b_1 x_1 + b_2 x_2 + b_{12} x_1 x_2
\end{aligned} \tag{7}$$

mit $A_0, A_1, A_2, A_{12}, B_0, B_1, B_2, B_{12}, b_0, b_1, b_2, b_{12} \in \{0, 1\}$. Aus (6) und (7) ergibt sich dann (8):

$$\begin{aligned}
&F(G(x_1, x_2), G(x_1', x_2')) && = H(F(x_1, x_1'), F(x_2, x_2'))\\
&= A_0 + A_1 B_0 + A_2 B_0 + A_{12} B_0 && = b_0 + b_1 A_0 + b_2 A_0 + b_{12} A_0\\
&+ x_1 (A_1 B_1 + A_{12} B_0 B_1) && + x_1 (b_1 A_1 + b_{12} A_0 A_1)\\
&+ x_2 (A_1 B_2 + A_{12} B_0 B_2) && + x_2 (b_2 A_1 + b_{12} A_0 A_1)\\
&+ x_1' (A_2 B_1 + A_{12} B_0 B_1) && + x_1' (b_1 A_2 + b_{12} A_0 A_2)\\
&+ x_2' (A_2 B_2 + A_{12} B_0 B_2) && + x_2' (b_2 A_2 + b_{12} A_0 A_2)\\
&+ x_1 x_2 (A_1 B_{12} + A_{12} B_0 B_{12}) && + x_1 x_2 (b_{12} A_1)\\
&+ x_1' x_2' (A_2 B_{12} + A_{12} B_0 B_{12}) && + x_1' x_2' (b_{12} A_2)\\
&+ x_1 x_1' (A_{12} B_1) && + x_1 x_1' (b_1 A_{12} + b_{12} A_0 A_{12})\\
&+ x_2 x_1' (A_{12} B_1 B_2) && + x_2 x_1' (b_{12} A_1 A_2)\\
&+ x_1 x_2' (A_{12} B_1 B_2) && + x_1 x_2' (b_{12} A_1 A_2)\\
&+ x_2 x_2' (A_{12} B_2) && + x_2 x_2' (b_2 A_{12} + b_{12} A_0 A_{12})\\
&+ x_1 x_2 x_1' (A_{12} B_{12} B_1) && + x_1 x_2 x_1' (b_{12} A_{12} A_1)\\
&+ x_1 x_2 x_2' (A_{12} B_{12} B_2) && + x_1 x_2 x_2' (b_{12} A_{12} A_1)\\
&+ x_1 x_1' x_2' (A_{12} B_{12} B_1) && + x_1 x_1' x_2' (b_{12} A_{12} A_2)\\
&+ x_2 x_1' x_2' (A_{12} B_{12} B_2) && + x_2 x_1' x_2' (b_{12} A_{12} A_2)\\
&+ x_1 x_2 x_1' x_2' (A_{12} B_{12}). && + x_1 x_2 x_1' x_2' (A_{12} b_{12}).
\end{aligned} \tag{8}$$

Jede Funktion $F \in \{F_6, \ldots, F_{15}\}$ ist in (7) und (8) durch Vorgabe der Koeffizienten A_0, A_1, A_2, A_{12} festgelegt.

(8) gilt genau dann, wenn die Koeffizienten an gleichen Variablenkombinationen gleich sind; und aus (8) ergibt sich gleichwertig ein System von 16 Gleichungen

für die Koeffizienten $B_0, B_1, B_2, B_{12}, b_0, b_1, b_2, b_{12}$ der antivalenten Normalform von G und H. Aus (8) bestimmen wir zunächst G und H und schließen dann auf $g_t^{\mathfrak{A}}(x_1, x_2, \ldots, x_t)$ für jedes $t \geqq 1$.

Wir wollen das allgemeine Verfahren am Beispiel der Funktionen F_6 und anschließend F_{14} (vgl. Tab. 1) ausführlich darstellen.

Es gilt $F_6(x_1, x_2) = x_1 + x_2$ und somit $A_0 = A_{12} = 0$, $A_1 = A_2 = 1$. Damit erhalten wir aus (8)

$$b_{12} = 0, \quad B_{12} = 0, \quad B_2 = b_2, \quad B_1 = b_1, \quad b_0 = 0.$$

Lösungen sind somit

$$H(x_1, x_2) = b_1 x_1 + b_2 x_2, \tag{9}$$

$$G^b(x_1, x_2) = B_0 + b_1 x_1 + b_2 x_2. \tag{10}$$

Um auf $g_t^{\mathfrak{A}}(x_1, \ldots, x_t)$ zu schließen, stellen wir $g_t^{\mathfrak{A}}$ wieder in antivalenter Normalform (3) dar:

$$g_t^{\mathfrak{A}}(x_1, \ldots, x_t) = a_0 + \sum_{1 \leqq i \leqq t} a_i x_i + \sum_{1 \leqq i < j \leqq t} a_{ij} x_i x_j + \ldots + a_{1 \ldots t} x_1 \ldots x_t.$$

Mit $F_6(x, x) = x + x = 0$ und (5) gilt

$$H(x_1, x_2) = g_t^{\mathfrak{A}}(x_1, x_2, 0, \ldots, 0). \tag{11}$$

Wegen $H(0, 0) = 0$ (nach (9)) ergibt sich aus (11), (9), (3): $a_0 = 0$.

Wir nehmen nun an, es gäbe einen Koeffizienten $a_{i_1 \ldots i_k} = 1$ mit $k \geqq 2$; k sei minimal. O.B.d.A. sei $a_{12 \ldots k} = 1$. Es gilt dann für die spezielle (partielle) Belegung $b = [x_1, x_2, \underbrace{1, \ldots, 1}_{k-l}, 0, \ldots, 0]$ die Beziehung $g_t^{\mathfrak{A}}(b) = c_1 x_1 + c_2 x_2 + c_0 + x_1 x_2$ mit $c_0, c_1, c_2 \in \{0, 1\}$ im Widerspruch zu (10), (4). Damit hat $g_t^{\mathfrak{A}}$ die Form

$$g_t^{\mathfrak{A}}(x_1, \ldots, x_t) = \sum_{1 \leqq i \leqq t} a_i x_i \quad \text{mit} \quad a_1, \ldots, a_t \in \{0, 1\}. \tag{12}$$

Da man unmittelbar überprüft, daß jeder Automat $\mathfrak{A}$, für den $g_t^{\mathfrak{A}}$ die Bedingung (12) erfüllt, $+$-superponierbar ist (d.h. (1) genügt), ist gezeigt:

Satz 3. *Ein binärer Automat $\mathfrak{A}$ ist $+$-superponierbar genau dann, wenn es für jedes $t \geqq 1$ Koeffizienten $a_1^{(t)}, \ldots, a_t^{(t)} \in \{0, 1\}$ so gibt, daß stets gilt:*

$$g_t^{\mathfrak{A}}(x_1, \ldots, x_t) = \sum_{i=1}^{t} a_i^{(t)} x_i.$$

Wir betrachten jetzt die Disjunktion (F_{14} in Tab. 1). Es gilt

$$F_{14}(x_1, x_2) = x_1 \vee x_2 = x_1 + x_2 + x_1 \cdot x_2;$$
$$A_0 = 0, \quad A_1 = A_2 = A_{12} = 1.$$

Aus (8) ergeben sich für $G(x_1, x_2)$ die 5 Lösungen G_1, G_2, G_3, G_4, G_5:

$$G_1(x_1, x_2) = 0, \quad G_2(x_1, x_2) = 1,$$
$$G_3(x_1, x_2) = x_1, \quad G_4(x_1, x_2) = x_2,$$
$$G_5(x_1, x_2) = x_1 + x_2 + x_1 x_2 \tag{13}$$

und $H=G$. Um auf $g_t^{\mathfrak{A}}(x_1, \ldots, x_t)$ zu schließen, gehen wir wieder von der antivalenten Normalform (3) aus.

Fall 1. $a_0=1$. — Wir nehmen an, es gäbe einen von a_0 verschiedenen nichtverschwindenden Koeffizienten $a_{i_1 \ldots i_k}$, $k \geqq 1$; o.B.d.A. sei k minimal, $a_{1 \ldots k}=1$. Für $k=1$ ergibt sich $g_t^{\mathfrak{A}}(x_1, 0, \ldots, 0)=1+x_1$ und für $k \geqq 2$: $g_t^{\mathfrak{A}}(x_1, x_2, 0, \ldots, 0)=1+x_1x_2$ im Widerspruch zu (13). Im Fall $a_0=1$ verschwinden also alle anderen Koeffizienten.

Fall 2. $a_0=0$. — Es sei o.B.d.A. $a_1=\ldots=a_k=1$, $a_{k+1}=\ldots=a_t=0$, $0 \leqq k \leqq t$. Wir zeigen zunächst:

$$a_{i_1 \ldots i_l}=0, \quad \text{falls} \quad i_l > k \quad (l \geqq 2). \tag{14.1}$$

Wir nehmen, (14.1) gilt nicht und betrachten einen Koeffizienten $a_{i_1 \ldots i_l}$ mit $a_{i_1 \ldots i_l}=1$, $i_l > k$, $l \geqq 2$, l minimal. O.B.d.A. sei $a_{m+1 \ldots m+l}=1$, $l \geqq 2$, $m+l>k$, l minimal. Dann gilt. $g_t^{\mathfrak{A}}(0, \ldots, 0, x_{m+1}, 1, \ldots, 1, x_{m+l}, 0, \ldots, 0)=C_1x_{m+1}+x_{m+1}x_{m+l}+C_2$ mit $C_1, C_2 \in \{0, 1\}$ im Widerspruch zu (13). Damit ist (14.1) nachgewiesen; $g_t^{\mathfrak{A}}$ hängt von den Variablen $x_{k+1}, \ldots, x_t$ nicht ab.

Wir zeigen nun:

$$a_{i_1 \ldots i_l}=1, \quad \text{falls} \quad i_l \leqq k \quad (l \geqq 2). \tag{14.2}$$

Für $k<2$ ist nichts zu zeigen. Es sei jetzt $k \geqq 2$. Wir nehmen an, (14.2) gilt nicht und betrachten einen Koeffizienten $a_{i_1 \ldots i_l}$ mit $l \geqq 2$, $i_l \leqq k$, $a_{i_1 \ldots i_l}=0$, l minimal. O.B.d.A. sei $a_{1 \ldots l}=0$, $2 \leqq l \leqq k$, l minimal. Dann ist im Fall $l=2$ $g_t^{\mathfrak{A}}(x_1, x_2, 0, \ldots, 0)= =x_1+x_2$ im Widerspruch zu (13). Im Fall $l>2$ gilt

$$g_t^{\mathfrak{A}}(x_1, x_2, \underbrace{1, \ldots, 1}_{l-2}, 0, \ldots, 0) =$$

$$= \sum_{i=1}^{l-2} \binom{l-2}{i} + (x_1+x_2) \sum_{i=0}^{l-2} \binom{l-2}{i} + x_1x_2 \sum_{i=0}^{l-3} \binom{l-2}{i} = 1+x_1x_2$$

wegen $\sum_{i=0}^{l} \binom{l}{i} = 2^i = 0$ (mod. 2). Mit diesem Widerspruch zu (13) ist (14.2) verifiziert; Koeffizienten, deren sämtliche Indizes zwischen 1 und k liegen, verschwinden nicht.

Gilt also nicht $g_t^{\mathfrak{A}}=1$, so läßt sich $g_t^{\mathfrak{A}}$ allgemein schreiben in der Form

$$g_t^{\mathfrak{A}}(x_1, \ldots, x_t) = \sum_{i \in I} x_i + \sum_{\substack{1 \leqq i < j \leqq t \\ i, j \in I}} x_i x_j + \ldots + \prod_{i \in I} x_i, \tag{15}$$

wobei $I \subseteqq \{1, \ldots, t\}$ ist. Gilt (15), so ist

$$g_t^{\mathfrak{A}}(x_1, \ldots, x_t) = \begin{cases} 1, & \text{falls es ein} \quad i \in I \quad \text{mit} \quad x_i = 1 \quad \text{gibt,} \\ 0 & \text{sonst;} \end{cases}$$

d.h., (15) läßt sich auch schreiben in der Form

$$g_t^{\mathfrak{A}}(x_1, \ldots, x_t) = \bigvee_{i=I} x_i. \tag{16}$$

Man prüft sofort nach, daß die konstanten Funktionen mit dem Wert 1 und die durch (16) festgelegten Funktionen tatsächlich $\vee$-superponierbar sind. Damit haben wir gezeigt:

Satz 4. *Ein binärer Automat $\mathfrak{A}$ ist $\vee$-superponierbar genau dann, wenn für jedes $t \geqq 1$ gilt: $g_t^{\mathfrak{A}}(x_1, \dots, x_t) = 1$ oder es gibt Koeffizienten $a_1^{(t)}, \dots, a_t^{(t)} \in \{0, 1\}$ so, daß gilt:*

$$g_t^{\mathfrak{A}}(x_1, \dots, x_t) = \bigvee_{i=1}^{t} a_i^{(t)} x_i.$$

Mit der Beweismethode zu Satz 3 bzw. Satz 4 zeigt man weiter:

Satz 5. *Ein binärer Automat $\mathfrak{A}$ ist*
a) *F_7-superponierbar ($\equiv$-superponierbar)*
b) *F_8-superponierbar ($\wedge$-superponierbar)*
c) *F_9-superponierbar*
d) *F_{10}-superponierbar*
e) *F_{11}-superponierbar ($\rightarrow$-superponierbar)*
f) *F_{12}-superponierbar*
g) *F_{13}-superponierbar*
h) *F_{15}-superponierbar,*
genau dann, wenn für jedes $t \geqq 1$ gilt:

Für die Koeffizienten $a_0, a_1, \dots, a_t, a_{12}, \dots, a_{1\dots t}$ in der antivalenten Normalform von $g_t^{\mathfrak{A}}$,

$$g_t^{\mathfrak{A}}(x_1, \dots, x_t) = a_0 + \sum_{i=1}^{t} a_i x_i + \sum_{1 \leqq i_1 < i_2 \leqq t} a_{i_1 i_2} x_{i_1} x_{i_2} + \dots + a_{1\dots t} x_1 \dots x_t,$$

gilt:
a) $a_0 + \sum_{i=1}^{t} a_i = 1$ *und alle Koeffizienten mit mehr als einem Index verschwinden*
b) *Höchstens einer der Koeffizienten verschwindet nicht*
c) *Es verschwindet genau ein Koeffizient nicht, dies ist einer der Koeffizienten $a_1, \dots, a_t$ (d.h. $\mathfrak{A} \in \mathscr{A}_{\text{Ausw}}$)*
d) *Es verschwindet höchstens ein Koeffizient nicht, dies ist dann einer der Koeffizienten $a_1, \dots, a_t$*
e) *Es verschwindet genau ein Koeffizient nicht, dies ist einer der Koeffizienten $a_1, \dots, a_t, a_0$*
f) *wie* d)
g) *wie* e)
h) *wie* c).

Nennen wir einen binären Automaten *wesentlich*, wenn es ein $t \geqq 2$ so gibt, daß die Funktion $g_t^{\mathfrak{A}}(x_1, \dots, x_t)$ von mindestens 2 Variablen echt abhängt, so gilt:

Satz 6.
1. *Es gibt keine wesentlichen binären Automaten, die F_9-, F_{10}-, F_{11}-, F_{12}-, F_{13}-oder F_{15}-superponierbar sind.*
2. *Ein wesentlicher binärer Automat $\mathfrak{A}$ ist*
   a) *$+$-superponierbar (F_6)*
   b) *$\equiv$-superponierbar (F_7)*
   c) *$\wedge$-superponierbar (F_8) bzw.*
   d) *$\vee$-superponierbar (F_{14})*

genau dann, wenn es für jedes $t \geqq 1$ eine Menge $I_t \subseteq \{1, \dots, t\}$ so gibt, daß

a) $g_t^{\mathfrak{A}}(x_1, \ldots, x_t) = \sum\limits_{i \in I_t} x_i$

b) $g_t^{\mathfrak{A}}(x_1, \ldots, x_t) = \mathop{\equiv}\limits_{i \in I_t} x_i$

c) $g_t^{\mathfrak{A}}(x_1, \ldots, x_t) = \bigwedge\limits_{i \in I_t} x_i$ *oder* $g_t^{\mathfrak{A}}(x_1, \ldots, x_t) = 0$ *bzw.*

d) $g_t^{\mathfrak{A}}(x_1, \ldots, x_t) = \bigvee\limits_{i \in I_t} x_i$ *oder* $g_t^{\mathfrak{A}}(x_1, \ldots, x_t) = 1$.

gilt (*vgl. Tab.* 1); *dabei ist* $\sum\limits_{i \in \emptyset} x_i = \bigvee\limits_{i \in \emptyset} x_i = 0, \ \mathop{\equiv}\limits_{i \in \emptyset} x_i = \bigwedge\limits_{i \in \emptyset} x_i = 1.$

Die im Satz 6.2 charakterisierten wesentlichen binären Automaten werden in [3, 4] explizit zustandsabhängig charakterisiert.

Kurzfassung

Ausgehend vom Superpositionsprinzip für lineare Automaten mit eindimensionalem Input und Output über GF(2) wird in der vorliegenden Arbeit dieses Prinzip von der Operation + (Addition modulo 2, Antivalenz) auf andere Operationen (Boolesche Funktionen) übertragen. Für jede zweistellige Boolesche Funktion wird die Klasse aller Automaten charakterisiert, die dem Superpositionsprinzip bzgl. dieser Booleschen Funktion genügen.

Generalized Superposition by Binary Automata

In this paper the superposition principle for one-dimensional linear automata over GF(2) is extended from the operation + (addition mod. 2) to other operations (Boolean functions). For every Boolean function of two variables is characterized the class of all automata, for which hold the superposition principle with respect to this Boolean function.

Обобщенный принцип суперпозиции у бинарных автоматов

Исходя с принципа суперпозиции для линейных автоматов с одномерным входом и выходом над полем $GF(2)$ (сложение по модулю 2, антивалентность) обобщается этот принцип на другие операции (булевы функции). Для каждой двухместных булевых функций характеризуется класс всех автоматов, подчиняющихся этому принципу относительно данной булевой функций.

ZENTRALINSTITUT FÜR KYBERNETIK
UND INFORMATIONSPROZESSE DER ADW
DDR-1199 BERLIN-ADLERSHOF
RUDOWER CHAUSSEE 5

Literatur

[1] Gill. A., *Linear sequential circuits*, McGraw Hill, New York, 1966.

[2] Reischer, C., D. A. Simovici, Linear Boolean automata, *Wiss. Z. Techn. Hochsch. Ilmenau*, v. 17, 1971, pp. 83—96.

[3] Gössel, M., Binäre +- und ≡- superponierbare Automaten, *Z. elektr. Inform.- u. Energietechnik*, Leipzig, v. 6, 1976, pp. 225—236.

[4] Gössel, M., Binäre $\vee$- und $\wedge$-superponierbare Automaten, *Z. elektr. Inform.- u. Energietechnik*, Leipzig, v. 6, 1976, pp. 325—330.

[5] Жегалкин, И. И., Арифметизация символической логики. *Мат. сб.* v. 35, 1928, 3/4.

(Eingegangen am 7. August 1974)

Multicontrol Turing machines

By Gy. Révész

Multitape and multihead Turing machines are well-known generalizations of the basic model. All these generalizations have the common feature that they have a single finite state control and thus, they work in a synchronous manner. That is, the finite state control device coordinates the moves of the read-write heads so that they work at the same speed [1].

The new model introduced in the present paper can be considered as an abstract model of multi-processor systems, where the working speeds of the individual processors are independent from each-other. The only restriction is the exclusion of a symultaneous acces to the same storage location, i.e., each storage location can be accessed by only one processor at a time. This limitation is quite reasonable whenever more than one processors share the storage device. The model is asynchronous as there is no other connection between the individual processors except the common storage device through which they can pass information.

Suppose we have two Turing machines sharing a single tape (see Figure 1).

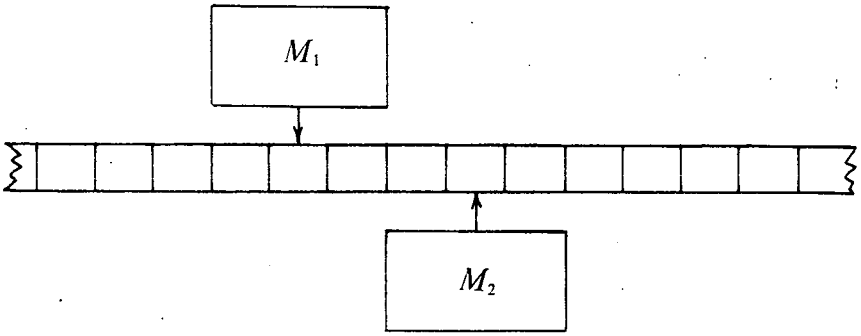


Figure 1

Each finite state control works at its own speed that may also vary in the course of the computation. If they attempt to access the same square on the tape one of them will be delayed until the other completes one step. If this step does not change the position of the corresponding read-write head then another choice is made for the finite control to be the next. The choice can be made in many different ways depending on some preassigned priorities or on the basis of some probabilities, etc. Here we are not concerned with the details of how to make this choice, but we are interested in the computational power of the general model. We will show that the computational power of multicontrol Turing machines is the same as that of the basic model. For this purpose we shall give first our formal definitions.

Definition. A single-control mondeterministic Turing machine is a quintuple $T=(K, Z, q_0, B, M)$, where K is a finite nonvoid set of *internal states*, Z is a finite nonvoid set of *tape symbols*, $q_0 \in K$ is the *initial state*, $B \in Z$ is the *blank symbol*, M is a mapping from $K \times Z$ into the subsets of $K \times (Z-\{B\}) \times \{-1, 0, +1\}$, called *moves*.

Definition. A configuration of a single-control nondeterministic Turing machine is a word XqY, where X and Y are words over $Z-\{B\}$ and $q \in K$.

The configuration denotes the nonblank portion of the tape, the actual position of the read-write head, and the actual internal state of the finite control.

The symbol scanned by the read-write head in this configuration is the first symbol of Y, or B if Y is the empty word.

A move of the Turing machine will change its configuration in the following steps.

a) The internal state of the finite state control is changed.

b) A symbol of $Z-\{B\}$ is printed on the tape in place of the scanned symbol.

c) The read-write head moves one square to the left, remains unchanged, or moves one square to the right as expressed by the values -1, 0, or $+1$, respectively.

The transformation of the configuration induced by the mapping M defines the relation $\Rightarrow$ such that $X_1 q_1 Y_1 \Rightarrow X_2 q_2 Y_2$ iff there is a move in M that changes the configuration $X_1 q_1 Y_1$ into $X_2 q_2 Y_2$. In fact, the mapping M can be given as a set of rewriting rules. Namely.

(i) $qy \to pz \in M$ iff $(p, z, 0) \in M(q, y)$,

(ii) $xqy \to pxz \in M$ for all $x \in Z$ iff $(p, z, -1) \in M(q, y)$,

(iii) $qy \to zp \in M$ iff $(p, z, +1) \in N(q, y)$.

The reflexive and transitive closure of the relation $\Rightarrow$ will be denoted as usual by $\overset{*}{\Rightarrow}$.

Definition. A configuration XqY is final if for the first symbol y of Y the set $M(q, y)=\emptyset$, or if Y is the empty word and $M(q, B)=\emptyset$.

Definition. A computation of a Turing machine is a sequence of moves $q_0 P \overset{*}{\Rightarrow} XqY$, where q_0 is the initial state, P is a word over Z, and XqY is final. The input value of the computation is P while the output is represented by XY, the final contents of the storage tape.

Definition. A two-control mondeterministic Turing machine is an 8-tuple $T=(K', K'', Z, q_0', q_0'', B, M', M'')$, where $T'=(K', Z, q_0', B, M')$ and $T''=(K'', Z, q_0'', B, M'')$ are single-control Turing machines such that $K' \cap K''=\emptyset$.

Definition. A configuration of a two-control Turing machine is a word $Xq'Uq''Y$, where X, U, Y are words over $Z-\{B\}$, $q'\in K'$ and $q''\in K''$.

The relations $\Rightarrow$ and $\overset{*}{\Rightarrow}$ can be defined in a similar fashion as in the case of single-control Turing machines, taking into account that a symbol of K' will be adjacent to a symbol of K'' whenever the two read-write heads are scanning the same square. In such cases either of the two controls (but only one of them) may perform the next move as if the other were not there. Otherwise they work parallel and do not disturb each-other. Now we will show the following.

Theorem. Every computation performed by a two-control Turing machine can be performed by a single-control one.

Proof. In order to prove this theorem we will simulate the computations of a two-control Turing machine with the aid of a single-control one.

The essential feature of the simulation is that the parallel moves of the two control devices will be performed in a serial manner such that only one of them will be activated at a time while the other will be frozen. But the active and the frozen status can be exchanged between them any time that makes the simulation of every parallel computation possible.

Let $T_2=(K', K'', Z, q_0', q_0'', B, M', M'')$ be a two-control Turing machine. Then let $T_1=(K, Z_1, q_0, B, M)$ be defined such that

$$K=((K'\cup\{1,2\})\times(K''\cup\{1,2\}))\cup\{L, R, \text{⅃}, \text{Я}, F, \text{ꟻ}, H\},$$

$$Z_1=Z\cup((K'\cup K'')\times Z).$$

Internal states of the form $[q', q'']$ represent the coincidence of the two read-write heads. A pair of the form $[q', 1]$ or $[q'', 1]$, ($[q', 2]$ or $[q'', 2]$) means that the corresponding control device is acive and it is currently to the left (to the right) of the other. The meanings of the special state symbols are the following:

L (R): activating is passing over to the left (right),
⅃ (Я): activating is passing over to the left (right) leaving a final configuration frozen behind,
F (ꟻ): completely final configuration obtained on the left (right),
H: simulation halted.

The actual state of the frozen control device will be encoded onto the tape as a tape symbol of the form $[q', y]$ or $[q'', y]$. Now we have to specify the mapping M.

1) $M([q', q''], y)=\emptyset$ iff $M'(q', y)=M''(q'', y)=\emptyset$, and

$M([q', q''], y)=M([q'', q'], y)$ for all $q'\in K', q''\in K''$ and $y\in Z$,

2a) $([p', q''], z, 0)\in M([q', q''], y)$ iff $(p', z, 0)\in M'(q', y)$,

2b) $([q', p''], z, 0)\in M([q', q''], y)$ iff $(p'', z, 0)\in M''(q'', y)$.

From here on a pair of related specifications like 2a and 2b will be given by describing just the first of them.

3*a*) $([p', 1], [q'', z], -1) \in M([q', q''], y)$ iff $(p', z, -1) \in M'(q', y)$.

4*a*) $([p', 2], [q'', z], +1) \in M([q', q''], y)$ iff $(p', z, +1) \in M'(q', y)$.

5*a*) For $i = 1, 2$ and $j = -1, 0, +1$,
$([p', i], z, j) \in M([q', i], y)$ iff $(p', z, j) \in M'(q', y)$.

6*a*) $M([q', 1], [q'', y]) = M([q', 2], [q'', y]) = M([q', q''], y)$
for all $q' \in K', q'' \in K''$ and $y \in Z$.

7*a*) $(R, [q', y], +1) \in M([q', 1], y)$ and $(L, [q', y], -1) \in M([q', 2], y)$
iff $M(q', y) \neq \emptyset$.

8) $(R, y, +1) M(R, y)$ and $(L, y, -1) \in M(L, y)$ for all $y \in Z$.

9*a*) $(\text{Я}, [q', y], +1) \in M([q', 1], y)$ and $(\text{⅃}, [q', y], -1) \in M([q', 2], y)$
iff $M(q', y) = \emptyset$.

10) $(\text{Я}, y, +1) \in M(\text{Я}, y)$ and $(\text{⅃}, y, -1) \in M(\text{⅃}, y)$ for all $y \in Z$.

11*a*) $([q', 2], y, 0) \in M(R, [q', y])$ and $([q', 1], y, 0) \in M(L, [q', y])$
for all $q' \in K'$ and $y \in Z$.

12*a*) $([q', 2], y, 0) \in M(\text{Я}, [q', y])$ and $([q', 1], y, 0) \in M(\text{⅃}, [q', y])$
iff $M'(q', y) \neq \emptyset$.

13*a*) $(\text{ꟻ}, y, -1) \in M(\text{Я}, [q', y])$ and $(F, y, +1) \in M(\text{⅃}, [q', y])$
iff $M'(q', y) = \emptyset$.

14) $(\text{ꟻ}, y, -1) \in M(\text{ꟻ}, y)$ and $(F, y, +1) \in M(F, y)$ for all $y \in Z$.

15) $(H, y, 0) \in M(\text{ꟻ}, [q', y])$ and $(H, y, 0) \in M(F, [q', y])$
for all $q' \in K'$ and $y \in Z$.

Now let us see how T_1 simulates T_2. Suppose we have a computation of T_2 starting with a configuration $q_0' q_0'' P$ and ending with a final configuration $Xq'Uq''Y$. Then T_1 will be started with $[q_0' q_0''] P$. As long as the two read-write heads are scanning the same square the simulation is guaranted by specifications 1, 2*a* and 2*b*. As soon as they are parted one of them becomes active while the other becomes frozen (3*a*—*b*, 4*a*—*b*) and the control device of T_1 follows exactly the actions of the active control of T_2 (5*a*—*b*).

If the frozen control is encountered by the active one, either of them will be enabled to proceed (6*a*—*b*). The simulation of the active control may be suspended any time by T_1 so that T_1 switches over to the other. This is realized by T_1 with the aid of special states $R, L,$ Я and ⅃ which cause a search for the frozen control on the tape. T_1 always remembers the correct direction for the search via the state-suffix 1 or 2. (see 7*a*—*b*, 9*a*—*b*, 8, 10). The search is ended by activating the previously frozen control device (11*a*—*b*, 12*a*—*b*) unless both of them happen to be in a final configuration. In the latter case the frozen state will be cancelled from the tape and a message will be sent back to cancel the other encoded state as well and to stop the simulation (13*a*—*b*, 14, 15). If the final configuration of T_2 is of the form $Xq'q''Y$ then the simulation in T_1 will be finished with the configuration $X[q', q'']Y$. In this case the resulting tape inscription of T_1 will be exactly the same as that of T_2, but in other cases we must first get rid of the encoded final states (13*a*—*b*, 15).

The nondeterministic order of parallel steps in T_2 is properly simulated by the nondeterministic active — frozen status switching in T_1, and this completes the proof.

It can be observed that T_1 makes use of the same amount of tape squares as T_2 does. In particular, if T_2 is linearly bounded then so is T_1. On the other hand the above theorem can be extended to more than two control devices and thus, we have the following:

Corollary. Every asynchronous parallel computation performed by a multicontrol Turing machine can be also performed by a single-control Turing machine using the same amount of tape.

A number of interesting special cases of multicontrol Turing machines can be considered. One of them could be an abstract model of the so called pipeline processing where a streamlike information flow is processed simultaneously at different stages by several control units. The tape alphabet Z can be partitioned for this purpose in such a way that the input alphabet of each control unit forms a subset of the output alphabet of the previous one. This means that each control unit would work at full speed as long as the tape inscription permits.

Finally it should be mentioned that the simulation of asynchronous parallel processes was given above by a nondeterministic model even if the individual control units are deterministic. It is known that nondeterminism can be reduced to deterministic operation in case of Turing machines, but this would very likely require additional tape [2].

References

[1] HARTMANIS, J. & R. E. STEARNS, On the computational complexity of algorithms, *Trans. Amer. Math. Soc.*, v., 117, 1965, pp. 285—306.

[2] SAVITCH, W. J., Relationships between nondeterministic and deterministic tape complexities, *J. Comput. System Sci.*, v. 4, 1970, pp. 177—192.

[3] STOSS, H. J., k-Band-Simulation von k-Kopf-Turing-Maschinen, *Computing*, v. 6, 1970, pp. 309—317.

(Received May 12, 1976)

INDEX—TARTALOM

ISSN 0324—721 X

Felelős szerkesztő és kiadó: Gécseg Ferenc
A kézirat a nyomdába érkezett: 1976. december hó
Megjelenés: 1977. augusztus hó
Példányszám: 1000. Terjedelem: 9,63 (A/5) ív
Készült monószedéssel, íves magasnyomással
az MSZ 5601 és az MSZ 5602—55 szabvány szerint
76-5369—Szegedi Nyomda—F. v.: Dobó József