Volume 9 Number 2

ACTA CYBERNETICA

Szeged, 1989

Information for authors: Acta Cybernetica publishes only original papers in English in the field of computer sciences. Review papers are accepted only exceptionally. Manuscripts should be sent in triplicate to one of the Editors. The manuscript must be typed double-spaced on one side of the paper only. For the form of references, see one of the articles previously published in the journal. A list of special symbols should be supplied by the authors.

On the worst-case performance of the *NkF* bin-packing heuristic*

J. CSIRIK, B. IMREH

Bolyai Institute, Department of Computer Science,
Aradi vértanúk tere 1, H—6720 Szeged, Hungary

Introduction

In bin packing, we are given a list

$$L = (s_1, s_2, \ldots, s_n)$$

of items (elements) with a weight function on items and a sequence of unit-capacity bins $B_1, B_2, \ldots$. In this paper, we assume that the item weights are real numbers in the range (0, 1] and that the list is given by the weights. The problem is to find a packing of the items in the bins such that the sum of the items in each bin is not greater than 1, and the number of bins used is minimized.

This problem is NP-hard [GJ] and therefore heuristic algorithms which give "good" solutions in an acceptable computing time are investigated [J], [JDUGG]. We are interested in the worst-case behaviour of the Next-k Fit (*NkF*) algorithm. For this, an upper and a lower bound were given in Johnson's paper. We shall improve both bounds.

Preliminary definitions and notations

For a list L, let $\mathrm{OPT}(L)$ be the number of bins in optimal packing. For a given heuristic algorithm A, let $A(L)$ be the number of bins used by A to pack L. Let $R_A^N = \max\left\{\frac{A(L)}{\mathrm{OPT}(L)} \mid L \text{ is a list with } \mathrm{OPT}(L)=N\right\}$.

* This paper was supported by a grant from the Hungarian Academy of Sciences (OTKA Nr. 1135.)

The *asymptotic worst-case ratio* of A is then defined as

$$R_A = \limsup_{N\to\infty} R_A^N.$$

Let

$$s(L) = \sum_{i=1}^{n} s_i$$

and let $s(B_i)$ denote the sum of the weights of the items in B_i.

We investigate the NkF algorithm, which is defined as follows: we always use k bins at the same time. If the next element, a_j, is coming, we place it into the first of the k used bins which has enough room for it. If no such bin has enough room, we close the first (oldest) of these k bins, open a new one, and put a_j into this bin (this will now be the k-th or youngest bin).

Johnson has proved for the asymptotic worst-case ratio of NkF that

$$1.7+\frac{3}{10k} \leqq R_{NkF} \leqq 2.$$

In this paper we prove that

$$R_{N2F} = 2$$

and that for $k\geqq 3$

$$1.7+\frac{3}{10(k-1)} \leqq R_{NkF} \leqq 1.75+\frac{7}{4(2k+3)}.$$

However, the exact worst-case ratio is not known for $k\geqq 3$.

Results

First we give an upper bound on R_{NkF} for $k\geqq 3$. Let L be an arbitrary list and let us pack the elements of L by means of NkF. Let $B_1, B_2, \ldots, B_r$ denote the sequence of bins used and let m be a fixed nonnegative integer. For any positive integer $i \leqq r+1-m$ the sequence of the bins $B_i, B_{i+1}, \ldots, B_{i+m-1}$ is called a *parcel consisting of m bins* if the following conditions hold

(a) $s(B_t) > 1/2 \quad (t = i, \ldots, i+m-1)$,

(b) $i+m-1 = r$ or $i+m-1 < r \,\&\, s(B_{i+m}) \leqq 1/2$.

We classify the bins of a parcel consisting of m bins with respect to their contents as follows:

(A) $\Sigma s_t \geqq 2/3 \,\&\, (\exists t)(s_t > 1/2)$,

(B) $\Sigma s_t \geqq 2/3 \,\&\, (\forall t)(s_t \leqq 1/2)$,

(C) $\Sigma s_t < 2/3 \,\&\, (\exists t)(s_t > 1/2)$,

(D) $\Sigma s_t < 2/3 \,\&\, (\forall t)(s_t \leqq 1/2)$, where t runs through the set of indices of the items contained in the considered bin. Obviously, we obtain a partition of the bins $B_i, \ldots, B_{i+m-1}$. We shall use the terminology *X-bin* for a bin which is contained in the class determined by the property X, where $X \in \{A, B, C, D\}$. It may be observed that any D-bin contains at least two items; moreover, it contains an item with $s_t \leqq 1/3$.

For the D-bins, the following statement holds.

Lemma 1. There are at most two D-bins among any $k+1$ successive bins of any parcel consisting of $m \geqq k+1$ bins.

Proof. Let $B_1^*, B_2^*, \ldots, B_{k+1}^*$ denote the considered bins. Let $1 \leqq i < j \leqq k+1$ and let us suppose that B_i^* is the D-bin with smallest index and that B_j^* is the D-bin with second smallest index. If $j=k+1$, then the statement obviously holds. Now let us assume that $j<k+1$. After the packing of L the empty room in B_i^* is greater than 1/3. Accordingly, the empty room in it is greater than 1/3 when the first item is packed in B_j^*. Therefore $1/3 < s_1$ holds for this item. By our assumption, B_j^* is a D-bin; thus, $s_1 \leqq 1/2$ and during the further packing at least one item with weight less than 1/3 will be packed in B_j^*. Let us investigate the circumstance of the packing of the first such item. It should be observed that the bin B_i^* contains enough empty room for this item. Therefore, the packing of this item in B_j^* implies that at this time the bin B_i^* is already closed. This results that, up to the closing of B_i^* the content of B_j^* is not greater than 1/2. But then, the weight of the first packed item in B_{j+u}^* is greater than 1/2 if $u \in \{1, \ldots, k+1-j\}$. This means that $B_{j+1}^*, \ldots, B_{k+1}^*$ are of types A or C, which yields the validity of our statement.

Lemma 2. For any $k+1$ successive bins of any parcel consisting of $m \geqq k+1$ bins if there exists a C-bin among the considered bins and if there exists a D-bin among the bins succeeding the C-bin, then the bins succeeding the D-bin are of types A or C.

Proof. In the proof of Lemma 1 we only made use of the fact that B_i^* has empty room greater than 1/3 and this property holds for any C-bin, too; thus, by repeating the proof of Lemma 1, we obtain the validity of Lemma 2.

Any $k+2$ successive bins of a parcel consisting of $m \geqq k+2$ bins is called a *block*. We classify the blocks as follows:

(1) it contains at most one D-bin,

(2) it contains two D-bins or it contains three D-bins and at least one B-bin,

(3) it contains three D-bins and at least one A-bin; moreover, the remaining $k-2$ bins are of types A or C,

(4) it contains three D-bins and $k-1$ C-bins.

From Lemma 1 it follows that any block contains at most three D-bins, and so the above classification induces a partition of the blocks. We shall use the terminology *j-block* or block of type j if it has the j-th property for some $j \in \{1, \ldots, 4\}$.

Now let us consider an arbitrary block of a parcel consisting of $m \geqq k+2$ bins. Let s denote the sum of the weights of the items contained in the bins of the block and let q' and q be the numbers of its A-bins and C-bins, respectively.

The following statement then holds.

Lemma 3. For any $r \in \{1, \ldots, 4\}$ if a block is of type r, then the r-th assertion holds for it among the following ones:

(1) $s \geqq (k+2)\frac{2}{3}-(q+1)\frac{1}{6}$,

(2) $s \geqq (k+2)\frac{2}{3}-(q+2)\frac{1}{6}$,

(3) $s \geqq (k+2)\frac{2}{3}-(q+3)\frac{1}{6}\ \&\ q+q' = k-1\ \&\ q' > 0$,

(4) $s \geqq (k+2)\frac{2}{3}-(q+3)\frac{1}{6}\ \&\ q = k-1$.

Proof. In the cases $r=1$, $r=3$ and $r=4$ the statement follows from the definitions. If $r=2$ and the block contains only two D-bins, then the assertion is again obvious.

Now let us suppose that the considered block contains three D-bins and at least one B-bin. Let $B_1^*, \ldots, B_{k+2}^*$ denote the bins of the block. Since it contains three D-bins, by using Lemma 1 twice, we obtain that B_1^* and B_{k+2}^* are D-bins. Let us assume that B_j^* is the intermediate D-bin for some $2 \leqq j \leqq k+1$. By our assumption, the block contains a B-bin. Let B_l^* denote this bin, where $2 \leqq l \leqq k+1$ and $l \neq j$. We distinguish the following two cases.

Case 1. Let us suppose that $l<j$. Then $l \leqq k$, and so, at the time of opening of B_l^*, the bin B_1^* is open. On the other hand, B_1^* is a D-bin, and so, after the packing of all elements of L, the bin B_1^* contains empty room with weight $\frac{1}{3}+\Delta$, where $\Delta > 0$. But then B_1^* contains empty room with weight at least $\frac{1}{3}+\Delta$ when the first item is packed in the bin B_l^*. Therefore, $\frac{1}{3}+\Delta < s_1$ holds for this item. Moreover, since B_l^* is a B-bin, $s_1 \leqq 1/2$. We now distinguish two subcases.

If at the time of the packing of the second item of B_l^*, the bin B_1^* is open, then for the weight s_2 of this item $\frac{1}{3}+\Delta < s_2 \leqq 1/2$ again holds. But then

$$s(B_1^*)+s(B_l^*) \geqq \frac{2}{3}-\Delta+2\left(\frac{1}{3}+\Delta\right) \geqq 2\cdot\frac{2}{3},$$

and so

$$s = s(B_1^*)+s(B_l^*)+\sum_{t\neq 1, t\neq l} s(B_t^*) \geqq 2\cdot\frac{2}{3}+k\cdot\frac{2}{3}-(q+2)\frac{1}{6},$$

which yields the validity of our statement.

If at the time of the packing of the second item of B_l^* the bin B_1^* is closed, then up to the closing of B_1^* the content of B_l^* is not greater than $\frac{1}{2}$. This results that the weight of the first packed item in B^*_{l+u} is greater than 1/2 if $u \in \{1, \dots, k+1-l\}$. This means that the bins $B^*_{l+1}, \dots, B^*_{k+1}$ are of types A or C, which contradicts our assumption on B^*_j. Therefore, this case is impossible.

Case 2. Let us suppose that $j<l$. Then $2 \leqq j < l \leqq k+1$, and so, at the opening of B^*_l the bin B^*_j is open. Next, in the same way as in Case 1 we obtain that $\frac{1}{3}+\Delta< <s_1 \leqq 1/2$ holds for the first packed item in B^*_l.

If at the time of the packing of the second item of B^*_l the bin B^*_j is open, then, similarly as in Case 1, we obtain the validity of (2).

If at the considered time B^*_j is closed, then up to the closing of B^*_j the content of B^*_l is not greater than 1/2. This yields that the weight of the first packed item in B^*_{l+u} is greater than 1/2 if $u \in \{1, \dots, k+2-l\}$. But then, $B^*_{l+1}, \dots, B^*_{k+2}$ are of types A or C, which contradicts our assumption. Therefore this case is impossible, which completes the proof of Lemma 3.

Lemma 4. For any parcel consisting of m bins, the following assertions hold:

(I) there exists at most one D-bin among the last $z=\min\{k, m\}$ bins of the parcel;

(II) if $m \geqq k+2$, then the type of the block consisting of the last $k+2$ bins of the parcel is less than 4;

(III) if the first block among two successive blocks of the parcel is of type 4, then the type of the second block is 1 or 2, and in the last case the block contains at least one A-bin.

Proof. For assertions (I) and (II), we have to distinguish two subcases according to the definition of the parcel.

Case I/a. Let us suppose that the last z bins of the considered parcel are the last z bins of the packing of L. Then, these bins are all open at the packing of the very last item of L. Let $B^*_1, \dots, B^*_z$ denote the considered bins and let us assume that B^*_i and B^*_j are D-bins, where $1 \leqq i < j \leqq z$. Then B^*_i has empty room with weight $\frac{1}{3}+\Delta$, where $\Delta>0$. Therefore, $\frac{1}{3}+\Delta<s_1 \leqq 1/2$ holds for the first packed item (s_1) in B^*_j, and so $s_2<1/3$ holds for the weight s_2 of the second item of B^*_j. At the time of the packing of this item, the bin B^*_i is open and has empty room with weight $\frac{1}{3}+\Delta$; thus the NkF algorithm places this item in B^*_i, which is a contradiction. Therefore, there exists at most one D-bin among the considered z bins.

Case I/b. Let us suppose that the considered $B^*_1, \dots, B^*_z$ bins are not the last z bins of the packing of L and that $s(B^*_{z+1}) \leqq 1/2$ holds for the following bin B^*_{z+1} of the packing. Now let us assume that B^*_i and B^*_j are D-bins, where $1 \leqq i < j \leqq z$. Then B^*_i has empty room with weight $\frac{1}{3}+\Delta$, where $\Delta>0$. Therefore, $\frac{1}{3}+\Delta<s_1 \leqq 1/2$ holds for the first packed item (s_1) in B^*_j, and so $s_2<1/3$ holds for the weight s_2 of the second item of B^*_j. Thus, at the time of the packing of this item the bin B^*_i

is closed. This yields that, up to the closing of B_i^* the content of B_j^* is not greater than 1/2. But then, the weight of the first packed item in B_{j+u}^* is greater than 1/2 if $u\in\{1, ..., z-j+1\}$. This contradicts our assumption on B_{z+1}^*. Therefore, there is at most one D-bin among the considered z bins.

Case II/a. Let us assume that the bins of the considered block are the last $k+2$ bins of the packing of L and that the block is of type 4. Let us $B_1^*, ..., B_{k+2}^*$ denote the considered bins. Then, by using Lemma 1 twice, we obtain that B_1^* and B_{k+2}^* are D-bins. Now let us suppose that B_j^* is the intermediate D-bin for some $2\leqq j\leqq k+1$. If $j>2$, then B_2^* is a C-bin, since the block contains only D-bins and C-bins. But then, by Lemma 2, we obtain that the bins $B_{j+1}^*, ..., B_{k+2}^*$ are not of type D, which is a contradiction. Thus, $j=2$ and $B_3^*, ..., B_{k+1}^*$ are of type C. Since the considered $k+2$ bins are the last $k+2$ bins of the packing, the bins B_3^*, ..., B_{k+2}^* are all open when the second item is placed in B_{k+2}^*. On the other hand, B_3^* is a C-bin, and so it has empty room with weight $\frac{1}{3}+\Delta$, where $\Delta>0$. Thus, $\frac{1}{3}+\Delta<s_1\leqq\leqq 1/2$ holds for the weight s_1 of B_{k+2}^* and $s_2<1/3$ holds for the weight s_2 of the second item of B_{k+2}^*. But, at the time of the packing of this item, B_3^* is open and it has empty room with weight $\frac{1}{3}+\Delta$; thus the NkF algorithm places this item in B_3^*, which is a contradiction. Therefore, the type of the considered block is less than 4.

Case II/b. Let us suppose that the considered m bins are not the last m bins of the packing and that $s(B')\leqq 1/2$ holds for the bin B' immediately succeeding the last bin of the parcel. Moreover, let us assume that the block is of type 4. Let $B_1^*, ..., B_{k+2}^*$ denote the bins of the block, and let B_{k+3}^* denote the bin B'. Then, by our assumption, $s(B_{k+3}^*)\leqq 1/2$. Now, in the same ways as in Case II/a, we obtain that B_1^*, B_2^*, B_{k+2}^* are D-bins and $B_3^*, ..., B_{k+1}^*$ are C-bins. Since B_3^* is a C-bin, it has empty room with weight $\frac{1}{3}+\Delta$, where $\Delta>0$. Thus, $\frac{1}{3}+\Delta<s_1\leqq 1/2$ holds for the weight s_1 of the first packed item in B_{k+2}^*. On the other hand, B_{k+2}^* is a D-bin, and so $s_2<1/3$ holds for the weight s_2 of the second item of B_{k+2}^*. Thus, at the time of the packing of this item, the bin B_3^* is closed. Therefore, up to the closing of B_3^* the content of B_{k+2}^* is not greater than 1/2. But then, the weight of the first packed item in B_{k+3}^* is greater than 1/2, which contradicts our assumption. Therefore, the type of the considered block is less than 4.

Case III. Let us suppose that the parcel contains two successive blocks and that the first of them is of type 4. Let $B_1^*, ..., B_{k+2}^*$ denote the bins of the first block and $B_{k+3}^*, ..., B_{2k+4}^*$ the bins of the second block. Then, in the same way as above, we obtain that B_1^*, B_2^*, B_{k+2}^* are D-bins and B_{k+1}^* is a C-bin. But then, by Lemma 2, the bins $B_{k+3}^*, ..., B_{2k+1}^*$ are of types A or C. On the other hand, $B_{k+4}^*, ..., B_{2k+4}^*$ are $k+1$ successive bins of the parcel, and so, by Lemma 1, there are at most two D-bins among these bins. Since B_{k+3}^* is of type A or C, we obtain that the second block contains at most two D-bins. Now let us investigate the bins $B_{k+3}^*, ..., B_{2k+1}^*$. Since $k\geqq 3$, the number of the investigated bins is at least 2. If there exists an A-bin among these bins, then assertion (III) obviously holds. In the opposite case, B_{k+3}^* and B_{k+4}^* are C-bins. On the other hand, the bins $B_{k+4}^*, ..., B_{2k+4}^*$ are $k+1$ successive bins of the parcel, and so, by Lemma 2, we obtain that there exists at most one D-bin among these bins, which results the validity of assertion (III).

This ends the proof of Lemma 4.

For any parcel consisting of m bins let s denote the sum of the weights of the items contained in the bins of the parcel and let q' and q denote the numbers of its A-bins and C-bins, respectively. Let $w=q+q'$. Then, the following statement holds.

Theorem 1. For any parcel consisting of m bins

$$s \geqq \frac{2}{3}m-\frac{1}{6}(w+1)-\frac{1}{3}\frac{m-1}{k+2}.$$

Proof. Depending on the value of m, we distinguish five cases.

1. $m=0$. In this case the statement obviously holds.

2. $1 \leqq m \leqq k$. Then, by assertion (I) of Lemma 4, we obtain that the parcel contains at most one D-bin, and so

$$s \geqq \frac{2}{3}m-\frac{1}{6}(q+1) \geqq \frac{2}{3}m-\frac{1}{6}(w+1)-\frac{1}{3}\frac{m-1}{k+2}.$$

3. $m=k+1$. Then, by Lemma 1, the parcel contains at most two D-bins, and so

$$s \geqq \frac{2}{3}m-\frac{1}{6}(q+2) \geqq \frac{2}{3}m-\frac{1}{6}(w+1)-\frac{1}{3}\frac{m-1}{k+2}.$$

4. $m=r(k+2)$ where r is a positive integer. Let us index the successive blocks with the numbers $1, \ldots, r$ according to their sequence, and let $I=\{1, \ldots, r\}$. Let $i \in I$ and let q_i' and q_i denote the numbers of A-bins and C-bins of the i-th block, respectively. For any index $j \in \{1, \ldots, 4\}$, let u_j denote the number of j-blocks and I_j the set of indices of these blocks. By assertion (II) of Lemma 4, the r-th block is not a 4-block, and so, there exists a further block for any 4-block from the considered blocks. On the other hand, by assertion (III) of Lemma 4, the block succeeding some 4-block of the parcel is of type 1 or 2. Using this observation, we classify the 4-blocks into the following two classes.

The first class contains all 4-blocks for which the following block is of type 1. Let u_{41} denote the number of these 4-blocks and I_{41} the set of their indices.

The other class contains the remaining 4-blocks.

The block succeeding some 4-block from this class is then of type 2. Let u_{42} denote the number of the blocks of the second class and I_{42} the set of their indices.

It is now obvious that $u_4=u_{41}+u_{42}$, $I_4=I_{41} \cup I_{42}$, $u_1+u_2+u_3+u_4=r$ and $\bigcup_{j=1}^{4} I_j=I$. Using the introduced notations, by Lemma 3, we obtain

$$s \geqq \sum_{j=1}^{2} \sum_{i \in I_j} \left[(k+2)\frac{2}{3}-(q_i+j)\frac{1}{6}\right]+\sum_{i \in I_3 \cup I_4} \left[(k+2)\frac{2}{3}-(q_i+3)\frac{1}{6}\right],$$

and so

$$s \geqq \sum_{i\in I}(k+2)\frac{2}{3}-\frac{1}{6}\sum_{i\in I} q_i-\frac{1}{6}\sum_{i\in I_1}1-\frac{1}{6}\sum_{i\in I_2}2-\frac{1}{6}\sum_{i\in I_3\cup I_4}3=$$

$$=\frac{2}{3}m-\frac{1}{6}q-\frac{1}{6}u_1-\frac{2}{6}u_2-\frac{3}{6}(u_3+u_4)=$$

$$=\frac{2}{3}m-\frac{1}{6}q-\frac{2}{6}(u_1+u_2+u_3+u_4)+\frac{1}{6}u_1-\frac{1}{6}u_3-\frac{1}{6}u_4=$$

$$=\frac{2}{3}m-\frac{1}{6}q-\frac{1}{3}r+\frac{1}{6}u_1-\frac{1}{6}u_3-\frac{1}{6}u_{41}-\frac{1}{6}u_{42}=$$

$$=\frac{2}{3}m-\frac{1}{6}q-\frac{1}{3}\frac{m}{k+2}+\frac{1}{6}(u_1-u_{41})-\frac{1}{6}u_3-\frac{1}{6}u_{42}.$$

From the definition of u_{41}, it follows that $u_1 \geqq u_{41}$. Thus

$$s \geqq \frac{2}{3}m-\frac{1}{6}q-\frac{1}{3}\frac{m}{k+2}-\frac{1}{6}u_3-\frac{1}{6}u_{42}=$$

$$=\frac{2}{3}m-\frac{1}{6}\Big(1+q+\sum_{i\in I}q_i'\Big)+\frac{1}{6}+\frac{1}{6}\sum_{i\in I}q_i'-\frac{1}{3}\frac{m}{k+2}-\frac{1}{6}u_3-\frac{1}{6}u_{42}=$$

$$=\frac{2}{3}m-\frac{1}{6}(w+1)-\frac{1}{3}\frac{m}{k+2}+\frac{1}{6}+\frac{1}{6}\Big(\sum_{i\in I_3}q_i'-u_3\Big)+\frac{1}{6}\Big(\sum_{i\in I_2}q_i'-u_{42}\Big)+\frac{1}{6}\sum_{i\in I_1\cup I_4}q_i'.$$

From the definition of 3-blocks, we obtain that $\sum_{i\in I_3} q_i'-u_3\geqq 0$, moreover, from Lemma 4 and from the definition of u_{42} it follows that $\sum_{i\in I_2} q_i'-u_{42}\geqq 0$. Therefore,

$$s \geqq \frac{2}{3}m-\frac{1}{6}(w+1)-\frac{1}{3}\frac{m}{k+2}+\frac{1}{6}+\frac{1}{6}\sum_{i\in I_1\cup I_4}q_i'.$$

On the other hand, $\sum_{i\in I_1\cup I_4} q_i'\geqq 0$, and so we obtain the following inequality:

(i) $$s \geqq \frac{2}{3}m-\frac{1}{6}(w+1)-\frac{1}{3}\frac{m-1}{k+2}+\frac{1}{6}-\frac{1}{3(k+2)}.$$

Since $k\geqq 3$, $\frac{1}{6}-\frac{1}{3(k+2)}\geqq 0$. But then

$$s \geqq \frac{2}{3}m-\frac{1}{6}(w+1)-\frac{1}{3}\frac{m-1}{k+2},$$

which completes the proof of this case.

5. $m=r(k+2)+l$, where r and l are positive integers and $1\leqq l\leqq k+1$. We dinstinguish two cases depending on the r-th block.

Case 5/a. Let us suppose that the r-th block is not of type 4. Then disregarding the last l bins, for the remaining $r(k+2)$ bins the same conditions holds as in the previous case. Thus, for the sum $\bar{s}$ of the weights of the items contained in these bins the inequality (i) holds, i.e.

$$\bar{s} \geqq \frac{2}{3} r(k+2) - \frac{1}{6}\Big(1 + \sum_{i=1}^{r} (q_i + q_i')\Big) - \frac{1}{3} \frac{r(k+2)-1}{k+2} + \frac{1}{6} - \frac{1}{3(k+2)}.$$

On the other hand, it may be observed that the last l bins form a parcel consisting of l bins. Thus for the sum $\tilde{s}$ of the weights of the items contained in these bins, it holds that

$$\tilde{s} \geqq \frac{2}{3} l - \frac{1}{6}(\bar{q} + \bar{q}' + 1) - \frac{1}{3} \frac{l-1}{k+2}$$

where $\bar{q}$ and $\bar{q}'$ denote the numbers of A-bins and C-bins, respectively, for the last l bins. Now, using the above inequalities, we obtain that

$$s = \bar{s} + \tilde{s} \geqq \frac{2}{3} m - \frac{1}{6}(w+1) - \frac{1}{3} \frac{m-1}{k+2}.$$

Case 5/b. Now let us suppose that the r-th block is of type 4. Then, by assertion (III) of Lemma 4, the $(r-1)$-th block is not of type 4, assuming that there exists such a block, i.e. $r>1$. Then, disregarding the last $k+2+l$ bins, for the remaining $(r-1)(k+2)$ bins the same conditions hold as above, and so, for the sum $\bar{s}$ of the weights of the items contained in these bins the inequality (i) holds. Thus,

$$\bar{s} \geqq \frac{2}{3}(r-1)(k+2) - \frac{1}{6}\Big(1 + \sum_{i=1}^{r-1} (q_i + q_i')\Big) - \frac{1}{3} \frac{(r-1)(k+2)-1}{k+2} + \frac{1}{6} - \frac{1}{3(k+2)}.$$

It may be observed that the right-hand side of the inequality is equal to 0, if $r=1$. Therefore, we may use it in the case $r=1$, too.

We now investigate the remaining $k+2+l$ bins. Let $B_1^*, \dots B_{k+2+l}^*$ denote them. Since the bins $B_1^*, \dots B_{k+2}^*$ form a 4-block, the bins B_1^*, B_2^*, B_{k+2}^* are D-bins and $B_3^* \dots, B_{k+1}^*$ are C-bins. Let us distinguish two cases depending on l.

If $l \leqq k-1$ then, by Lemma 2, the bins $B_{k+3}^*, \dots B_{k+2+l}^*$ are of type A or C. Thus, for the sum $\tilde{s}$ of the weights of the items contained in the considered $k+2+l$ bins the following inequality holds

$$\tilde{s} \geqq \frac{2}{3}(k+2) - \frac{1}{6}(q_r + 3) + \frac{2}{3} l - \frac{1}{6}\bar{q} = \frac{2}{3}(k+2+l) - \frac{1}{6}(q_r + \bar{q} + 3),$$

where $\bar{q}$ denotes the number of C-bins with respect to the last l bins.

If $k-1 < l \leqq k+1$ then it may be observed that since B_{k+1}^* is a C-bin and B_{k+2}^* is a D-bin, by Lemma 2, the bins $B_{k+3}^*, \dots, B_{2k+1}^*$ are of types A or C.

If there exists at least one A-bin among $B_{k+3}^*, \dots, B_{2k+1}^*$, then $\bar{q}' \geqq 1$, where $\bar{q}'$ denotes the number of A-bins for the last l bins. On the other hand, the bins $B_{k+4}^*, \dots$ $\dots, B_{k+2+l}^*$ form the last $l-1$ bins of the parcel, and so, by (I) of Lemma 4, there exists at most one D-bin among them. Therefore, we obtain that there is at most one D-bin among the last l bins. Thus for $\tilde{s}$ we have

$$\tilde{s} \geqq \frac{2}{3}(k+2)-\frac{1}{6}(q_r+3)+\frac{2}{3}l-\frac{1}{6}(\bar{q}+1) =$$

$$= \frac{2}{3}(k+2+l)-\frac{1}{6}(q_r+\bar{q}+\bar{q}'+3)+\frac{1}{6}(\bar{q}'-1) \geqq$$

$$\geqq \frac{2}{3}(k+2+l)-\frac{1}{6}(q_r+\bar{q}+\bar{q}'+3).$$

If the bins $B^*_{k+3}, \ldots, B^*_{2k+1}$ are all C-bins, then after the packing B^*_{2k+1} has empty room with weight $\frac{1}{3}+\Delta$, where $\Delta>0$. From this, similarly as in the proof of assertion (I) of Lemma 4, we obtain that the remaining bins (B^*_{2k+2} or B^*_{2k+2}, B^*_{2k+3}) are not of type D. But then there is no D-bin among the last l bins, and so

$$\tilde{s} \geqq \frac{2}{3}(k+2)-\frac{1}{6}(q_r+3)+\frac{2}{3}l-\frac{1}{6}\bar{q} = \frac{2}{3}(k+2+l)-\frac{1}{6}(q_r+\bar{q}+3)$$

where $\bar{q}$ denotes the number of C-bins for the last l bins again.

Now, using the common lower bound, we obtain the following inequalities:

$$s = \bar{s}+\tilde{s} \geqq \frac{2}{3}m-\frac{1}{6}\Big(1+q_r+\bar{q}+\bar{q}'+\sum_{i=1}^{r-1}(q_i+q_i')\Big)-\frac{3}{6}-\frac{1}{3}\frac{(r-1)(k+2)}{k+2}+\frac{1}{6} =$$

$$= \frac{2}{3}m-\frac{1}{6}(w+1)+\frac{1}{6}q_r'-\frac{1}{3}\frac{r(k+2)}{k+2} =$$

$$= \frac{2}{3}m-\frac{1}{6}(w+1)-\frac{1}{3}\frac{m-1}{k+2}+\frac{l-1}{3(k+2)}+\frac{1}{6}q_r'.$$

Since $q_r' \geqq 0$ and $l \geqq 1$, we obtain that

$$s \geqq \frac{2}{3}m-\frac{1}{6}(w+1)-\frac{1}{3}\frac{m-1}{k+2}$$

which completes the proof of Theorem 1.

Now let L be an arbitrary list and let us pack the elements of L with the NkF algorithm. Let $B_1, \ldots, B_m$ denote the sequence of bins used by NkF and let w denote the number of all bins containing items with weight greater than 1/2. Then, for $s=s(L)$, the following statement holds.

Theorem 2.

$$s \geqq \frac{2}{3}m-\frac{1}{6}w-\frac{m}{3(k+2)}-\frac{5}{6}.$$

Proof. We distinguish two cases, depending on the contents of the bins.

Case 1. Let us suppose that $s(B_i)>1/2$ $(i=1, \ldots, m)$. Then, the considered bins form a parcel consisting of m bins, and so, by Theorem 1, we obtain the validity of Theorem 2.

Case 2. Let us suppose that there exists a bin B_i $(1\leqq i\leqq m)$ with $s(B_i)\leqq 1/2$. Let $i_1, i_2, \ldots, i_r$ denote the increasing sequence of indices of all such bins. Let $t\in\{i_1, \ldots \ldots, i_r\}$ be arbitrary, and let us investigate the contents of B_t and $B_{t+1}, \ldots, B_{t+k}$, assuming that there exist such bins. After the packing of L, the relation $s(B_t)\leqq 1/2$ holds; thus, throughout the packing too, $s(B_t)\leqq 1/2$. But then, the weight of the first packed item in B_{t+u} is greater than $1/2$ if $u\in\{1, \ldots, k\}$. Therefore, $i_q+k< <i_{q+1}$ $(q=1, \ldots, r-1)$ and, if $i_r<m$, then the weight of the first packed item in B_{i_r+u} is greater than $1/2$ for any $1\leqq u\leqq z=\min\{k, m-i_r\}$. We now distinguish further two cases.

Case 2/a. Let us suppose that $i_r+k\leqq m$. Then the weight of the first packed item in B_{i_t+u} is greater than $1/2$ if $1\leqq t\leqq r$; $1\leqq u\leqq k$. Thus, for the sum $\bar{s}_t$ of the weights of the items contained in the bins $B_{i_t}, B_{i_t+1}, \ldots, B_{i_t+k}$, the inequality $\bar{s}_t\geqq(k+1)\frac{1}{2}$ holds, since $s(B_{i_t})+s(B_{i_t+1})>1$ and $s(B_{i_t+u})>1/2$ if $2\leqq u\leqq k$. On the other hand, it may be observed that the sequence

$$B_1, \ldots, B_{i_1-1};\ B_{i_1+k+1}, \ldots, B_{i_2-1};\ \ldots;\ B_{i_{r-1}+k+1}, \ldots, B_{i_r-1};\ B_{i_r+k+1}, \ldots, B_m$$

form parcels consisting of i_1-1, $i_2-i_1-k-1, \ldots, i_r-i_{r-1}-k-1$, $m-i_r-k$ bins, respectively, where any parcel of them may be an empty one. Let $m_1=i_1-1$, $m_2= =i_2-i_1-k-1, \ldots, m_r=i_r-i_{r-1}-k-1$, $m_{r+1}=m-i_r-k$ and let w_i denote the number of A-bins and C-bins of the i-th parcel for any $i\in\{1, \ldots, r+1\}$. Then, by Theorem 1, for the sum s_i of the weights of the items contained in the bins of the i-th parcel the following inequality holds:

$$\text{(a)}\quad s_i\geqq\frac{2}{3}m_i-\frac{1}{6}(w_i+1)-\frac{m_i-1}{3(k+2)}.$$

But then for the sum s of the weights of the items contained in the bins $B_1, \ldots, B_m$ we obtain

$$s=\sum_{i=1}^{r+1}s_i+\sum_{t=1}^{r}\bar{s}_t\geqq\sum_{i=1}^{r+1}s_i+r(k+1)\frac{1}{2}\geqq$$

$$\geqq\sum_{i=1}^{r+1}\left(\frac{2}{3}m_i-\frac{1}{6}(w_i+1)-\frac{m_i-1}{3(k+2)}\right)+r(k+1)\frac{1}{2}=$$

$$=\frac{2}{3}\Big(\sum_{i=1}^{r+1}m_i+r(k+1)\Big)-\frac{1}{6}\Big(\sum_{i=1}^{r+1}w_i+rk\Big)-\frac{2r+1}{6}-\frac{\sum_{i=1}^{r+1}(m_i-1)}{3(k+2)}.$$

Since $m=\sum_{i=1}^{r+1} m_i+r(k+1)$ and $w=\sum_{i=1}^{r+1} w_i+rk$, we have

$$s \geqq \frac{2}{3}m-\frac{1}{6}w-\frac{r}{3}-\frac{1}{6}-\frac{\Sigma m_i}{3(k+2)}+\frac{r+1}{3(k+2)}=$$

$$=\frac{2}{3}m-\frac{1}{6}w-\frac{\Sigma m_i+r(k+2)}{3(k+2)}-\frac{1}{6}+\frac{r+1}{3(k+2)}=$$

$$=\frac{2}{3}m-\frac{1}{6}w-\frac{m}{3(k+2)}+\frac{1}{3(k+2)}-\frac{1}{6}.$$

But $\frac{1}{3(k+2)}-\frac{1}{6}\geqq-\frac{5}{6}$, and so,

$$s \geqq \frac{2}{3}m-\frac{1}{6}w-\frac{m}{3(k+2)}-\frac{5}{6}$$

which completes the proof of this case.

Case 2/b. Let us assume that $i_r+k>m$. Then the number of bins succeeding the bin B_{i_r} is $l=m-i_r$. Moreover, if $l>0$, then the weight of the first packed item in B_{i_r+u} is greater than 1/2 for any $u\in\{1, \ldots, l\}$. Thus, for the sum s^* of the weights of the items contained in the bins $B_{i_r}, \ldots, B_m$ the following inequality holds

$$s^* \geqq s(B_{i_r})+(m-i_r)\frac{1}{2}.$$

On the other hand, for the sum $\bar{s}_t$ of the weights of the items contained in B_{i_t}, $B_{i_t+1}, \ldots, B_{i_t+k}$ again $\bar{s}_t\geqq(k+1)\frac{1}{2}$ holds if $t<r$. Finally, the sequences $B_1, \ldots$ $\ldots, B_{i_1-1}$; $B_{i_1+k+1}, \ldots, B_{i_2-1}; \ldots; B_{i_{r-1}+k+1}, \ldots, B_{i_r-1}$ again form parcels. Thus, with the notations of the previous case and inequality (a), for the sum s of the weights of the items contained in $B_1, \ldots, B_m$, the following inequalities hold:

$$s=\sum_{i=1}^{r} s_i+\sum_{t=1}^{r-1}\bar{s}_t+s^* \geqq \sum_{i=1}^{r} s_i+(r-1)(k+1)\frac{1}{2}+s^* \geqq$$

$$\geqq \sum_{i=1}^{r}\left(m_i\frac{2}{3}-\frac{1}{6}(w_i+1)-\frac{m_i-1}{3(k+2)}\right)+(r-1)(k+1)\frac{1}{2}+s(B_{i_r})+(m-i_r)\frac{1}{2}=$$

$$=\frac{2}{3}\Big(\sum_{i=1}^{r} m_i+(r-1)(k+1)\Big)-\frac{1}{6}\Big(\sum_{i=1}^{r} w_i+r+(r-1)(k+1)\Big)-$$

$$-\frac{\sum_{i=1}^{r}(m_i-1)}{3(k+2)}+s(B_{i_r})+(m-i_r)\frac{1}{2}=\frac{2}{3}\Big(\sum_{i=1}^{r} m_i+(r-1)(k+1)+(m-i_r)\Big)-$$

$$-\frac{1}{6}\Big(\sum_{i=1}^{r} w_i+(r-1)(k+1)+(m-i_r)+r\Big)-\frac{\sum_{i=1}^{r}(m_i-1)}{3(k+2)}+s(B_{i_r}).$$

Since $\sum_{i=1}^{r} m_i+(r-1)(k+1)+m-i_r=m-1$, we obtain that

$$s \geqq \frac{2}{3}(m-1)-\frac{1}{6}\Big(\sum_{i=1}^{r} w_i+(r-1)k+m-i_r\Big)-\frac{1}{6}(2r-1)-\frac{\sum_{i=1}^{r}(m_i-1)}{3(k+2)}+s(B_{i_r}).$$

Now, it may be observed that $w=\sum_{i=1}^{r} w_i+(r-1)k+m-i_r$, and so

$$s \geqq \frac{2}{3}(m-1)-\frac{1}{6}w-\frac{r-1}{3}-\frac{1}{6}-\frac{\sum_{i=1}^{r} m_i}{3(k+2)}+\frac{r}{3(k+2)}+s(B_{i_r}) =$$

$$= \frac{2}{3}(m-1)-\frac{1}{6}w-\frac{(r-1)(k+2)+\sum_{i=1}^{r} m_i}{3(k+2)}+\frac{r}{3(k+2)}-\frac{1}{6}+s(B_{i_r}) =$$

$$= \frac{2}{3}(m-1)-\frac{1}{6}w-\frac{\sum_{i=1}^{r} m_i+(r-1)(k+1)}{3(k+2)}+\frac{1}{3(k+2)}-\frac{1}{6}+s(B_{i_r}) =$$

$$= \frac{2}{3}(m-1)-\frac{1}{6}w-\frac{\sum_{i=1}^{r} m_i+(r-1)(k+1)+m-i_r+1}{3(k+2)}+\frac{m-i_r+2}{3(k+2)}-\frac{1}{6}+s(B_{i_r}) =$$

$$= \frac{2}{3}m-\frac{1}{6}w-\frac{m}{3(k+2)}+\frac{m-i_r+2}{3(k+2)}-\frac{5}{6}+s(B_{i_r}).$$

But $\frac{m-i_r+2}{3(k+2)}-\frac{5}{6}+s(B_{i_r}) \geqq -\frac{5}{6}$, and so

$$s \geqq \frac{2}{3}m-\frac{1}{6}w-\frac{m}{3(k+2)}-\frac{5}{6}$$

which completes the proof of Theorem 2.

We can now prove the following result.

Theorem 3.

$$R_{NkF} \leqq \frac{7}{4}+\frac{7}{4}\cdot\frac{1}{2k+3}.$$

Proof. Let L be an arbitrary list and let us pack its elements with the NkF algorithm. Let m denote the number of bins used by NkF and let s denote the sum of the weights of the items contained in these bins. Moreover, let w denote the number of those bins which contain some item with weight greater than 1/2. Now, depending on w we distinguish three cases.

Case 1. Let us suppose that $w=0$. Then, by Theorem 2, we obtain

$$s \geqq \frac{2}{3}m - \frac{m}{3(k+2)} - \frac{5}{6}.$$

On the other hand $s \leqq \mathrm{OPT}(L)$, and so

$$\frac{m}{\mathrm{OPT}(L)} \leqq \frac{m}{s} \leqq \frac{1}{\frac{2}{3} - \frac{1}{3(k+2)} - \frac{5}{6m}} \leqq$$

$$\frac{1}{\frac{2}{3} - \frac{1}{6}\cdot\frac{4k+6}{7(k+2)} - \frac{1}{3(k+2)} - \frac{5}{6m}} = \frac{7(k+2)}{4k+6-\frac{7(k+2)}{m}\frac{5}{6}}.$$

Case 2. Let us assume that $w \neq 0$ and $\frac{m}{w} \leqq \frac{7(k+2)}{4k+6}$. From the definition of w it follows that $w \leqq \mathrm{OPT}(L)$. But then

$$\frac{m}{\mathrm{OPT}(L)} \leqq \frac{m}{w} \leqq \frac{7(k+2)}{4k+6}.$$

Case 3. Let us suppose that $w \neq 0$ and $\frac{7(k+2)}{4k+6} < \frac{m}{w}$. Again, by Theorem 2,

$$s \geqq \frac{2}{3}m - \frac{1}{6}w - \frac{m}{3(k+2)} - \frac{5}{6},$$

and so,

$$\frac{m}{\mathrm{OPT}(L)} \leqq \frac{m}{s} \leqq \frac{m}{\frac{2}{3}m - \frac{1}{6}w - \frac{m}{3(k+2)} - \frac{5}{6}} = \frac{1}{\frac{2}{3} - \frac{1}{6}\frac{w}{m} - \frac{1}{3(k+2)} - \frac{5}{6m}}.$$

By our assumption on m/w, $\frac{w}{m} < \frac{4k+6}{7(k+2)}$, and so

$$\frac{m}{\mathrm{OPT}(L)} \leqq \frac{1}{\frac{2}{3} - \frac{1}{6}\frac{4k+6}{7(k+2)} - \frac{1}{3(k+2)} - \frac{5}{6m}} = \frac{7(k+2)}{4k+6-\frac{7(k+2)}{m}\frac{5}{6}}.$$

Now let $k \geqq 3$ be a fixed integer. It may be observed that if $\mathrm{OPT}(L) \to \infty$ then $m \to \infty$, and so, under the fixed k, $\frac{7(k+2)}{m}\frac{5}{6} \to 0$.

Therefore

$$\limsup_{n\to\infty}\left\{\frac{NkF(L)}{\mathrm{OPT}(L)} : \mathrm{OPT}(L) = n\right\} \leqq \frac{7(k+2)}{4k+6} = \frac{7}{4} + \frac{7}{4}\cdot\frac{1}{2k+3},$$

which completes the proof of Theorem 3. □

We now improve the lower bound given by Johnson. For this purpose, we define a sequence of lists such that $\mathrm{OPT}(L_i) \to \infty$ and the lists have bad behaviour on NkF packing. Let j now be a fixed positive integer.

Let $n(j)$ denote the number of elements in the j-th list and let

$$n(j) = 30j(k-2)+30j.$$

Let

$$\delta \ll 18^{-j(k-2)}$$

and let $L_{n(j)}$ denote the j-th list in the sequence. We divide the items into three parts:

(1) In the first part there are elements about $1/6$; there are $j(k-2)$ blocks, with 10 items in each (thus, in the first part there are $10j(k-2)$ items). Let us denote the items of the i-th block by

$$a_{0i}, a_{1i}, \ldots, a_{9i}.$$

The exact definition of the weights is as follows. Let

$$\delta_i = \delta \cdot 18^{j(k-2)-i} \quad (1 \leqq i \leqq j(k-2))$$

and

$$a_{0i} = 1/6+33\delta_i,$$

$$a_{1i} = 1/6-3\delta_i,$$

$$a_{2i} = 1/6-7\delta_i = a_{3i},$$

$$a_{4i} = 1/6-13\delta_i,$$

$$a_{5i} = 1/6+9\delta_i,$$

$$a_{6i} = 1/6-2\delta_i = a_{7i} = a_{8i} = a_{9i}.$$

Then, the first $10j(k-2)$ items of the list are $a_{01}, a_{11}, \ldots, a_{91}, a_{02}, a_{12}, \ldots, a_{92}, \ldots$ $\ldots, a_{0,j(k-2)}, \ldots, a_{9,j(k-2)}$. Clearly

$$a_{0i}+a_{1i}+a_{2i}+a_{3i}+a_{4i} = 5/6+3\delta_i,$$

$$a_{5i}+a_{6i}+a_{7i}+a_{8i}+a_{9i} = 5/6+\delta_i,$$

and thus we fill $2j(k-2)$ bins with this part.

(2) In the second part, there are elements about $1/3$; there are also $j(k-2)$ blocks, with 10 items in each. Let us denote the items of the i-th block by

$$b_{0i}, b_{1i}, \ldots, b_{9i},$$

and the items

$$b_{01}, b_{11}, \ldots, b_{91}, b_{02}, b_{12}, \ldots, b_{92}, \ldots, b_{0,j(k-2)}, \ldots, b_{9,j(k-2)}$$

follow the items of the first part.

The exact definition of these items is as follows:

$b_{0i} = 1/3+46\delta_i,$

$b_{1i} = 1/3-34\delta_i,$

$b_{2i} = 1/3+6\delta_i = b_{3i},$

$b_{4i} = 1/3+12\delta_i,$

$b_{5i} = 1/3-10\delta_i,$

$b_{6i} = 1/3+\delta_i = b_{7i} = b_{8i} = b_{9i}.$

Clearly

$b_{0i}+b_{1i} = 2/3+12\delta_i,$

$b_{2i}+b_{3i} = 2/3+12\delta_i,$

$b_{4i}+b_{5i} = 2/3+2\delta_i,$

$b_{6i}+b_{7i} = 2/3+2\delta_i,$

$b_{8i}+b_{9i} = 2/3+2\delta_i,$

and thus we fill $5j(k-2)$ bins with the second part.

(3) In the third part, there are elements about $1/2$. We have here $10j$ blocks, with $k+1$ items in each. In the i-th block, the first item is $1/2-\delta/(i+1)$, and the second is $1/2+\delta/i$. Then, we have a number $(k-2)$ of $1/2+\delta$ items and the last item of this block is a δ. Thus, with this part we exactly fill $10jk$ bins.

On summing the number of bins in the three parts, we obtain:

$$NkF(L_{n(j)}) = 2j(k-2)+5j(k-2)+10jk = 17jk-14j.$$

In the optimal packing of $L_{n(j)}$, we have to pack all $1/2+\delta$ items in separate bins. Thus, we pair the items from the first and second part in the following way:

i) $a_{l,i}+b_{l,i}$, if $2 \leqq l \leqq 9$, $1 \leqq i \leqq j(k-2)$,

ii) $a_{0i}+b_{1i}$, if $1 \leqq i \leqq j(k-2)$,

iii) $a_{1i}+b_{0,(i+1)}$, if $1 \leqq i \leqq j(k-2)-1$.

Clearly, we can pack all pairs with a $1/2+\delta$ element together. Accordingly, we fill $10j(k-2)-1$ bins, and b_{01}, $a_{1,j(k-2)}$ are not used. From the third part, one $1/2+\delta$ item, a number $10j$ of δ items and the following items are not used:

$$\frac{1}{2}-\frac{\delta}{2}, \frac{1}{2}+\frac{\delta}{1},$$

$$\frac{1}{2}-\frac{\delta}{3}, \frac{1}{2}+\frac{\delta}{2},$$

$$\vdots$$

$$\frac{1}{2}-\frac{\delta}{10j+1}, \frac{1}{2}+\frac{\delta}{10j}.$$

Here $1/2-\delta/i$ and $1/2+\delta/i$ fill a bin $(i=2, 3, \ldots, 10j)$ and so we have a further $10j-1$ bins. All other items can be packed into three bins, if δ is small enough. Thus,

$$\mathrm{OPT}(L_{n(j)}) \leqq 10j(k-2)-1+10j-1+3 = 10jk-10j+1.$$

Then

$$\frac{NkF(L_{n(j)})}{\mathrm{OPT}(L_{n(j)})} \geqq \frac{17jk-14j}{10jk-10j+1},$$

and hence

$$R_{NkF} \geqq \liminf_{j\to\infty} \frac{NkF(L_{n(j)})}{\mathrm{OPT}(L_{n(j)})} = \frac{17k-14}{10k-10}.$$

We have obtained

Theorem 4. For $k\geqq 3$

$$R_{NkF} \geqq 1.7+\frac{3}{10(k-1)}.$$

From Theorem 3 and Theorem 4, we conclude our

Main results. For $k\geqq 3$

$$1.7+\frac{3}{10(k-1)} \leqq R_{NkF} \leqq \frac{7}{4}+\frac{7}{4}\cdot\frac{1}{2k+3}.$$

To conclude this paper, we give R_{N2F}. For this, we define a sequence of lists as follows. Here the j-th list has a number $n(j)=30j$ of items. Let

$$L_{n(j)} = \left(\frac{1}{2}-\frac{\delta}{2}, \frac{1}{2}+\frac{\delta}{1}, \delta, \frac{1}{2}-\frac{\delta}{3}, \frac{1}{2}+\frac{\delta}{2}, \delta, \ldots, \frac{1}{2}-\frac{\delta}{10j+1}, \frac{1}{2}+\frac{\delta}{10j}, \delta\right).$$

Then we use $20j$ bins in the $N2F$ packing, and $10j+1$ bins in the optimal packing. Thus, we get:

Corollary 1. $R_{N2F}=2$.

Acknowledgement. We are grateful to E. Mâté for valuable discussions.

References

[GJ] Garey, M. R., Johnson, D. S.: Computers and intractability, W. H. Freeman and Company, San Francisco, 1979.

[J] Johnson, D. S.: Fast algorithms for bin-packing. Journal of Computer and Systems Sciences 8 (1974), 272—314.

[JDUGG] Johnson, D. S., Demers, A., Ullman, J. D., Garey, M. R., Graham, R. L.: Worst-case performance bounds for simple one-dimensional packing algorithms, SIAM J. Comput. 3 (1974), 299—325.

(Received February 13, 1989)

On the performance of on-line algorithms for partition problems*

ULRICH FAIGLE,[1] WALTER KERN[1] and GYÖRGY TURÁN[2,3]

[1]*Faculty of Applied Mathematics, University of Twente NL-7500 AE Enschede, The Netherlands*
[2]*Department of Mathematics, Statistics and Computer Science University of Illinois at Chicago, Chicago, IL, 60680, USA*
and
[3]*Automata Theory Research Group of the Hungarian Academy of Sciences, Szeged, 6720, Hungary.*

Abstract

We consider the performance of the greedy algorithm and of on-line algorithms for partition problems in combinatorial optimization. After surveying known results we give bounds for matroid and graph partitioning, and discuss the power of non-adaptive adversaries for proving lower bounds.

1. Introduction

There are several combinatorial optimization problems where a set is to be partitioned into a minimal number of classes having certain properties. Examples of such problems are graph coloring and bin packing. A general heuristic to find an approximate solution is the greedy (or first-fit) method where the partition is constructed by processing the elements in some order and placing each element into the first class it fits into.

A partitioning algorithm is on-line if it considers the elements one after the other and puts each element into a class at the time when it is considered according to some rule, based on information about elements processed earlier (thus the greedy method is a special case). The main feature of an on-line algorithm is that the decision made about an element cannot be modified later on. An on-line algorithm in general does not have to be polynomial time computable or even computable..

There are several interesting results about the performance of on-line algorithms for various partition problems. After giving a general problem formulation in Section 2. we survey these results in Section 3.

* Supported by Hungarian Academy of Sciences. (OTKA Nr. 1135)

In Section 4. we consider the matroid partitioning problem and the special cases of graphic matroids and graphs. There are polynomial time algorithms solving this problem (Edmonds [6], see also Lawler [18]), but these algorithms are not on-line. We show that the performance ratio of the greedy algorithm on n element matroids is $\theta(\log n)$ and that the performance ratio of every on-line matroid partitioning algorithm is $\Omega(\log n/\log\log n)$. We also show that bounded performance is not possible even in the special case when we want to partition a graph into forests.

All known lower bound proofs for on-line algorithms are based on the construction of an adversary which plays against the algorithm by providing the new elements of the input so that the algorithm is forced to produce more classes than necessary. In many cases the adversary satisfies a condition called non-adaptiveness. In Section 5. we consider examples comparing the power of non-adaptive adversaries and general ones for lower bound proofs.

Section 6. contains some further remarks and open problems.

2. Partition problems, definitions

First we give a list of partition problems discussed later on. For definitions not given here see Bollobás [2], Lawler [18], Lovász [22], Welsh [30].

MATROID PARTITIONING: given a matroid $M=(E, \mathcal{F})$, partition the ground set E into a minimal number of independent subsets.

GRAPHIC MATROID PARTITIONING: the same as above for a graphic matroid M.

(As the complexity of the algorithms is not taken into consideration we may assume that the matroids are presented by listing their independent subsets.)

GRAPH PARTITIONING: given a graph $G=(V, E)$, partition E into a minimal number of forests.

GRAPH COLORING: given a graph $G=(V, E)$, partition V into a minimal number of independent subsets.

CHAIN DECOMPOSITION OF ORDERED SETS: given an ordered set $P=(V, <)$, partition V into a minimal number of chains.

GRAPH EDGE COLORING: given a graph $G=(V, E)$, partition E into a minimal number of matchings.

BIN PACKING: given $A=\{a_1, ..., a_n\}$ $(0<a_i\leqq 1)$, partition A into a minimal number of sets each having sum $\leqq 1$.

GRAPH BIN PACKING: given a fixed "pattern" graph $G_0=(V_0, E_0)$ and a graph $G=(V, E)$, partition E into a minimal number of sets each being a subgraph of G_0.

A common framework for considering these problems can be described using independence systems.

An independence system is a pair $I=(E, \mathcal{F})$, where E is the ground set and $\mathcal{F}\subseteq\mathcal{P}(E)$ is a set of subsets of E such that if $F\in\mathcal{F}$ and $F'\subseteq F$ then $F'\in\mathcal{F}$. An independence system is ordered if in addition there is a linear ordering $<$ on E. All ordered independence systems considered here are finite, we write $I_n=(E_n, \mathcal{F}_n)$, $E_n=\{e_1, ..., e_n\}$, $e_1<...<e_n$. An ordered independence system $I_k=(E_k, \mathcal{F}_k)$ is an initial segment of I_n (denoted by $I_k<I_n$) if $k\leqq n$, $E_k=\{e_1, ..., e_k\}$ and for every $F\subseteq E_k$ it holds that $F\in\mathcal{F}_k$ iff $F\in\mathcal{F}_n$.

An independent partition of $I=(E, \mathscr{F})$ is an ordered partition $(F_1, \ldots, F_l)$ of E such that $F_i \in \mathscr{F}$ $(1 \leqq i \leqq l)$. Let $p(I) := \min \{l$: there is an independent partition $(F_1, \ldots, F_l)$ of $E\}$.

Let $\mathscr{I}$ be a class of finite independence systems. The PARTITION PROBLEM FOR $\mathscr{I}$ is the following problem: given $I=(E, \mathscr{F}) \in \mathscr{I}$, find an independent partition of E into $p(I)$ sets.

Assume that furthermore $\mathscr{I}$ consists of ordered independence systems and is closed under taking initial segments (i.e. $I \in \mathscr{I}$, $I' < I$ imply $I' \in \mathscr{I}$).

An on-line algorithm A for the partition problem for $\mathscr{I}$ is a function defined on $\mathscr{I}$ such that for every $I=(E, \mathscr{F}) \in \mathscr{I}$ $A(I)$ is an ordered independent partition of I and if $I'=(E', \mathscr{F}') < I$ then $A(I')=A(I)|E'$ i.e. $A(I')$ is the restriction of $A(I)$ to E', or equivalently, $A(I)$ is an extension of $A(I')$. Thus A provides an approximate solution to the partition problem for $\mathscr{I}$.

For the greedy algorithm A_{gr}, $A_{gr}(I_n)$ is obtained from $A_{gr}(I_{n-1})$ by placing e_n into the first subset in the ordered partition $A_{gr}(I_{n-1})$ which remains independent if e_n is added to it, and opening a new set for e_n if there is no such set.

For an on-line algorithm A let $|A(I)|$ be the number of subsets in the partition $A(I)$ and let (with some abuse of notation)

$$A(n) := \max \{|A(I)|/p(I) \colon I = (E, \mathscr{F}) \in \mathscr{I},\ |E| = n\}$$

be the performance ratio function of A. A has bounded performance with bounding function $f \colon \mathbf{N} \to \mathbf{N}$ if for every $I \in \mathscr{I}$ it holds that $|A(I)| \leqq f(p(I))$. (Thus if $A(n) \to \infty$ by considering inputs with $p(I)$ bounded by some constant then A does not have bounded performance.) The performance ratio of A is

$$r_A := \inf \{r \geqq 1 \colon |A(I)| \leqq rp(I) \text{ for every } I \in \mathscr{I}\}$$

and the asymptotic performance ratio of A is

$$r_A^\infty := \inf \{r \geqq 1 \colon \exists c \colon A(I) \leqq rp(I) + c \text{ for every } I \in \mathscr{I}\}$$

(thus $r_A, r_A^\infty \in \mathbf{R} \cup \{\infty\}$). Let

$r_{\mathscr{I}} := \inf \{r_A$: A is an on-line algorithm for the partition problem for $\mathscr{I}\}$,

$r_{\mathscr{I}}^\infty := \inf \{r_A^\infty$: A is an on-line algorithm for the partition problem for $\mathscr{I}\}$.

For matroid partitioning (resp. graphic matroid partitioning) the class $\mathscr{I}$ could be the class of all finite ordered matroids (resp. finite ordered graphic matroids) on a fixed countable set. For graph partitioning the class $\mathscr{I}$ could consist of all finite (edge-)ordered subgraphs of a countable complete graph. For bin packing the class $\mathscr{I}$ could consist of all finite ordered subsets of countably many copies of (0, 1].

There is a difference between the first two and the last two examples. For matroid partitioning and graphic matroid partitioning we may assume that if $I_1 < I_2 < \ldots < I_n$, $I_1' < I_2' < \ldots < I_n'$ and $I_j \cong I_j'$ $(1 \leqq j \leqq n)$ then the partitions determined by A are also isomorphic, in particular $|A(I_n)| = |A(I_n')|$. This holds because $\mathscr{I}$ is homogeneous, i.e. every isomorphism of two inputs I and I' can be extended to an automorphism of $\mathscr{I}$. With other words on-line algorithms for these problems can only use information about independence.

For the other two problems there is additional information provided by specifying edges resp. numbers: 2 edges with or without a common endpoint are isomorphic as independence systems but none of their isomorphisms can be extended to an automorphism of $\mathscr{I}$; resp. there is no automorphism of $\mathscr{I}$ for bin packing mapping a copy of 1/2 to a copy of 1/3.

3. A survey of results about on-line algorithms

a) Graph coloring

Johnson [12] observed that $A_{gr}(n)=\Omega(n)$ even for bipartite graphs. Szegedy [25] showed that for every on-line graph coloring algorithm $A(n)=\Omega(n/(\log n)^2)$. Lovász, Saks and Trotter [23] gave an on-line algorithm with $A(n)=O(n/\log^* n)=o(n)$. For trees Bean [1] and Gyárfás and Lehel [11] noted that $A(n)=\Omega(\log n)$ for every on-line algorithm. Kierstead and Trotter [16] gave an on-line algorithm coloring interval graphs with $r_A^\infty=3$ and showed that this is best possible. Kierstead [15] showed that for this problem $r_{A_{gr}}^\infty<\infty$. Gyárfás and Lehel [11] showed that $r_{A_{gr}}^\infty<\infty$ for several special classes of graphs such as split graphs, complements of bipartite graphs and complements of chordal graphs.

b) Chain decomposition of ordered sets

Kierstead [14] proved that there is an on-line algorithm for this problem which has bounded performance with bounding function $(5^n-1)/4$. This appears to be the first result on on-line algorithms formulated in the language of recursion theoretic combinatorics. For the greedy algorithm $A_{gr}(n)=\Omega(n)$. Szemerédi [26] showed that for every on-line algorithm A and every w there are orders P with width w and $|A(P)|=\Omega(w^2)$ thus for every on-line algorithm A $r_A^\infty=\infty$. An order is an interval order if it is isomorphic to a set of intervals $\{J_1, \ldots, J_n\}$ on a line with $J_i<J_j$ iff J_i is completely to the left of J_j. Kierstead and Trotter [16] gave an on-line algorithm for interval orders with $r_A^\infty=3$ and showed that this is optimal. (We note that the difference between the chain decomposition problem and the graph coloring problem for incomparability graphs is that comparable pairs form an ordered resp. an unordered pair.) An order is series-parallel if it can be obtained from orders on one element by repeated application of series composition ("place order P_1 above P_2") and parallel composition ("let all elements of P_1 be incomparable to all elements of P_2"). If the orders are restricted to be series-parallel then the greedy algorithm always gives an optimal solution [7].

c) Graph edge coloring

If $\triangle$ is the maximal degree of the graph $G=(V, E)$ then clearly $\geqq\triangle$ colors are needed for an edge coloring of G (by Vizing's theorem [29] (see also Bollobás [2]) $\triangle+1$ colors are always sufficient). It is easy to see that the greedy algorithm never uses more than $2\triangle-1$ colors. On the other hand every on-line algorithm A uses $\geqq 2\triangle-1$ colors for some forest with maximal degree $\triangle$ (here the minimal number of colors needed is easily seen to be $\triangle$). To see this, consider first a forest of $(\triangle-1)\cdot\binom{2\triangle-2}{\triangle-1}+1$ stars with $\triangle-1$ edges. Then A either uses $\geqq 2\triangle-1$ colors or there will be $\triangle$ stars colored with the same set of $\triangle-1$ colors. Add $\triangle$ new edges by connecting a new root to the root of these stars to get a forest with maximal degree $\triangle$.

Every new edge must be colored with a color not occurring in the stars selected and thus $\geqq 2\Delta-1$ colors will be used.

d) Bin packing

Johnson, Demers, Ullman, Garey and Graham [13] showed that $r^{\infty}_{A_{gr}}=1.7$. Yao [31] gave an on-line algorithm with $r^{\infty}_{A}=5/3$. The on-line algorithm of Lee and Lee [19] has $r^{\infty}_{A}\leqq 1.692$ and it also satisfies the additional requirement of having only a bounded number of active bins at any time. Brown [4] and Liang [20] showed that $r^{\infty}_{A}\geqq 1.536$ for every on-line algorithm. This result is generalized by Galambos [8] to the case when items are from $(0, \alpha]$ $(\alpha<1)$. We note that there are polynomial time algorithms A_{ε} (which are not on-line) with $r_{A_{\varepsilon}}<1+\varepsilon$ for every $\varepsilon>0$ (de la Vega and Lueker [28]). On-line algorithms for dual bin packing (where the aim is to fill as many bins as possible) are considered by Csirik and Totik [5]. For the graph bin packing problem it is shown in [27] that for complete bipartite graphs $G_0=K_{k,l}$, $k\leqq l$, $r_{A_{gr}}=\Theta(\max(k, l/k))$, thus for fixed l the greedy algorithm has the best performance guarantee when $k\sim\sqrt{l}$.

We note that there are results about on-line algorithms for problems of a different nature than the ones discussed here (see Borodin, Linial and Saks [3], Manasse, McGeoch and Sleator [24] and the further references in these papers).

4. Matroid partitioning

First we consider the performance of the greedy algorithm. The upper bound holds for matroids in general, the lower bound already holds in the special case of graphs.

Theorem 1. a) For the matroid partitioning problem $A_{gr}(n)\leqq\ln(n)$.
b) For the graph partitioning problem $A_{gr}(n)\geqq\lfloor\log n\rfloor/2$.

Proof. a) Let $I_n=(E_n, \mathcal{F}_n)$ be a matroid and $(F_1, \ldots, F_l)$ be the partition formed by the greedy algorithm. Then F_i is a maximal independent set in $E_n\setminus(F_1\cup\ldots\cup F_{i-1})$. As I_n restricted to $E_n\setminus(F_1\cup\ldots\cup F_{i-1})$ is again a matroid, F_i is also a maximum independent set in $E_n\setminus(F_1\cup\ldots\cup F_{i-1})$. Thus $(F_1, \ldots, F_l)$ is a greedy solution of the set covering problem for I_n. The performance ratio of the greedy algorithm for set covering is $\leqq\ln(n)$ (Johnson [12], Lovász [21]).

b) For $k\geqq 1$ let $G_k:=(V_k, E_k)$, where

$$V_k=\{v_0, \ldots, v_{2^k-1}\},\quad E_k=\bigcup_{i=0}^{k-1} P_i,$$

$$P_i=\{(v_{j2^i}, v_{(j+1)2^i})\colon j=0, \ldots, 2^{k-1-i}-1\}.$$

For later use let v_0 be the initial vertex of G_k and v_{2^k-1} be the terminal vertex of G_k. Order the edges in G_k in such a way that edges in P_i precede edges in P_{i+1} $(0\leqq i\leqq k-2)$. Then the greedy algorithm gives a different color to each P_i (we refer to this partition of E_k as the greedy partition), hence for this ordering $|A_{gr}(G_k)|=k$. Note that $|E_k|=2^k-1$. On the other hand coloring the edges of P_i alternatingly red and blue (for every i) gives a partition of E_k into 2 trees and so $p(G_k)=2$. $\square$

Theorem 1. can be generalized to the case when $\mathscr{I}$ consists of independence systems that are the intersections of k matroids (thus for every $I=(E, \mathscr{F})\in\mathscr{I}$ there are k matroids $I^i=(E, \mathscr{F}^i)$ $(1\leq i\leq k)$ such that for every $F\subseteq E$, $F\in\mathscr{F}$ iff $F\in\mathscr{F}^i$ for every $i=1, \ldots, k$).

Corollary 2. Assume that for every $I\in\mathscr{I}$, I is the intersection of k matroids. Then for the partitioning problem for $\mathscr{I}$ it holds that $A_{gr}(n)\leq k\cdot\ln(n)$.

Proof. Korte and Hausmann [17] showed that if $I=(E, \mathscr{F})$ is the intersection of k matroids, F is a maximal independent set in $\mathscr{F}$ and F' is a maximum independent set in $\mathscr{F}$ then $|F|\geq(1/k)\cdot|F'|$. Thus the partition given by the greedy algorithm is a "$1/k$-greedy" solution to the set covering problem on I in the sense that we always choose a set which has size $\geq 1/k$ times the size of a largest set in the system. The proof of Johnson [12] and Lovász [21] can be applied to this case to show that the number of sets used in the covering is $\leq k\cdot\ln(n)$ times the optimal. □

Now we turn to the discussion of on-line algorithms.

Theorem 3. For every on-line matroid partitioning algorithm A $A(n)=$ $=\Omega(\log n/\log\log n)$.

Proof. For G_i constructed in the proof of Theorem 1. let $s\cdot G_i$ be the graph obtained by taking a sequence of s copies of G_i and identifying the terminal vertex of each copy (except the last one) with the initial vertex of the next one.

For a graph G let $M(G)$ be the cycle matroid of G.

Then $M(sG_i)$ is the direct sum of s copies of $M(G_i)$. (The direct sum of matroids on disjoint ground sets is obtained by taking the union of the ground sets as the new ground set and letting a subset be independent if its intersection with each ground set is independent.) If M is a matroid isomorphic to $M(sG_i)$ then it has a unique decomposition into s matroids isomorphic to $M(G_i)$, called the components of M. An ordered partition of M into independent subsets is called the greedy partition if on each component it corresponds to the greedy partition of G_i.

The graph $2G_i$ is a subgraph of G_{i+1} and therefore a matroid $M\cong M(G_i)\oplus M(G_i)$ (where $\oplus$ denotes the direct sum) can be extended to a matroid isomorphic to $M(G_{i+1})$ by adding one more element to it.

Now let $g(1):=1$, $g(k):=(k-1)(2g(k-1)-1)+1$ for $k>1$ and $f(k):=$ $:=\sum_{i\leq k} g(i)$ for $k\geq 1$.

We show that the algorithm A uses $\geq k$ colors to partition some 2-partitionable matroid on $f(k)$ elements.

Using an adversary strategy we prove that giving $g(k)+\ldots+g(k-i)$ elements $(0\leq i\leq k-2)$ to A it can be forced either to use $\geq k$ colors or to form the greedy partition on a submatroid isomorphic to $M(2g(k-i-1)G_{i+1})$.

For $i=0$, giving $g(k)$ independent elements to A it either uses $\geq k$ colors or it assigns the same color to $2g(k-1)$ elements and $M(G_1)$ consists of a single element.

For the induction step assume that after adding $g(k)+\ldots+g(k-i+1)$ elements to A it formed the greedy partition on a submatroid $M_i\cong M(2g(k-i)G_i)$. Pair the components of M_i and add $g(k-i)$ elements (one to each pair) to extend each pair to a matroid isomorphic to $M(G_{i+1})$. As A cannot use any of the i colors used for M_i it either uses $\geq k-i$ colors different from these or it assigns the same color to $2g(k-i-1)$ new elements. The union of these components is $M_{i+1} \cong \cong M(2g(k-i-1)G_{i+1})$ and A formed the greedy partition on M_{i+1}.

For $i=k-2$ we get $M_{k-1} \cong M(2G_{k-1})$ such that A formed the greedy partition on M_{k-1}. Adding a new element to obtain $M_k \cong M(G_k)$ forces A to use the k^{th} color.

As the components of the matroid M formed by all elements given to A are isomorphic to $M(G_i)$ for some i, M is 2-partitionable.

Finally the bound follows from noting that $g(k) \leqq 2kg(k-1)$, thus $g(k) \leqq 2^k \cdot k!$. Hence $f(k) \leqq 2^k \cdot k \cdot k!$ and so $k = \Omega(\log n / \log\log n)$. □

Corollary 4. For every on-line algorithm A partitioning graphic matroids $A(n) = \Omega(\log n / \log\log n)$.

Proof. All matroids constructed in the previous proof are graphic. □

We remark that the proof of Theorem 3. does not work for graphs. This is related to the remarks made following the definitions in Section 2. For graphs the adversary is in a more difficult situation as e.g. 2 independent elements in the first phase of the construction can be completed to a triangle by adding a new element if we are dealing with general (or graphic) matroids but in graphs this can only be done if the 2 edges have a common endpoint.

Let $g(1) := 1$, $g(k) := (2k)^{(k-1)(2g(k-1)-1)+1} - 1$ for $k > 1$ and $f(k) := \sum_{i \leqq k} g(i)$ for $k \geqq 1$.

Theorem 5. Every on-line graph partitioning algorithm A forms at least k classes for some 2-partitionable graph having $f(k)$ edges.

Proof. We describe an adversary strategy by induction on k, for $k=1$ the statement is obvious. First we prove a lemma.

Lemma 6. For every l $(2 \leqq l \leqq k)$, by building a forest on $g(l)+1$ vertices A can be forced either to use at least l colors or to form a monochromatic path P of length $2g(l-1)$.

Proof. A forest is rooted if each of its components has a distinguished vertex called the root. An l-edge colored rooted forest with j roots is an (i, j)-forest if there are numbers $t_1, \ldots, t_l$ with $t_1 + \ldots + t_l = i$ such that for every root v and every r $(1 \leqq r \leqq l)$ v is the endpoint of a monochromatic path of color r and length t_r.

We show that for every $i = 0, \ldots, (l-1)(2g(l-1)-1)+1$ by building a forest A can be forced either to use $\geqq l$ colors or to form an $(i, (g(l)+1)/(2l)^i)$-forest.

For $i=0$ the empty graph on $g(l)+1$ vertices is a $(0, g(l)+1)$-forest. Assume we constructed an $(i-1, (g(l)+1)/(2l)^{(i-1)})$-forest. Add $(g(l)+1)/(2(2l)^{(i-1)})$ new edges forming a matching of the roots. Then A either uses $\geqq l$ colors to color these edges or $\geqq (g(l)+1)/(2l)^i$ new edges get the same color. In this case select an endpoint of each of these edges and let them be the new roots. Deleting the components without a selected root we get an $(i, (g(l)+1)/(2l)^i)$-forest and the whole graph built is a forest.

For $i = (l-1)(2g(l-1)-1)+1$ we get an $(i, 1)$-forest, i.e. a tree with $t_1 + \ldots + t_l = (l-1)(2g(l-1)-1)+1$. Thus for some r $(1 \leqq r \leqq l)$ it holds that $t_r \geqq 2g(l-1)$. The path P required can be chosen to be the corresponding path of color r. □

Now we describe the adversary strategy S_k.

1) Force A either to use $\geqq k$ colors or to form a monochromatic path P of length $2g(k-1)$ by building a forest on a set V_k of $g(k)+1$ vertices. (This can be done by Lemma 6.)

2) Apply S_{k-1} to the set V_{k-1} consisting of every second vertex of P (thus $|V_{k-1}|=g(k-1)+1$).

Note that after completing phase 1) V_{k-1} is an independent set of vertices and in later stages the color of the path P cannot be used as otherwise a monochromatic cycle is created. Thus by induction S_k indeed forces A to use $\geqq k$ colors and the construction implies that the graph G built by the adversary has $\leqq f(k)$ edges.

Finally we claim that G is 2-partitionable. This follows by induction. Assume that the graph G' built on V_{k-1} is 2-partitionable and let (F_1, F_2) be a partition of its edges into 2 forests. Then adding the edges of P to F_1 and F_2 alternatingly and adding the remaining edges of G arbitrarily we get a 2-partition of G. $\square$

By definition, Theorem 5. implies the following.

Corollary 7. For every on-line graph partitioning algorithm A $A(n)\to\infty$ and A does not have bounded performance. $\square$

5. Non-adaptive adversaries

Several lower bounds for on-line algorithms are based on the existence of instances I such that for every independent partition of I there is an initial segment of I for which the restriction of the partition is far from being optimal. This shows that no on-line algorithm can have good performance on every initial segment of I.

Thus the adversary providing I is non-adaptive in the sense that for every algorithm A it provides a counterexample which depends on A in a very restricted way only through the choice of the initial segment of I. With other words the only liberty the adversary has is to decide when to stop giving new elements.

All known lower bounds for bin packing are non-adaptive. On the other hand the lower bounds for graph coloring and chain decomposition (e.g. [25], [14], [16]), and the lower bounds of the preceding section are adaptive, i.e. when the adversary determines the next extension of the current instance it takes into consideration the previous decisions made by the algorithm.

For $I_n=(E_n, \mathscr{F}_n)$ let $I_1<\ldots<I_n$ be the initial segments of I_n, $P_n=(F_1, \ldots, F_l)$ be an independent partition of E_n and $P_k=P_n|E_k$ $(1\leqq k\leqq n)$ be the restriction of P_n to E_k. With these notations let

$$s_{\mathscr{I}}^{\infty}:=\inf\{r\colon \exists c \forall I_n\in\mathscr{I}\exists P_n \forall P_k\colon\ |P_k|\leqq rp(I_k)+c\}$$

($s_{\mathscr{I}}$ could be defined analogously). By the argument above $s_{\mathscr{I}}^{\infty}\leqq r_{\mathscr{I}}^{\infty}$. We consider the question of how good a lower bound is $s_{\mathscr{I}}^{\infty}$ to $r_{\mathscr{I}}^{\infty}$.

For graph coloring restricted to forests clearly $s_{\mathscr{I}}^{\infty}=1$ and as mentioned in Section 3. $r_{\mathscr{I}}^{\infty}=\infty$ (as $A(n)=\Omega(\log n)$ for every on-line algorithm). We mention another example where both $s_{\mathscr{I}}^{\infty}$ and $r_{\mathscr{I}}^{\infty}$ are finite but different.

As it is mentioned in Section 3., Kierstead and Trotter [16] showed that $r_{\mathscr{J}}^{\infty}=3$ for the chain decomposition problem restricted to interval orders.

Proposition 8. For the chain decomposition problem restricted to interval orders $s_{\mathscr{J}}^{\infty}\leqq 2$.

Proof. The bound follows directly from the proof of Kierstead and Trotter [16]. Let P be an interval order of width w on the ground set $V=\{v_1, \ldots, v_n\}$. Then V is partitioned into w sets $L_1, \ldots, L_w$ by considering the elements $v_1, \ldots, v_n$ one after the other and putting each element into the first set so that the conditions $\text{width}(P_n|L_1\cup\ldots\cup L_i)=i$ remain satisfied for every $i\leqq w$ such that $L_i\neq\emptyset$. It is shown in [16] that then $\text{width}(L_i)\leqq 2$ for every $i\leqq w$. The proposition follows by considering a chain decomposition of P which consists of the chain L_1 and $\leqq 2$ chains covering L_i for $2\leqq i\leqq w$. □

Now we give an example where $s_{\mathscr{J}}^{\infty}=r_{\mathscr{J}}^{\infty}$.

Let RESTRICTED BIN PACKING be the bin packing problem restricted to items with sizes $(1/2)-\varepsilon$ and $(1/2)+\varepsilon$ (for some fixed $\varepsilon<1/6$). We denote $(1/2)-\varepsilon$ by a and $(1/2)+\varepsilon$ by b.

Theorem 9. For the restricted bin packing problem $s_{\mathscr{J}}^{\infty}=r_{\mathscr{J}}^{\infty}=4/3$.

Proof. The lower bound $s_{\mathscr{J}}^{\infty}\geqq 4/3$ is noted e.g. in Liang [20]. Consider $I'<I$ where I contains n a-items followed by n b-items and I' is the first half of I. If an algorithm A fills k bins with 2 a-items each after processing I' then

$$|A(I')|/p(I') = 2-2(k/n), \quad |A(I)|/p(I) \geqq 1+(k/n)$$

which implies the bound for $s_{\mathscr{J}}^{\infty}$.

To prove the upper bound we describe an on-line algorithm with $r_{\mathscr{J}}^{\infty}=4/3$.

We distinguish 4 types of bins: a-bins, b-bins, aa-bins and ab-bins, corresponding to the items contained in the bin. The algorithm will also pair some bins, the possible bin-pair types will be (aa, a), (aa, b), and (aa, ab). If a bin is not paired with any other bin it is called unpaired.

A new element is processed according to the following rules:

a) for a new element a^*:
if there is a b-bin B *then* put a^* into B
else if there is an unpaired a-bin B *then* put a^* into B
else if there is an unpaired aa-bin B *then* put a^* into a new bin B' and pair B and B'
else open a new bin B for a^*;

b) for a new element b^*:
if there is an a-bin B *then* put b^* into B
else if there is an unpaired aa-bin B *then* put b^* into a new bin B' and pair B and B'
else open a new bin B for b^*.

If there are several bins satisfying a condition then the choice is arbitrary, for definiteness let us always choose the first one.

It is easy to see that all possible bin-pair types that may be formed by the algorithm are indeed (aa, a), (aa, b) and (aa, ab).

Let us assume thatafter processing a list I the algorithm created c_1 unpaired a-bins, c_2 unpaireb b-dins, c_3 nnpaired aa-bins, c_4 unpaired ab-bins, c_5 (aa, a) bin-pairs, c_6 (aa, b) bin-pairs and c_7 (aa, ab) bin-pairs.

By definition

$$|A(I)| = c_1+c_2+c_3+c_4+2c_5+2c_6+2c_7, \tag{1}$$

as the number of b-items is a lower bound to $p(I)$

$$p(I) \geqq c_2+c_4+c_6+c_7, \tag{2}$$

and as the half of the number of items is a lower bound to $p(I)$

$$p(I) \geqq (1/2)c_1+(1/2)c_2+c_3+c_4+(3/2)c_5+(3/2)c_6+2c_7. \tag{3}$$

Subtracting (2) resp. (3) from (1) we get

$$|A(I)|-p(I) \leqq c_1+c_3+2c_5+c_6+c_7, \tag{4}$$

$$|A(I)|-p(I) \leqq (1/2)c_1+(1/2)c_2+(1/2)c_5+(1/2)c_6. \tag{5}$$

We note that there cannot be both an a-bin and a b-bin in the packing as in this case the item arriving later would not be put into a separate bin.

Lemma 10. $c_1+c_3+c_6 \leqq 1$.

Proof. We consider 6 different cases.

1) There cannot be 2 unpaired a-bins as otherwise the a-item arriving later would not have to be put in a separate bin.

2) There cannot be an unpaired a-bin B and an unpaired aa-bin B'. Indeed, if the a-item in B comes last, then B could be paired with B', if one of the a-items in B' comes last then before the arrival of this element we get a contradiction to 1).

3) There cannot be 2 unpaired aa-bins as otherwise before the arrival of the last item we get a contradiction to 2).

4) There cannot be an unpaired a-bin and an (aa, b) bin-pair by the remark preceding the lemma.

5) There cannot be an unpaired aa-bin and an (aa, b) bin-pair. Again by the remark preceding the lemma the item coming last must be the b-item. But then before the arrival of this item we get a contradiction to 3).

6) There cannot be 2 (aa, b) bin-pairs. Again, the last item arriving must be a b-item. But then before the arrival of this item we get a contradiction to 5). □

In the proof of the theorem we distinguish 2 cases.

Case 1. $c_2=0$.

Then using Lemma 10., (5) and $c_5 \leqq (2/3)p(I)$ following from (3) we get

$$|A(I)|-p(I) \leqq (1/2)c_5+(1/2) \leqq (1/3)p(I)+(1/2)$$

hence

$$|A(I)| \leqq (4/3)p(I)+(1/2).$$

Case 2. $c_2>0$.

From the remark preceding Lemma 10. in this case $c_5=0$ and so we get from (4) and (5) using Lemma 10.

$$|A(I)|-p(I) \leqq 1+c_7 \tag{6}$$

$$|A(I)|-p(I) \leqq (1/2)+(1/2)c_2. \tag{7}$$

Adding (7) twice and (6) and using $c_2+c_7 \leqq p(I)$ (cf. (2))

$$3(|A(I)|-p(I)) \leqq 2+c_2+c_7 \leqq 2+p(I)$$

and so

$$|A(I)| \leqq (4/3)p(I)+(2/3). \quad \square$$

6. Some remarks and problems

1. (Greedy algorithm vs. on-line algorithms.)

The chain decomposition problem for series-parallel orders is an example where the greedy algorithm gives an optimal solution. For the edge coloring problem $r^\infty_{A_{gr}}=2$ and no on-line algorithm can have better performance. Thus for these problems on-line algorithms cannot perform better than the greedy algorithm.

On-line algorithms give a large improvement for the general chain decomposition problem (where $A_{gr}(n)=\Omega(n)$ and there is an on-line algorithm with bounded performance), for the graph coloring problem (where $A_{gr}(n)=\Omega(n)$ and there is an on-line algorithm with $A(n)=o(n)$) and for the bin packing problem (where $r^\infty_{A_{gr}}=1.7$ and there is an on-line algorithm with $r^\infty_A=5/3$).

There appears to be no example known where the greedy algorithm is not optimal but there is an on-line algorithm giving an optimal solution. Also for none of the examples considered does it hold that $r^\infty_{A_{gr}}=\infty$ but there is an on-line algorithm A with $r^\infty_A<\infty$.

2. (Bounds for particular problems.)

It would be interesting to improve the bounds for the performance of on-line algorithms for matroid and graph partitioning, in particular to decide if on-line algorithms can perform better than the greedy algorithm for partitioning graphs.

Concerning adversaries it appears to be not known if adaptive adversaries can lead to stronger lower bounds for the bin packing problem. Another question is the following: is $s^\infty_{\mathcal{G}}=\infty$ for the graph coloring problem? (Coloring optimally with i new colors those initial segments for which the chromatic number is i gives a coloring which uses $\leqq i(i+1)/2$ colors for every initial segment of chromatic number i.)

A related partition problem which does not fit into the class of problems discussed here but which would be interesting to study in the context of on-line algorithms is the m-machine scheduling problem: given n tasks with execution times $t_1, \ldots, t_n$ find a schedule for m machines to minimize finishing time (thus here the number of the classes is fixed and we want to minimize the maximal weight). The greedy algorithm has performance ratio $2-(1/m)$ (Graham [10]). No on-line algorithm appears to be known which improves this for any m. The lists (1, 1, 2) and (1, 1, 1, 3, 3, 3, 6) show that no improvement is possible for $m=2$ and $m=3$. The list (1 m times, $1+\sqrt{2}$ m times, $2(1+\sqrt{2})$ once) shows that $1+(1/\sqrt{2})$ is a lower bound for the performance ratio of on-line algorithms for every $m\geqq 4$.

Acknowledgement. We thank Collette Coullard, János Csirik, Gábor Galambos and László Lovász for their valuable remarks.

References

[1] D. Bean: Effective coloration, J. Symb. Logic 41 (1976), 469—480.
[2] B. Bollobás: Extremal Graph Theory, Academic Press, London, 1976.
[3] A. Borodin, N. Linial, M. Saks: An optimal on-line algorithm for for metrical task systems, 19. STOC (1987), 373—382.
[4] D. J. Brown: A lower bound for on-line one-dimensional bin packing algorithms, Tech. Rep. No. R-864, Coord. Sci. Lab., Univ. of Illinois, Urbana, IL, 1979.
[5] J. Csirik, V. Totik: On-line algorithms for a dual version of bin packing, Discr. Appl. Math. 21 (1988), 163—167.
[6] J. Edmonds: Minimum partition of a matroid into independent subsets, J. Res. Nat. Bur. St. 69B (1965), 67—72.
[7] U. Faigle, Gy. Turán: Notes on on-line algorithms, Proc. Conf. SOR (1988), to appear.
[8] G. Galambos: Parametric lower bound for on-line bin packing, SIAM J. Alg. Disc. Meth. 7 (1986), 362—367.
[9] M. R. Garey, D. S. Johnson: Computers and Intractability: a Guide to the Theory of NP-Completeness, Freeman, New York, 1979.
[10] R. L. Graham: Bounds for certain multiprocessing anomalies, Bell Syst. Tech. J. 45 (1966), 1563—1581.
[11] A. Gyárfás, J. Lehel: On-line and first-fit colorings of graphs, J. Graph Th. 12 (1988), 217—227.
[12] D. S. Johnson: Approximation algorithms for combinatorial problems, J. Comp. Syst. Sci. 9 (1974), 256—278.
[13] D. S. Johnson, A. Demers, J. D. Ullman, M. R. Garey, R. L. Graham: Worst-case performance bouns for simple one-dimensional bin packing algorithms, SIAM J. Comp. 3 (1974), 299—325.
[14] H. A. Kierstead: An effective version of Dilworth's theorem, Trans. AMS 268 (1981), 63—77.
[15] H. A. Kierstead: The linearity of first-fit coloring of interval graphs, SIAM J. Disch. Math. 1 (1988), 526—530.
[16] H. A. Kierstead, W. T. Trotter: An extremal problem in recursive combinatorics, Congr. Numer. 33 (1981), 143—153.
[17] B. Korte, D. Hausmann: An analysis of the greedy heuristic for independence systems, in: Algorithmic Aspects of Combinatorics (B. Alspach, P. Hell, D. J. Miller eds.), Annals of Discr. Math. 2 (1978), 65—74., North-Holland, Amsterdam.
[18] E. L. Lawler: Combinatorial Optimization: Networks and Matroids, Holt, Reinhart and Winston, New York, 1976.
[19] C. C. Lee, D. T. Lee: A simple on-line bin packing algorithm, J. ACM 32 (1985), 562—572.
[20] F. M. Liang: A lower bound for on-line bin packing, Inf. Proc. Lett. 10 (1980), 76—79.
[21] L. Lovász: Covers, packings and some heuristic algorithms, Proc. 5th British Comb. Conf., Util. Math. (1976), 417—429.
[22] L. Lovász: Combinatorial Problems and Exercises, Akadémiai Kiadó, Budapest, and North-Holland, Amsterdam, 1979.
[23] L. Lovász, M. Saks, W. T. Trotter: An on-line graph coloring algorithm with sublinear performance, preprint (1988).
[24] M. S. Manasse, L. A. McGeoch, D. D. Sleator: Competitive algorithms for on-line problems, 20. STOC (1988), 322—333.
[25] M. Szegedy, unpublished (1986).
[26] E. Szemerédi, unpublished (1982).

[27] GY. TURÁN: On the greedy algorithm for an edge-partitioning problem, in: Coll. Math. Soc. J. Bolyai 44. Theory of Algorithms, (L. Lovász, E. Szemerédi eds.), Pécs (Hungary), 1984, 405—423.
[28] W. F. DE LA VEGA, G. S. LUEKER: Bin packing can be solved within $1+\varepsilon$ in linear time, Combinatorica 1 (1981), 349—355.
[29] V. G. VIZING: On the value of the chromatic class of a p-graph, Discr. Anal. 3 (1964), 25—30., Novosibirsk. (In Russian.)
[30] D. J. A. WELSH: Matroid Theory, Academic Press, London, 1976.
[31] A. C. YAO: New algorithms for bin packing, J. ACM 27 (1980), 207—227.

(Received January 5, 1989)

Determination of the structure of the class $\mathscr{A}(R, S)$ of (0, 1)-matrices

A. KUBA

Kalmár Laboratory of Cybernetics József Attila University
Szeged, Árpád tér 2. H-6720 Hungary

Summary

The class $\mathscr{A}(R, S)$ contains the (0, 1)-matrices having row and column sum vectors R and S, respectively. The problem of the structure of $\mathscr{A}(R, S)$ is considered, that is the problem of determining the sets of invariant 1's, invariant 0's and variant positions. Two methods are given, whereby the structure can be determined if an element of $\mathscr{A}(R, S)$ or the vectors R and S are known. Furthermore, a new proof is given to Ryser's theorem constructing the variant and invariant positions of the class $\mathscr{A}$.

1. Definitions

Let A be a (0, 1)-matrix of size n by m. The sum of row i of A is denoted by r_i:

$$r_i = \sum_{j=1}^{m} a_{ij} \quad (i = 1, 2, ..., n),$$

and the sum of column j of A is denoted by s_j:

$$s_j = \sum_{i=1}^{n} a_{ij} \quad (j = 1, 2, ..., m).$$

We call $R=(r_1, r_2, ..., r_n)$ the *row sum vector* and $S=(s_1, s_2, ..., s_m)$ the *column sum vector* of A. R and S are also called the *projections* of A. There is an extensive literature on different questions concerning binary matrices and their projections (for surveis see e.g. [9] and [1]). Let $\mathscr{A}(R, S)$ denote the class of $n\times m$ (0, 1)-matrices with row sum vector R and column sum vector S. Gale [2] and Ryser [6] have proved that the class $\mathscr{A}(R, S)$ is non-empty if and only if

$$\sum_{j=1}^{k} \bar{s}_j \geqq \sum_{j=1}^{k} s_j$$

for all $k=1, 2, \ldots, m$, where $\bar{S}=(\bar{s}_1, \bar{s}_2, \ldots, \bar{s}_m)$ is the column sum vector of binary matrix $\bar{A}$ defined as

$$\bar{A} = \begin{pmatrix} \delta_1 \\ \delta_2 \\ \vdots \\ \delta_n \end{pmatrix},$$

where

$$\delta_i = (1, 1, \ldots, 1, 0, 0, \ldots, 0)$$

with r_i number of 1's and $(m-r_i)$ number of 0's $(0 \leqq r_i \leqq m)$. There is exactly one matrix in $\mathscr{A}(R, S)$ if and only if

$$\sum_{j=1}^{k} \bar{s}_j = \sum_{j=1}^{k} s_j$$

for all $k=1, 2, \ldots, m$ (see e.g. [10]).

Consider the matrices

$$A_1 = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \quad \text{and} \quad A_2 = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}.$$

An *interchange* is a transformation of the elements of A that changes a minor of type A_1 into type A_2 or vica versa and leaves all other elements of A unaltered. We say that the four elements of the minor form a *switching component* in A. The interchange theorem of Ryser [6] says that if A and A' are in $\mathscr{A}(R, S)$, then A is transformable into A' by a finite sequence of interchanges.

Let $A \in \mathscr{A}(R, S)$. A is *ambiguous* (with respect to R and S) if there is a different $A' \in \mathscr{A}(R, S)$ $(A' \neq A)$. In the other case, A is *unambiguous*. It is easy to prove (see e.g. [2]) that A is ambiguous if and only if it has a switching component.

An element $a_{ij}=1$ (or 0) of A is called an *invariant* 1 (or 0) if there is no sequence of interchanges which, when applied to A, replaces it by 0 (or 1). Otherwise, a_{ij} is a *variant* element of A. By the interchange theorem, if a_{ij} is an invariant 1 (or 0) of $A \in \mathscr{A}(R, S)$, then a'_{ij} is also an invariant 1 (or 0) of every $A' \in \mathscr{A}(R, S)$. In this sense, we can speak about the *invariant 1, invariant 0* and *variant (i, j) positions* of the class $\mathscr{A}(R, S)$.

Without loss of generality, we can suppose that

$$r_1 \geqq r_2 \geqq \ldots \geqq r_n > 0 \tag{1.1}$$

and

$$s_1 \geqq s_2 \geqq \ldots \geqq s_m > 0, \tag{1.2}$$

because this situation can be reached by excluding zero rows and zero columns and by permuting rows and columns so that the row-sums and the column-sums are non-increasing. A non-empty class $\mathscr{A}(R, S)$ with R and S satisfying (1.1) and (1.2) is said to be *normalized.*

In the determining of the invariant positions of the normalized class $\mathscr{A}(R, S)$, a useful device is the *structure matrix* [8]. Let A be in the normalized class $\mathscr{A}(R, S)$ and let us write

$$A = \begin{pmatrix} W & X \\ Y & Z \end{pmatrix},$$

where W is of size $e \times f$ $(0 \leqq e \leqq n,\ 0 \leqq f \leqq m)$. Let Q be a (0, 1)-matrix, and let $N_0(Q)$ denote the number of 0's in Q, let $N_1(Q)$ denote the number of 1's in Q. Now let

$$t_{ef} = N_0(W) + N_1(Z)$$

$e=0, 1, \ldots, n$; $f=0, 1, \ldots, m$. We call the $(n+1)\times(m+1)$ matrix

$$T = (t_{ef})$$

the structure matrix of $\mathscr{A}(R, S)$. It is easy to see that

$$t_{ef} = e \cdot f + \sum_{i=e+1}^{n} r_i - \sum_{j=1}^{f} s_j.$$

Ryser proved the following

Theorem 1.1 [7]. The normalized class $\mathscr{A}(R, S)$ is with invariant 1's if and only if the matrices in $\mathscr{A}(R, S)$ are of the form

$$A = \begin{pmatrix} J & * \\ * & O \end{pmatrix}.$$

Here O is a zero matrix and J is a matrix of 1's of size $e \times f$ $(0 < e \leqq n,\ 0 < f \leqq m)$ specified by

$$t_{ef} = 0.$$

(The integers e and f are not necessarily unique, but they are determined by R and S and are independent of the particular choice of A in $\mathscr{A}$.)

By Theorem 1.1, one can construct the structure of class $\mathscr{A}(R, S)$ with the help of matrix T. In this paper, another way is given to construct the invariant and variant positions of class $\mathscr{A}$. First, the structure of the variant elements of the (not necessarily normalized) class $\mathscr{A}$ is given. From the determination of the positions of the variant elements, it is also possible to give the whole structure of $\mathscr{A}$. In Section 3, the case of the normalized class is discussed applying the idea of double-projection used earlier in characterization problems of binary matrices [5]. A direct and demonstrative relation between the structure of $\mathscr{A}$ and the vectors R and S is given in Section 4, from which the mode of construction of the structure of $\mathscr{A}$ follows.

2. The structure of the class $\mathscr{A}(R, S)$

First, consider the variant elements of $\mathscr{A}(R, S)$.

Lemma 2.1. Let A be a matrix in $\mathscr{A}(R, S)$, and let

$$a_{ij_1}, a_{ij_2}, \ldots, a_{ij_k},$$

$$a_{i_1j}, a_{i_2j}, \ldots, a_{i_lj},$$

be variant elements of A such that $1 \leqq i_1, i_2, \ldots, i_l \leqq n$, $1 \leqq j_1, j_2, \ldots, j_k \leqq m$, $i \in \{i_1, i_2, \ldots, i_l\}$, $j \in \{j_1, j_2, \ldots, j_k\}$, where $1 < l \leqq n$ and $1 < k \leqq m$. Then, $a_{i'j'}$ is variant for all $(i', j') \in \{i_1, i_2, \ldots, i_l\} \times \{j_1, j_2, \ldots, j_k\}$ (see Fig. 1/a).

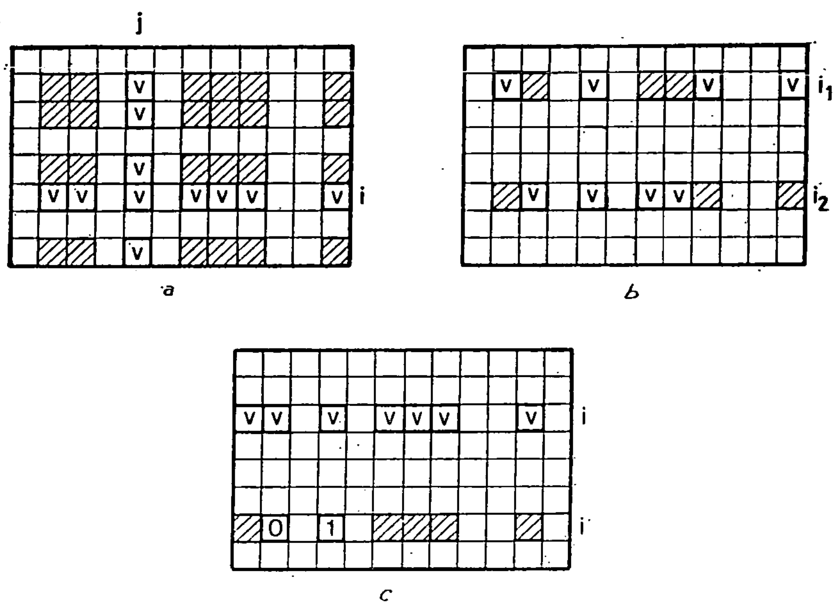


Figure 1. The variant elements ▨ induced by the variant elements ☑ according to *a)* Lemma 2.1, *b)* Lemma 2.2 and *c)* Lemma 2.3

Proof. The assumptions of Lemma 2.1 include that a_{ij} is a variant element of A. Let (i', j') $(i' \neq i, j' \neq j)$ be an otherwise arbitrary element of $\{i_1, i_2, \ldots, i_l\} \times \times \{j_1, j_2, \ldots, j_k\}$. If

$$a_{i'j'} = a_{ij}, \tag{2.1}$$

$$a_{i'j'} = 1 - a_{i'j}, \tag{2.2}$$

$$a_{i'j'} = 1 - a_{ij'}, \tag{2.3}$$

then a_{ij}, $a_{i'j}$, $a_{ij'}$ and $a_{i'j'}$ form a switching component in A, and hence $a_{i'j'}$ is variant. If any of the equalities (2.1)—(2.3) is not satisfied, then, since a_{ij}, $a_{i'j}$ and $a_{ij'}$ are variant, it is possible to alter any of them (occasionally all of them) by a suitable interchange in order to get a switching component at $\{i', i'\}\times\{j, j'\}$. That is, (i', j') is a variant position in $\mathscr{A}(R, S)$. A simple consequence of Lemma 2.1 is the following

Lemma 2.2. Let A be a matrix in $\mathscr{A}(R, S)$, and let

$$J = \{j_1, j_2, \dots, j_k\}, \quad J' = \{j_1', j_2', \dots, j_l'\},$$

$$J \cap J' \neq \emptyset,$$

such that $a_{i_1j_1}, a_{i_1j_2}, \dots, a_{i_1j_k}$ and $a_{i_2j_1'}, a_{i_2j_2'}, \dots, a_{i_2j_l'}$ are variant. Then, a_{ij} is variant for all $(i,j)\in\{i_1, i_2\}\times(J\cup J')$ (see Fig. 1/*b*).

Proof. By Lemma 2.1, the elements of A at $\{i_1, i_2\}\times J$ and $\{i_1, i_2\}\times J'$ are variant.

Lemma 2.3. Let A be a matrix in $\mathscr{A}(R, S)$, and let $J=\{j_1, j_2, \dots, j_k\}$ be the indices of variant elements in row i $(1\leq i\leq n)$. If there is a row i' $(i'\neq i)$ such that the elements $a_{i'j_1}, a_{i'j_2}, \dots, a_{i'j_k}$ also include 0 and 1, then $a_{i'j_1}, a_{i'j_2}, \dots, a_{i'j_k}$ are variant elements (see Fig. 1/c).

Proof. Let us suppose that $a_{i'j_1}=0$ and $a_{i'j_2}=1$ (by a suitable rewriting of the indices, we can always reach such a situation). We shall construct a switching component at $\{i, i'\}\times\{j_1, j_2\}$: If $a_{ij_1}=1$ and $a_{ij_2}=0$, then we are ready. If $a_{ij_1}=1$ and $a_{ij_2}=1$, then, since a_{ij_2} is variant, there is a switching component whereby a_{ij_2} will be 0 (a_{ij_1} and $a_{i'j_2}$ remain unchanged). Similarly, if $a_{ij_1}=0$ and $a_{ij_2}=0$, then there is a switching component whereby a_{ij_1} will be 1 (in this case a_{ij_2} and $a_{i'j_1}$ remain unchanged). In the last case, if $a_{ij_1}=0$ and $a_{ij_2}=1$, then we can change a_{ij_1} and a_{ij_2} by at most two interchanges (without changing $a_{i'j_1}$ and $a_{i'j_2}$).

Theorem 2.1. The variant positions of class $\mathscr{A}(R, S)$, if there are any, are in sets $T_1, T_2, \dots, T_p$ $(p=0$ is also possible) such that

$$T_s = I_s\times J_s,$$

$s=1, 2, \dots, p$, where I_s are pairwise disjunct subsets of $\{1, 2, \dots, n\}$ and J_s are pairwise disjunct subsets of $\{1, 2, \dots, m\}$.

Proof. Consider the set of column indices of the variant elements in row i, denoted by J_i. Let

$$I_i = \{l \mid J_l\cap J_i \neq \emptyset\},$$

and let

$$\bar{J}_i = \bigcup_{l\in I_i} J_l.$$

By Lemma 2.2, every position (i,j) is variant for which $(i,j)\in I_i\times\bar{J}_i$. By definition, it is clear that (i,j), $(i',j')\in I_i\times\bar{J}_i$ if and only if

$$I_i\times\bar{J}_i = I_{i'}\times\bar{J}_{i'}.$$

That is, by applying the procedure for all $i=1, 2, \ldots, n$, we get disjoint subsets $I_1, I_2, \ldots, I_p$ and $J_1, J_2, \ldots, J_p$, and the sets

$$T_s = I_s \times J_s,$$

$s=1, 2, \ldots, p$, contain all of the variant positions of $\mathcal{A}(R, S)$.

3. The structure of the normalized class $\mathcal{A}(R, S)$

Henceforth, we take $\mathcal{A}(R, S)$ normalized.

Lemma 3.1. Let A be a binary matrix in the normalized class $\mathcal{A}(R, S)$, and let

$$u_i = \begin{cases} \max\{j \mid a_{ij} = 1\}, & \text{if} \quad a_{ij} = 1 \quad \text{for some} \quad j = 1, 2, \ldots, m \\ 0, & \text{if} \quad a_{ij} = 0 \quad \text{for all} \quad j = 1, 2, \ldots, m \end{cases}$$

and

$$z_i = \begin{cases} \min\{j \mid a_{ij} = 0\}, & \text{if} \quad a_{ij} = 0 \quad \text{for some} \quad j = 1, 2, \ldots, m \\ m+1, & \text{if} \quad a_{ij} = 1 \quad \text{for all} \quad j = 1, 2, \ldots, m \end{cases}$$

for all $i=1, 2, \ldots, n$. If $z_i < u_i$ for some i, then a_{ij} is variant for all j, $z_i \leqq j \leqq u_i$.

Proof. If there is an i, $1 \leqq i \leqq n$, such that $a_{ij}=0$, $a_{ij'}=1$, $j \leqq j'$, then, since $s_j \geqq s_{j'}$, there is an i', $1 \leqq i' \leqq n$, such that $a_{i'j}=1$, $a_{i'j'}=0$. That is, $a_{ij}, a_{ij'}, a_{i'j}$ and $a_{i'j'}$ form a switching component. Therefore, all of the positions between z_i and u_i are variant.

An analogous lemma is true for the columns:

Lemma 3.2. Let A be a (0, 1)-matrix in the normalized class $\mathcal{A}(R, S)$, and let

$$v_j = \begin{cases} \max\{i \mid a_{ij} = 1\}, & \text{if} \quad a_{ij} = 1 \quad \text{for some} \quad i = 1, 2, \ldots, n \\ 0, & \text{if} \quad a_{ij} = 0 \quad \text{for all} \quad i = 1, 2, \ldots, n \end{cases}$$

and

$$w_j = \begin{cases} \min\{i \mid a_{ij} = 0\}, & \text{if} \quad a_{ij} = 0 \quad \text{for some} \quad i = 1, 2, \ldots, n \\ n+1, & \text{if} \quad a_{ij} = 1 \quad \text{for all} \quad i = 1, 2, \ldots, n \end{cases}$$

for all $j=1, 2, \ldots, m$. If $w_j < v_j$ for some j, then a_{ij} is variant for all i, $w_j \leqq i \leqq v_j$.

Theorem 3.1. The variant positions of the normalized class $\mathcal{A}(R, S)$ are in the sets $T_1, T_2, \ldots, T_p$ ($p=0$ is also possible) such that

$$T_s = I_s \times J_s,$$

$s=1, 2, \ldots, p$, where

$$I_s = \{i'_s, i'_s+1, \ldots, i''_s\}, \quad 1 \leqq i'_1 < i''_1 < i'_2 < i''_2 \ldots < i'_p < i''_p \leqq n,$$

$$J_s = \{j'_s, j'_s+1, \ldots, j''_s\}, \quad 1 \leqq j'_p < j''_p < j'_{p-1} < j''_{p-1} < \ldots < j'_1 < j''_1 \leqq m.$$

Proof. We know that the variant elements of $\mathcal{A}(R, S)$, which are recognized by Lemmas 3.1 and 3.2, follow in rows and in columns consecutively. Following the same idea as in the Proof of Theorem 2.1, we have that the sets $T_s = I_s \times J_s$, $s=1, 2, \ldots, p$, are the places of variant elements, where I_s and J_s contain the indices of consecutive rows and columns, respectively. Furthermore, $I_s \cap I_{s'} = \emptyset$ and $J_s \cap J_{s'} = \emptyset$ if $s \neq s'$. From this construction, it is clear that $(\{1, 2, ..., i'_s - 1\} \times J_s) \cup (I_s \times \{1, 2, ..., j'_s - 1\})$ contains only 1's and $(\{i''_s + 1, i''_s + 2, ..., m\} \times J_s) \cup (I_s \times \{j''_s + 1, j''_s + 2, ..., n\})$ contains only 0's. Since the elements of R and S are in decreasing order, $i''_r < i'_s$, $1 \leqq r, s \leqq p$, if and only if $j'_r > j''_s$. That is, if $T_1, T_2, ..., T_p$ are indexed so that

$$1 \leqq i'_1 < i''_1 < i'_2 < i''_2 < ... < i'_p < i''_p \leqq n,$$

then

$$1 \leqq j'_p < j''_p < j'_{p-1} < j''_{p-1} < ... < j'_1 < j''_1 \leqq m.$$

It is easy to see that the set $\{1, 2, ..., n\} \times \{1, 2, ..., m\} \setminus \cup_s T_s$ contains only invariant positions and so $\cup_s T_s$ is the set of the variant positions of the normalized class $\mathcal{A}(R, S)$.

The following algorithm can be used to determine the sets of the indices of the variant elements, $I_s = \{i'_s, i'_s + 1, ..., i''_s\}$ and $J_s = \{j'_s, j'_s + 1, ..., j''_s\}$:

Step 1: First, the indices z_i and u_i are computed for each row i. It is clear that $z_i \leqq u_i + 1$ $(1 \leqq i \leqq n)$.

Step 2: The sequence of indices u_i is modified taking the rows from down to up such that if $u_{i+1} > u_i$ then let $u_i = u_{i+1}$ $(n-1 \geqq i > 1)$.

Step 3: The rows are scanned one by one from $i=1$ to $i>n$ with an initial value $s=0$. If $z_i > u_i$ then there is no variant element in the row i. In the other case, i.e. if $z_i \leqq u_i$, then there are variant elements in this row and let $s = s+1$, $i'_s = i$, $j'_s = z_i$ (initially) and $j''_s = u_i$. The indices j'_s and i''_s can be determined by scanning the rows further while $j'_s \leqq u_i$ such that meanwhile if $j'_s > z_i$ then let $j'_s = z_i$. In the row, where $j'_s > u_i$, let $i''_s = i-1$ (this condition will be satisfied at least once if we set $u_{n+1} = z_{n+1} = -1$ at the beginning of the procedure).

Let us see two examples:

Example 3.1. Let the (0, 1)-matrix A be defined as

$$a_{ij} = \begin{cases} 1, & \text{if } i = j \\ 0, & \text{if } i \neq j \end{cases}$$

$i = 1, 2, ..., n$, $j = 1, 2, ..., n$. In this case $u_i = i$, $z_i = 1$, $i = 1, 2, ..., n$ (with the exception that $z_1 = 2$). Applying the algorithm, we get that the set T_1 containing the indices of the variant elements is

$$T_1 = \{1, 2, ..., n\} \times \{1, 2, ..., n\},$$

that is the whole matrix.

Example 3.2. Let A be given by Figure 2. Then

$$u_1 = 13, \quad z_1 = 14, \quad u_2 = 11, \quad z_2 = 12, \quad u_3 = 10, \quad z_3 = 11,$$

$$u_4 = 10, \quad z_4 = 11, \quad u_5 = 11, \quad z_5 = 9, \quad u_6 = 7, \quad z_6 = 8,$$

$$u_7 = 5, \quad z_7 = 3, \quad u_8 = 6, \quad z_8 = 4, \quad u_9 = 1, \quad z_9 = 2,$$

$$i'_1 = 3, \quad i''_1 = 5, \quad j'_1 = 9, \quad j''_1 = 11$$

13	1	1	1	1	1	1	1	1	1	1	1	1	1
11	1	1	1	1	1	1	1	1	1	1	1	0	0
10	1	1	1	1	1	1	1	1	1	1	0	0	0
10	1	1	1	1	1	1	1	1	1	1	0	0	0
9	1	1	1	1	1	1	1	1	0	0	1	0	0
7	1	1	1	1	1	1	1	0	0	0	0	0	0
4	1	1	0	1	1	0	0	0	0	0	0	0	0
4	1	1	1	0	0	1	0	0	0	0	0	0	0
1	1	0	0	0	0	0	0	0	0	0	0	0	0
	9	8	7	7	7	7	6	5	4	4	3	1	1

Figure 2. The structure of the normalized class $\mathscr{A}(R, S)$ of Example 3.2

and

$$i_2' = 7, \quad i_2'' = 8, \quad j_2' = 3, \quad j_2'' = 6.$$

That is,

$$T_1 = \{3, 4, 5\}\times\{9, 10, 11\}, \quad T_2 = \{7, 8\}\times\{3, 4, 5, 6\}.$$

4. Determination of the structure of class $\mathscr{A}(R, S)$ from the projections

Consider the matrices $A^{(x)}$ and $A^{(y)}$ defined by R and S as

$$a_{ij}^{(x)} = \begin{cases} 0, & \text{if } \ j > r_i; \\ 1, & \text{otherwise,} \end{cases}$$

and (4.1)

$$a_{ij}^{(y)} = \begin{cases} 0, & \text{if } \ i > s_j; \\ 1, & \text{otherwise,} \end{cases}$$

$i=1, 2, \ldots, n$, $j=1, 2, \ldots, m$ (see [5]). The projections of $A^{(x)}$ are $(R^{(x)}, S^{(x)})$, where $R^{(x)}=R$. The projections of $A^{(y)}$ are $(R^{(y)}, S^{(y)})$, where $S^{(y)}=S$. Similarly, the matrices $A^{(xy)}$ and $A^{(yx)}$ are defined by $S^{(x)}$ and $R^{(y)}$ as

$$a_{ij}^{(xy)} = \begin{cases} 0, & \text{if } \ i > s_j^{(x)}; \\ 1, & \text{otherwise,} \end{cases}$$

and (4.2)

$$a_{ij}^{(yx)} = \begin{cases} 0, & \text{if } \ j > r_i^{(y)}; \\ 1, & \text{otherwise} \end{cases}$$

$i=1, 2, \ldots, n$, $j=1, 2, \ldots, m$. The projections of $A^{(xy)}$ and $A^{(yx)}$ are denoted by $(R^{(xy)}, S^{(xy)})$, and $(R^{(yx)}, S^{(yx)})$, respectively. It is easy to see that $A^{(xy)}$ and $A^{(yx)}$ are unambiguous (they have no switching component). From the construction, it follows that $R^{(xy)}$ consists of the elements of R in decreasing order and $S^{(yx)}$ consists of the elements of S in decreasing order. That is, by constructing a (0, 1)-matrix B with projections $(R^{(xy)}, S^{(yx)})$ and making a suitable permutation of its rows and columns, we get a binary matrix of $\mathscr{A}(R, S)$.

If $A^{(xy)}=A^{(yx)}$, then let $B=A^{(xy)}(=A^{(yx)})$. As B is uniquely determined by its projections, it has no variant element, and so there is no variant element of A that can be constructed from B by suitable row and column permutations.

If $A^{(xy)}\neq A^{(yx)}$, then from matrix $A^{(xy)}$ the matrix B can be constructed by successively shifting the 1's from the left to the right in the rows of $A^{(xy)}$, similarly as in [10]:

Procedure to construct (0, 1)-matrix B:

Step 1: $j:=1$, $B:=A^{(xy)}$.

Step 2: Consider the jth column of B. If the number of 1's in this column is greater than $s_j^{(yx)}$, then find the first row, begin from the bottom position upward, which contains a 1 in the jth column and a 0 nearest to the right. Interchange the 1 and the 0 in B. Repeat in this fashion until only $s_j^{(yx)}$ 1's are left in this column.

Step 3: $j:=j+1$. If $j=m$, stop. Otherwise, go to Step 2.

The result of this Procedure is a (0, 1)-matrix B having row and column projections $R^{(xy)}$ and $S^{(yx)}$, respectively.

If $A^{(xy)}\neq A^{(yx)}$, then $S^{(xy)}\neq S^{(yx)}$, but even in this case

$$\sum_{j=1}^{k} s_j^{(xy)} \geqq \sum_{j=1}^{k} s_j^{(yx)}$$

for all k, $1\leqq k\leqq m$, so that there is inequality for at least one k. Let $1\leqq j_p'<j_p''<<j_{p-1}'<j_{p-1}''<\ldots<j_1'<j_1''\leqq m$ $(p\geqq 1)$ be the column indices such that

$$\sum_{j=1}^{k} s_j^{(xy)} > \sum_{j=1}^{k} s_j^{(yx)} \tag{4.3}$$

if $j_s'\leqq k<j_s''$ for all $s=1, 2, \ldots, p$, and

$$\sum_{j=1}^{k} s_j^{(xy)} = \sum_{j=1}^{k} s_j^{(yx)}$$

otherwise. It is easy to see that during the Procedure only the jth columns of B can be modified, where $j_s'\leqq j\leqq j_s''$. It is also clear that, if $a_{i_s''j_s'}^{(xy)}=1$ was the bottom 1 in the j_s'th column, then finally it will be in the j_s'' column of B: $b_{i_s''j_s'}=0$ and $b_{i_s''j_s''}=1$. Applying Lemma 3.1, we have $u_{i_s''}=j_s''$ and $z_{i_s''}=j_s'$. Hence, the elements of

$$T_s = I_s\times J_s$$

are invariant, where

$$I_s = \{i_s', i_s'+1, \ldots, i_s''\}$$

and

$$J_s = \{j_s', j_s'+1, \ldots, j_s''\}.$$

During the Procedure, the column j is unaltered if $j_s'\leqq j\leqq j_s''$ is not satisfied for any j_s' and j_s'', $1\leqq s\leqq p$. These columns of B are the same as these columns of $A^{(xy)}$. Therefore, all of the variant elements of $\mathcal{A}(R^{(xy)}, S^{(yx)})$ are in the columns j, where

$$j_s' \leqq j \leqq j_s''$$

for an s, $1 \leqq s \leqq p$. From the definition of $A^{(xy)}$, it follows that

$$w_{j''_s} = s^{(xy)}_{j''_s} + 1 = i'_s$$

and (4.4)

$$v_{j''_s} = s^{(xy)}_{j'_s} = i''_s$$

where $w_{j''_s}$ and $v_{j''_s}$ are defined for the class $\mathscr{A}(R^{(xy)}, S^{(yx)})$, as in Lemma 3.2. An analogous procedure and philosophy for the rows gives that all of the variant elements of $\mathscr{A}(R^{(xy)}, S^{(yx)})$ are in the rows i, where

$$i'_s \leqq i \leqq i''_s,$$

$s=1, 2, \ldots, p$, where $1 \leqq i'_1 < i''_1 < i'_2 < i''_2 < \ldots < i'_p < i''_p \leqq n$ $(p \geqq 1)$ are the row indices such that

$$\sum_{i=1}^{k} r_i^{(yx)} > \sum_{i=1}^{k} r_i^{(xy)} \tag{4.5}$$

if $i'_s \leqq k < i''_s$, $s=1, 2, \ldots, p$, and

$$\sum_{i=1}^{k} r_i^{(yx)} = \sum_{i=1}^{k} r_i^{(xy)}$$

otherwise. That is, from the projections $S^{(xy)}$ and $S^{(yx)}$ we can give the sets of the variant elements of B, T_s, $s=1, 2, \ldots, p$, by (4.3) and (4.4) (or equivalently by (4.3) and (4.5)) explicitly, as they are described in Theorem 3.1.

Let π_x denote a permutation of $S^{(yx)}$ such that $\pi_x(S^{(yx)})=S$, and let π_y denote a permutation of $R^{(xy)}$ such that $\pi_y(R^{(xy)})=R$. Let

$$\pi_y(I_s) = \{\pi_y(i'_s), \pi_y(i'_s+1), \ldots, \pi_y(i''_s)\}$$

and

$$\pi_x(J_s) = \{\pi_x(j'_s), \pi_x(j'_s+1), \ldots, \pi_x(j''_s)\}.$$

Since the sets $T_s = I_s \times J_s$, $s=1, 2, \ldots, p$, contain the indices of the variant elements of $\mathscr{A}(R^{(xy)}, S^{(yx)})$, the sets

$$\pi(T_s) = \pi_y(I_s) \times \pi_x(J_s), \tag{4.6}$$

$s=1, 2, \ldots, p$, contain the indices of the variant elements of the class $\mathscr{A}(R, S)$.

Theorem 4.1. The variant elements of the class $\mathscr{A}(R, S)$, if there are any, are in the sets $\pi(T_s)$, $s=1, 2, \ldots, p$ $(p=0$ is also possible), defined by (4.1)—(4.6).

Let us see two examples.

Example 4.1. Let $R=(1, 1, \ldots, 1)$ and $S=(1, 1, \ldots, 1)$. Then

$$S^{(xy)} = (n, 0, 0, \ldots, 0), \quad S^{(yx)} = (1, 1, \ldots, 1), \quad j'_1 = 1, \quad j''_1 = n, \quad i'_1 = 1, \quad i''_1 = 1,$$

$$p = 1, \quad I_1 = \{1, 2, \ldots, n\}, \quad J_1 = \{1, 2, \ldots, n\}, \quad \pi = (1, 2, \ldots, n),$$

$$\pi_y(I_1) = \{1, 2, \ldots, n\}, \quad \pi_x(J_1) = \{1, 2, \ldots, n\}, \quad \pi(T_1) = \{1, 2, \ldots, n\} \times \{1, 2, \ldots, n\}.$$

Example 4.2 (see Figure 3).

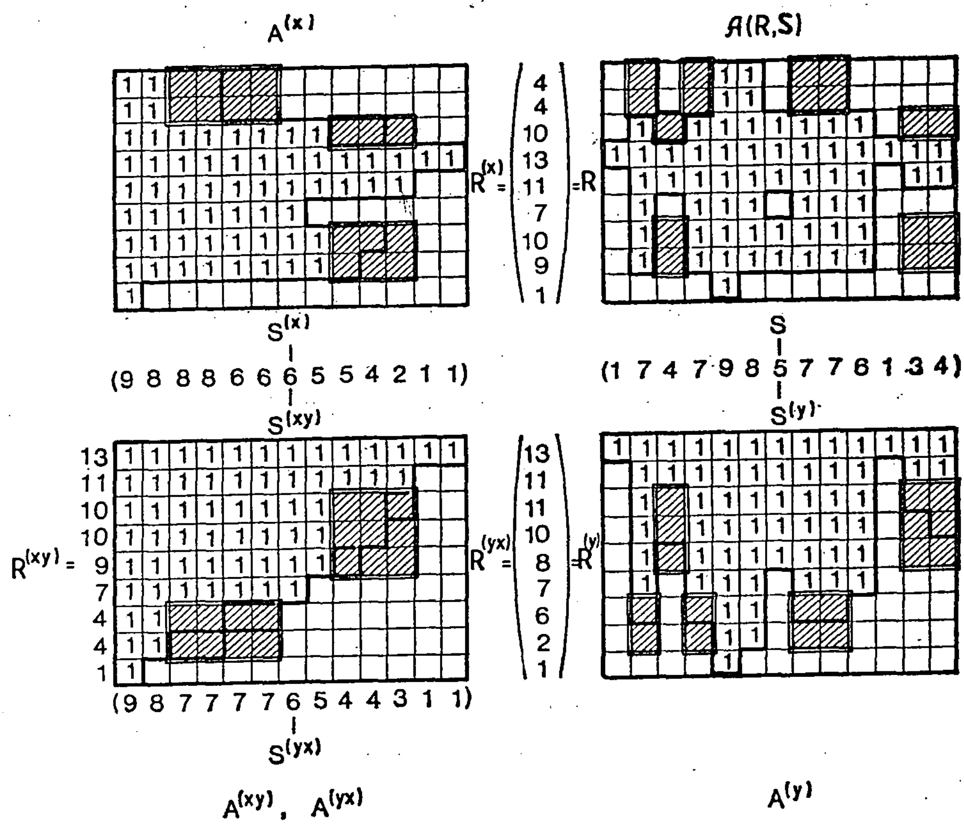


Figure 3. Determination of the structure of $\mathscr{A}(R, S)$ from the projections R and S, □ and ▨ denote the invariant 0's and the variant positions, respectively

Consequence 4.1. The (i, j) elements, $i=1, 2, \ldots, n$, $j=1, 2, \ldots, m$, can be divided into three sets: the positions of invariant 0's, invariant 1's and variant elements. From the construction of $T_1, T_2, \ldots, T_p$ from $R^{(xy)}$ and $S^{(yx)}$, it follows that the set of invariant 1's of the class $\mathscr{A}(R^{(xy)}, S^{(yx)})$ is

$$\{(i, j) | a_{ij}^{(xy)} = 1\} \setminus \bigcup_{s=1}^{p} T_s;$$

the set of variant elements of the class $\mathscr{A}(R^{(xy)}, S^{(yx)})$ is

$$\bigcup_{s=1}^{p} T_s;$$

the set of invariant 0's of the class $\mathscr{A}(R^{(xy)}, S^{(yx)})$ is

$$\{1, 2, \ldots, n\} \times \{1, 2, \ldots, m\} \setminus \{(i, j) | a_{ij}^{(xy)} = 1\} \setminus \Big(\bigcup_{s=1}^{p} T_s\Big).$$

Similarly, the set of invariant 1's of the class $\mathcal{A}(R, S)$ is

$$\{(i, j)|a_{ij} = 1\}\setminus\bigcup_{s=1}^{p} \pi(T_s);$$

the set of variant elements of the class $\mathcal{A}(R, S)$ is

$$\bigcup_{s=1}^{p} \pi(T_s);$$

the set of invariant 0's of the class $\mathcal{A}(R, S)$ is

$$\{1, 2, ..., n\}\times\{1, 2, ..., m\}\setminus\{(i, j)|a_{ij} = 1\}\setminus(\bigcup_{s=1}^{p} \pi(T_s)),$$

where A is an arbitrary element of the class $\mathcal{A}(R, S)$.

Consequence 4.2. From Ryser's Theorem [6], we know that if $A, A'\in\mathcal{A}(R, S)$, then A is transformable into A' by a finite sequence of interchanges. From the structure of $\mathcal{A}(R, S)$ given by Theorem 4.1, it is also clear that the four elements of an interchange are in one of the sets $\pi(T_s)$. That is, if $A, A'\in\mathcal{A}(R, S)$, then A is transformable into A' by a finite sequence of separate interchanges in $\pi(T_1)$, $\pi(T_2)$, ..., $\pi(T_p)$. Let n_s denote the number of different binary matrices generated from an $A\in\mathcal{A}(R, S)$ by interchanges only in $\pi(T_s)$, $s=1, 2, ..., p$. The number of elements of $\mathcal{A}(R, S)$ is an interesting unsolved problem (see [4] and [11]), which can be reduced to the determination of the numbers n_s, $s=1, 2, ..., p$, in the following way:

$$|\mathcal{A}(R, S)| = \prod_{s=1}^{p} n_s.$$

The author thanks Mrs. S. Silóczki and Mrs. E. Vida for the technical assistance in the preparing of the manuscript.

References

[1] R. A. Brualdi: Matrices of zeros and ones with fixed row and column sum vectors Linear Algebra and Its Applications 33, 159—231 (1980).
[2] S.-K. Chang: The reconstruction of binary patterns from their projections Comm. ACM 14, 21—25 (1971).
[3] D. Gale: A theorem on flows in networks Pacific J. Math. 7, 1073—1082 (1957).
[4] W. Honghui: Structure and cardinality of the class $\mathcal{A}(R, S)$ of (0, 1)-matrices J. Math. Res. Exposition 4, 87—93 (1984).
[5] A. Kuba: On the reconstruction of binary matrices Pubbl. Ist. Anal. Globale Apply., Serie „Problemi non ben posti ed inversi", No. 28. Firenze (1986).
[6] H. J. Ryser: Combinatorical properties of matrices of zeros and ones Canad. J. Math. 9, 371—377 (1957).
[7] H. J. Ryser: The term rank of a matrix Canad. J. Math. 10, 57—65 (1958).
[8] H. J. Ryser: Traces of matrices of zeros and ones Canad. J. Math. 12, 463—476 (1960).
[9] H. J. Ryser: Matrices of zeros and ones Bull. Amer. Math. Soc 66, 442—464 (1960).
[10] Y. R. Wang: Characterization of binary patterns and their projections IEEE Trans. Computers C-24, 1032—1035 (1975).
[11] W.-D. Wei: The class $\mathcal{A}(R, S)$ of (0, 1)-matrices Discrete Mathematics 39, 301—305 (1982).

(Received February 6, 1989)

Parallel programming structures and attribute grammars*

R. ALVAREZ GIL and Á. MAKAY

Kalmár Laboratory of Cybernetics, Áprád tér 2, H-6720 Szeged, Hungary

1. Introduction

The attribute grammars are useful tools to give the semantics of programming languages for compiler construction, thus many complier generators based on attribute grammars have been developed [4] [7] [8] [9] [11] [12].

Many papers deal with attribute grammars describing structure of sequential languages for compiler construction, but only a few deals with parallel programming structures.

In this paper we give the semantics of the bracket pair **cobegin-coend** and the symbol **and** in words, and afterwards we give the object which the parallel programming constructions will be translated to. In section 3 we give an attribute grammar able to perform the required translation. The concept of attribute grammars and the notations used can be seen in [1]. In section 4 we mention some experiences got in the implementation by means of attribute grammars of a parallel programming language in which processes communicate through Hoare's monitors.

The methods given in the paper were tested successfully in a CDC 3300 computer of the Hungarian Academy of Sciences with the help of the HLP/SZ compiler generator system [11].

* Supported by the Research Foundation of Hungary, Grant No. 1066, 1143.

2. Semantics and translation of the bracket pair cobegin-coend and the constructor and

The constructor **and** is used to separate instructions such as the symbol;, but the instructions separated by **and** may be executed in parallel. The priority of the symbol **and** is higher than the priority of the symbol;. Thus in the following part of a program: statement$_1$; statement$_2$ **and** statement$_3$; statement$_4$, statement$_2$ and statement$_3$ are executed parallel, but after finishing the execution of statement$_1$ and before beginning the execution of statement$_4$.

The bracket pair **cobegin-coend** is used to ecnlose a statement list such as the bracket pair **begin-end**, but the statement of a statement list enclosed in a bracket pair **cobegin-coend** are executed parallel. To separate the statements enclosed in **cobegin-coend**'s can be used; as well as **and** or mixing the two symbols.

For the translation of these parallel programming constructions we will use three primitives: **fork**, **join** and **quit** [2] [3]. We have selected these primitives because the operations **fork** and **quit** are available in all languages including the psosibility to creating and terminating processes, while **join** can be realized by a "go to" statement and a semafor.

Execution of the operation **fork** w creates a new process starting at the statement labelled w. If a process executes a primitive **join** t, w it is equivalent with $t:=t-1$; **if** $t=\emptyset$ **then goto** w as a unique and indivisable operation.

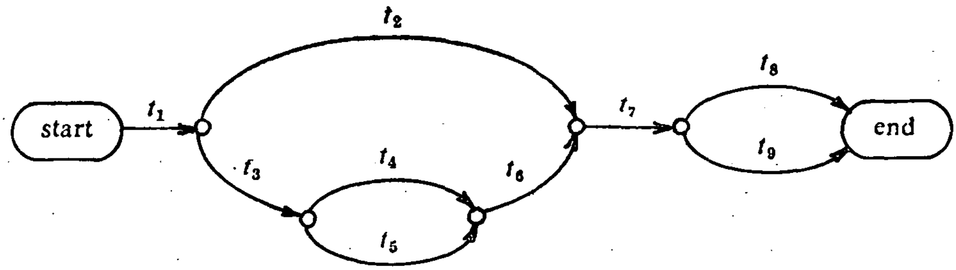
To determine the tasks statically we have to decompose the program into segments representing processes or parts of processes. Of course, processes are not uniquily determined. For example let's see the following program:

begin statement$_1$; statement$_2$; statement$_3$;
    cobegin begin statement$_4$; statement$_5$ **end**;
        begin statement$_6$; statement$_7$ **and** statement$_8$;
        statement$_9$; statement$_{10}$ **end**
    coend;
        statement$_{11}$; statement$_{12}$;
        begin statement$_{13}$; statement$_{14}$ **end and** statement$_{15}$
end

This program can be partitioned into the following segments:

t_1: statement$_1$; statement$_2$; statement$_3$
t_2: statement$_4$; statement$_5$
t_3: statement$_6$
t_4: statement$_7$
t_5: statement$_8$
t_6: statement$_9$; statement$_{10}$
t_7: statement$_{11}$; statement$_{12}$
t_8: statement$_{13}$; statement$_{14}$
t_9: statement$_{15}$.

Moreover we can associate to the program a task flow graph [10] in which each edge corresponds to the execution of a segment:



One of the possibilities to translate our program partitioned into those segments with the above primitives is the following:

begin t_1: statement$_1$; statement$_2$; statement$_3$; n_1:=2; **fork** t_2; **quit**;
t_2: **fork** t_3; statement$_4$; statement$_5$; **join** n_1, t_7; **quit**;
t_3: statement$_6$; n_3:=2; **fork** t_4; **quit**;
t_4: **fork** t_5; statement$_7$; **join** n_3, t_6; **quit**;
t_5: statement$_8$; **join** n_3, t_6; **quit**;
t_6: statement$_9$; statement$_{10}$; **join** n_1, t_7; **quit**;
t_7: statement$_{11}$; statement$_{12}$; n_7:=2; **fork** t_8; **quit**;
t_8: **fork** t_9; statement$_{13}$; statement$_{14}$; **join** n_7, end; **quit**;
t_9: statement$_{15}$; **join** n_7, end; **quit**;
end: **end**

An attribute grammar is able to define that kind of decomposition into segments and of translation to processes.

3. An attribute grammar to describe parallel programming structures for compiler construction

The translation of a structure **cobegin** statement$_1$; ...; statement$_n$ **coend** or statement$_1$ **and** ... **and** statement$_n$ will be as follows:

free m; t:=n; **fork** s_1; **quit**;
s_1: **fork** s_2; **occ** m_1, m_1'; ... code of statement$_1$...; **free** m_1;
join t, end; **quit**;
⋮
s_n: **nop**; **occ** m_n, m_n'; ... code of statement$_n$ ...; **free** m_n;
join t, end; **quit**;
end: **occ** m, m';

where **free** an **occ** are newly introduced macros to allocate and deallocate work-areas for processes.

We use the well known attributes "codelength" (synthesized) and "codeloc" (inherited) which give the length and the localization of the generated code. Another synthesized attribute is "level" to calculate the size of the work-area necessary for each process.

The code generation of parallel structures can be performed at the root of the subtree associated with them in the derivation tree after the generation of the code of each segment (statement$_1$, ..., statement$_n$). For the generation of the correct primitives and macros it is enough to know the size of the work-area and the localization of each statement, because the localization of a statement can be used to obtain the label of the work-area of the statement.

We use some other attributes in the grammars. The synthesized attribute "csloc" gives the necessary information (the localization of the generated code and the size of the work-area of each statement) upwards to the root of the subtree. The inherited attribute "loclev" is a pair (m, m') giving the label and the size of the work-area which has to be allocated at the beginnig and has to be deallocated at the end of the execution of a parallel structure. The inherited attribute "costat" tells us whether a statement is in a parallel structure or not.

The code generation can be performed by a synthesized attribute which is to be evaluated during the last pass. We do not deal with it, because it would have a long and trivial description in the 4-th, 7-th and 8-th syntactical rules of the attribute grammar. Furthermore in a syntactical rule $p: X_0::=X_1 \dots X_{n_p}$ we will omit the semantical rules of the form $X_0.a=X_j.a$ $(1 \leqq j \leqq n_p)$ when there is no other X_i $(1 \leqq i \leqq \leqq n_p$ and $i \neq j)$ which has the same attribute "a", and also the rules of the form $X_j.a=X_0.a$ $(1 \leqq j \leqq n_p)$.

Now see the attribute grammar:

Nonterminal symbols and their attributes:

program has no attributes
block has codelength, level, codeloc, loclev
coblock has codelength, codeloc, loclev
stat_list has codelength, level, csloc, codeloc, loclev, soctat
statement has codelength, level, codeloc, loclev, costat
partstat_list has codelength, csloc, codeloc

Syntactical rules with their semantical rules:

i) program::= block
    block.codeloc=2
    block.loclev=(1, block.level)

ii) program::= coblock
    coblock.codeloc=1
    coblock.loclev=(0, 0)

iii) block::= **begin** stat_list **end**
    stat_list.costat=false

iv) coblock::= **cobegin** stat_list **coend**
    coblock.codelength=stat_list.codelength+5
    stat_list.codeloc=coblock.codeloc+4
    stat_list.loclev=(0, 0)
    stat_list.costat=true

v) stat_list$_1$::= statement; stat_list$_2$
    stat_list$_1$.codelength=statement.codelength+stat_list$_2$.codelength

$$\text{stat\_list}_1.\text{level}=\begin{cases}\text{statement.level, if statement. level} \geqq \text{stat\_list}_2.\text{level}\\ \text{stat\_list}_2.\text{level, if statement. level} < \text{stat\_list}_2.\text{level}\end{cases}$$

    stat_list$_1$.csloc=((statement.codeloc, statement. level), (a_1, b_1), ..., (a_k, b_k)), where $((a_1, b_1, \dots, (a_k, b_k))$=stat_list$_2$.csloc
    stat_list$_2$.codeloc=stat_list$_1$.codeloc+statement.codelength

vi) stat_list::=statement
stat_list.csloc=((statement.codeloc, statement.level))

vii) $stat\_list_1$::=parstat_list; $stat\_list_2$

$$stat\_list_1.codelength=\begin{cases} parstat\_list.codelength+stat\_list_2.codelength+5, \\ \quad \text{if } stat\_list_1.costat=false \\ parstat\_list.codelength+stat\_list_2.codelength, \\ \quad \text{if } stat\_list_1.costat=true \end{cases}$$

$stat\_list_1.csloc=((a_1, b_1), \ldots, (a_k, b_k), (c_1, d_1), \ldots, (c_l, d_l))$, where
$((a_1, b_1), \ldots, (a_k, b_k))=parstat\_list.csloc$ and
$((c_1, d_1), \ldots, (c_l, d_l))=stat\_list_2.csloc$

$$parstat\_list.codeloc=\begin{cases} stat\_list_1.codeloc+4, \text{ if} \\ \quad stat\_list_1.costat=false \\ stat\_list_1.codeloc, \text{ if} \\ \quad stat\_list_1.costat=true \end{cases}$$

$$stat\_list_2.codeloc=\begin{cases} stat\_list_1.codeloc+ \\ parstat\_list.codelength+5, \text{ if} \\ \quad stat\_list_1.costat=false \\ stat\_list_1.codeloc+ \\ parstat\_list.codelength, \text{ if} \\ \quad stat\_list_1.costat=true \end{cases}$$

Note: in this syntactical rule there is code generation if $stat\_list_1.costat=false$

viii) stat_list::=parstat_list

$$stat\_list.codelength=\begin{cases} parstat\_list.codelength+5, \\ \quad \text{if } stat\_list.costat=false \\ parstat\_list.codelength, \\ \quad \text{if } stat\_list.costat=true \end{cases}$$

stat_list.level=0

$$parstat\_list.codeloc=\begin{cases} stat\_list.\ codeloc+4, \\ \quad \text{if } stat\_list.costat=false \\ stat\_list.codeloc, \\ \quad \text{if } stat\_list.costat=true \end{cases}$$

Note: in this syntactical rule there is code generation if stat_list.costat=false

ix) $parstat\_list_1$::=statement **and** $parstat\_list_2$
$parstat\_list_1.codelength=statement.codelength+parstat\_list_2.codelength$

$parstat\_list_1.csloc=((statement.codeloc, statement.level),$
$(a_1, b_1), \ldots, (a_k, b_k))$, where $((a_1, b_1), \ldots,$
$\ldots, (a_k, b_k))=parstat\_list_2.csloc$

statement.loclev=(0, 0)
statement.costat=true
$parstat\_list_2.codeloc=parstat\_list_1.codeloc+statement.codelength$

x) parstat_list::= $statement_1$ **and** $statement_2$

parstat_list.codelength = $statement_1$.codelength + $statement_2$.codelength

parstat_list.csloc = (($statement_1$.codeloc, $statement_1$.level), ($statement_2$.codeloc, $statement_2$.level))

$statement_1$.loclev = (0, 0)

$statement_1$.costat = true

$statement_2$.codeloc = parstat_list.codeloc + $statement_1$.codelength

$statement_2$.loclev = (0, 0)

$statement_2$.costat = ture

xi) statement::= block

$$\text{statement.codelength} = \begin{cases} \text{block.codelength} + 5, \text{ if} \\ \quad \text{statement.costat} = \text{true} \\ \text{block.codelength, if} \\ \quad \text{statement.costat} = \text{false} \end{cases}$$

$$\text{block.codeloc} = \begin{cases} \text{statement.codeloc} + 3, \text{ if} \\ \quad \text{statement.costat} = \text{true} \\ \text{statement.codeloc, if} \\ \quad \text{statement.costat} = \text{false} \end{cases}$$

$$\text{block. loclev} = \begin{cases} \text{(statement.codeloc} + 1, \text{ statement.level), if} \\ \quad \text{statement.costat} = \text{true} \\ \text{statement.loclev, if} \\ \quad \text{statement.costat} = \text{false} \end{cases}$$

xii) statement::= coblock

$$\text{statement.codelength} = \begin{cases} \text{coblock.codelength} + 5, \text{ if} \\ \quad \text{statement.costat} = \text{true} \\ \text{coblock.codelength, if} \\ \quad \text{statement.costat} = \text{false} \end{cases}$$

statement.level = 0

$$\text{coblock.codeloc} = \begin{cases} \text{statement.codeloc} + 3, \text{ if} \\ \text{statement.costat} = \text{true} \\ \text{statement.codeloc, if} \\ \text{statement.costat} = \text{false} \end{cases}$$

The method given here was tested in the CDC 3300 computer of the Hungarian Academy of Sciences with the help of the HLP-SZ compiler generator system. A sequential programming language was augmented with the bracket pair **cobegin-coend** and the symbol **and**, and we have produced a compiler based on an ASE (alternating semantics evaluator) attribute evaluation strategy [6] which has the same number of passes (five) as the compiler generated for the basic sequential language has. This fact and the introduction of only three new attributes show us that the complexity of a compiler based on an ASE strategy does not increase by the introduction of the parallel structures discussed here.

4. Some remarks about the implementation of processes communicating through Hoare's monitors

We have implemented a very simple experimental language in which parallel processes communicate through Hoare's monitors [5]. The language is block structured, and the scope rule for monitors is the usual: a monitor reference can appear in the block where the monitor was declared, or in a block contained in it. The structure of a block is the following:

begin
declarations of monitors local to the block;
declarations of variables local to the block;
... the block body ...
end;

A declaration of monitors has the form:

monitor $m_1, m_2, \ldots, m_n$ **of** m;

and creates the monitors $m_1, m_2, \ldots, m_n$ of type m, where m is a monitor type declared at the beginning of the program. For simplicity each monitor type must be declared at the beginning of the program. (In other implementations monitor types could be declared at the beginning of the blocks with the same scope rule of monitors). The structure of a monitor type is the following:

```
type monitor_type_name monitor;
    begin
    declaration of the condition variables;
    declarations of variables local to the monitor;
    procedure procedure_name (...formal parameters...);
              declarations of the normal parameters;
              begin
              ...the procedure body...
              end;
    ...declarations of other procedures local to the monitor...;
    ...initialization of local data of the monitor...
    end;
```

In the implementation of the experimental language each monitor has its local data area which contains the variables of the monitor, the queues of processes waiting on a condition or on a monitor call, and the queue of processes waiting after an issue of a signal operation.

We have to introduce many new attributes. Four of them are the most important, and they will be described here: the synthesized attributes MTL and MINTRN, and the inherited attributes LMT and MTOTAL.

The attribute MTL is used to construct a table in which informations are collected about the declared monitor types. We put into the table the following informations about each monitor type:

— monitor type name;
— list of the variables local to the monitor type;
— list of the condition variables of the monitor type;
— list of the procedures local to the monitor type which contains on each procedure the parameters of the procedure, the name of the procedure, and the list of condition variables which appear in a "wait" statement in the procedure;

— the object code of the initialization of the data local to the monitor and the length of the code.

The attribute LMT leads the table (the address of the table) from the root dawnwards the leafs of the derivation tree.

The attribute MINTERN is used to construct a table collecting information about the declared monitors. We put into the table the following informations about each monitor:

— the monitor name;
— the monitor type of the monitor;
— the number of condition variables of the monitor and the number of variables local to the monitor;
— addresses and lengths of the queues of the condition variables of the monitor;
— address and length of the queue of processes waiting in a monitor call;
— address and length of the queue of processes waiting by an executed "signal" statement.

The attribute MTOTAL gives the table of the monitors valid in the environment with respect to the scope rule for monitors.

The HLP/SZ is based on the programming language SIMULA, so we can use classes and objects, and attributes of type reference to work with tables. In other compiler generator systems based on attribute grammars the concept of global attribute is introduced to make it easy to work with tables.

Abstract

This paper gives an attribute grammar for the translation of parallel programming structures: the bracket pair **cobegin-coend** and the symbol **and**. The introduction of these constructions into a programming language does not increase the complexity of a compiler based on an ASE attribute evaluation strategy. We discuss the implementation of Hoare's monitors by means of attribute grammars. The methods given here were tested in a CDC 3300 computer of the Hungarian Academy of Sciences.

References

[1] ALVAREZ, R.: Giving mathematical semantics of nondeterministical and parallel programming structures by means of attribute grammars. Acta Cybernetica, Tom. 7. Fasc. 4. 1986. 413—423.
[2] CONWAY, M.: A multiprocessor system design. Proc. AFIPS 1963, Fall Joint Comput. Conf., 24. Spartan Books, New York, 139—146.
[3] DENNIS, J. B. and VON HORN, E. C.: Programming semantics for multiprogrammed computations. CACM 9, 3 (March 1966), 143—155.
[4] GANZINGER, H., RIPKEN, K. and WILHELM, R.: Automating Generation of Optimizing Multipass Compilers. In Information Processing '77, North-Holland Publ. Co., 1977, 535—540.
[5] HOARE, C. A. R.: Monitors: An Operating System Structuring Concept. CACM 17, 10 (October 1974), 549—557.
[6] JAZAYERI, M. and WALTER, K. G.: Alternating Semantic Evaluator. In Proc. of ACM 1975 Ann. Conf., 230—234.
[7] KASTENS, U.: GAG: A Practical Compiler Generator. Lecture Notes in Computer Sciences 141, 1982.
[8] LEWI, J., DE VLAMINCK, K., HUENS, J. and HUYBRECHTS, M.: A Programming Methodology in Compiler Construction. Part 1: Concepts. North-Holland Publ. Co., 1982.

[9] Raiha, K. J., Soarinen, M., Soisolon-Soinen, E. and Teinari, M.: The Compiler Writing System HLP (Helsinki Language Processor), Department of Computer Science, Report A-1978-2, University of Helsinki.
[10] Shaw, A. C.: The logical design of operating systems. Prentice Hall Inc., 1974.
[11] Simon, E. and Gyimóthy, T.: Attributum nyelvtanok és alkalmazásuk. Akadémiai Pályamunka. MTA Automataelméleti Tanszéki Kutató Csoport, Szeged, 1983.
[12] Makay, Á., Gyimóthy, T., Simon, E.: An implementation of the HLP. Acta Cybernetica, Tom. 6. Fasc. 3, 1984. 315—327.

(Received January 11, 1989)

Further remarks on fully initial grammars

ALEXANDRU MATEESCU and GHEORGHE PĂUN

University of Bucharest, Faculty of Mathematics, Str. Academiei 14, București, 70109 ROMANIA

We investigate those languages generated by (context-free) grammars in which all nonterminals are regarded as axioms (problem raised by S. Horváth, at a formal language workshop, in Budapest, 1987). Among the considered topics, we can list: motivations, necessary conditions, right/left — regular/linear variants (generative capacity and closure properties), and other questions.

1. Motivations

In a usual context-free grammar (in general, in a Chomsky grammar), a nonterminal symbol is distinguished and taken as axiom (all derivations have to start from this nonterminal). This is motivated by mathematical reasons, as well as by the "classical" applications of Chomsky grammars, namely in modelling the syntax of natural or programming languages. However, there are many circumstances where this restriction is not important. This was the reason for which S. Horváth proposed to consider grammars in which *a certain amount* of nonterminals are allowed to be axioms. In [3], [9], grammars in which *all* nonterminals are axioms are considered (they are called *fully initial*).

Besides the naturalness of this idea, many further reasons can be invoked for dealing with several-axiom grammars. Here are some of them. (1) For instance, in W-grammars (two-level grammars) [11], the meta-level is a context-free grammar for which no axiom is distinguished. (2) In pure grammars [7], one considers finite sets of axioms. (3) According to the well-known Ginsburg—Rice—Schutzenberger theorem, each context-free language is a component of the minimal solution of a system of equations on a free monoid [4]; the study of equation systems does not involve special variables ("start" variables). (4) Moreover, in [6] systems of equations in which the iteration process starts from an arbitrary n-tuple of finite sets (not from an n-tuple of empty sets, as usual) are considered; in this way a characterization of EOL languages is obtained. (5) The ADJ group [1] associates a many-sorted initial algebra with a context-free grammar so that the language generated by this grammar is the homomorphic image of a certain carrier of the initial algebra. The construction of this many-sorted initial algebra does not depend on the start symbol of the corresponding context-free grammar. (6) Generalizing the definition of hipernotions in W-grammars, in [2] H-systems are introduced and investigated; in them the start symbol is replaced by an arbitrary (not necessarily finite) language; the language generated by an H-system is then defined by using homomorphisms, not production rules.

As one can see, there are enough reasons for further investigation of grammars in which more than one (or all) nonterminals are axioms. Moreover, as an *a posteriori* reason, the problems raised and the results obtained about these grammars prove that the subject is worth considering, leading to interesting new insights about Chomsky grammars.

2. Definitions and notations

For a vocabulary V, we denote by V^* the free monoid generated by V under the operation of concatenation, and λ is the null element. The length of a string $x\in V^*$ is denoted by $|x|$. Inclusion and strict inclusion are denoted by $\subseteq$ and $\subset$, respectively.

A Chomsky grammar is a quadruple $G=(V_N, V_T, S, P)$; V_N is the nonterminal vocabulary, V_T is the terminal one, $S\in V_N$ is the axiom and P is the production set. The usual language generated by G is defined by

$$L(G) = \{x\in V_T^* | S \overset{*}{\Rightarrow} x\}.$$

The fully initial language generated by G is

$$L_{\mathrm{in}}(G) = \{x\in V_T^* | A \overset{*}{\Rightarrow} x \text{ for some } A\in V_N\}.$$

Clearly, $L(G)\subseteq L_{\mathrm{in}}(G)$. The family of languages generated by Chomsky grammars of type i, $i=0, 1, 2, 3$, is denoted by $\mathscr{L}_i$. The family of fully initial languages generated by grammars of type i is denoted by $\mathscr{F}\mathscr{L}_i$, $i=0, 1, 2, 3$.

When dealing with the fully initial language only, we shall write a grammar in the form $G=(V_N, V_T, P)$, thus omitting the useless axiom.

Usually, a language is said to be of type 3 if it can be generated by a right-linear or a left-linear grammar, in the classical case. (Right-linear and left-linear grammars have the same generative power.) For fully initial grammars this is not true, therefore we shall distinguish several classes of "type-3" grammars.

A grammar $G=(V_N, V_T, P)$ is called *right-linear (left-linear)* if $P\subseteq V_N\times (V_T^*\cup V_T^* V_N)$ $(P\subseteq V_N\times (V_T^*\cup V_N V_T^*))$. We denote by $\mathscr{F}\mathscr{L}_{\mathrm{rlin}}$, $\mathscr{F}\mathscr{L}_{\mathrm{llin}}$ the corresponding families of fully initial languages. Moreover, we distinguish between grammars with rules of the form $A\to xB$ with an arbitrary string $x\in V_T^*$ as above and grammars in which x must be a terminal. A grammar $G=(V_N, V_T, P)$ is called *right-regular (left-regular)*, if $P\subseteq V_N\times (V_T\cup V_T V_N)$ $(P\subseteq V_N\times (V_T\cup V_N V_T))$. The corresponding families of fully initial languages are denoted by $\mathscr{F}\mathscr{L}_{\mathrm{rreg}}$, $\mathscr{F}\mathscr{L}_{\mathrm{lreg}}$.

The above family $\mathcal{FL}_3$ is, in fact, $\mathcal{FL}_{\mathrm{rlin}} \cup \mathcal{FL}_{\mathrm{llin}}$. We shall also denote this family by $\mathcal{FL}_{\mathrm{lin}}^{\cup}$ and we shall consider the following families too:

$$\mathcal{FL}_{\mathrm{lin}}^{\cap} = \mathcal{FL}_{\mathrm{rlin}} \cap \mathcal{FL}_{\mathrm{llin}},$$

$$\mathcal{FL}_{\mathrm{reg}}^{\cup} = \mathcal{FL}_{\mathrm{rreg}} \cup \mathcal{FL}_{\mathrm{lreg}},$$

$$\mathcal{FL}_{\mathrm{reg}}^{\cap} = \mathcal{FL}_{\mathrm{rreg}} \cap \mathcal{FL}_{\mathrm{lreg}}.$$

As in many cases, we shall consider two languages identical if they differ by at most the empty string λ.

The sets of prefixes, suffixes and subwords of a given string x are denoted by Init (x), Fin (x), Sub (x), respectively, and these notations will be extended in the natural way to languages. When considering only proper prefixes, suffixes and subwords, we shall write Initp (x), Finp (x) and Subp (x), respectively.

For further details in formal language theory, the reader is referred to [10].

3. Necessary conditions for the context-free case

We shall consider here some necessary conditions for a language to be in $\mathcal{FL}_2$; some of these conditions will be also particularized to $\mathcal{FL}_3$ or to subfamilies of $\mathcal{FL}_3$.

Lemma 1. For each language $L \in \mathcal{FL}_2$ there is a λ-free grammar $G=(V_N, V_T, P)$ such that P does not contain chain rules (rules of the form $A \to B$, $A, B \in V_N$) and $L = L_{\mathrm{in}}(G)$.

Proof. The same as for usual context-free languages.

Lemma 2. For each language $L \in \mathcal{FL}_2$, $L \subseteq V^*$, there are two positive integers p, q such that each $z \in L$, $|z| > p$, can be written as $z = uvwxy$, $u, v, w, x, y \in V^*$, so that

(i) $|vwx| \leqq q$, $|vx| > 0$,

(ii) for all $k \geqq 0$, $uv^k wx^k y \in L$ and $v^k wx^k \in L$.

Proof. The same as for usual context-free languages, with the following two remarks:

— we start from a reduced grammar, G, in the sense of Lemma 1 (see Lemma 3.1.1 in [4]), not from a Chomsky normal form grammar (as in Theorem 6.4 in [10]);

— given a derivation tree T, all subtrees having the roots in the nonterminals of T correspond to substrings of the string associated to T and which belong to the fully initial language generated by the grammar; therefore, when we have a derivation $S \stackrel{*}{\Rightarrow} uAy \stackrel{*}{\Rightarrow} uvAxy \stackrel{*}{\Rightarrow} uvwxy$, then both $uv^k wx^k y$ and $v^k wx^k$ belong to $L_{\mathrm{in}}(G)$.

Corollary 1. If $L \in \mathcal{FL}_2$, then there is a constant p such that for all $z \in L$, $|z| > p$, we have Subp $(z) \cap L \neq \emptyset$.

Proof. Take p as in Lemma 2 and, for $z \in L$, $|z| > p$, write $z = uvwxy$ with the above properties. As $v^k wx^k \in L$ for $k \geqq 0$, when $k = 0$, we obtain $w \in L \cap \mathrm{Sub}(z)$. Moreover, $|vx| > 0$, hence we have, in fact, $w \in L \cap \mathrm{Subp}(z)$.

Corollary 2. If $L \in \mathcal{FL}_2$ is an infinite language, then also $L \cap \mathrm{Subp}(L)$ is infinite.

Proof. Let p, q be the constants of Lemma 2 and take $z \in L$, $|z| > p$, $z = uvwxy$. Each string $v^k w x^k$, $k \geqq 0$, is in L. Clearly, $v^k w x^k \in \mathrm{Subp}(L)$ and $v^k w x^k \neq v^{k+1} w x^{k+1}$, $k \geqq 0$ (we have $|vx| > 0$), therefore $L \cap \mathrm{Subp}(L)$ contains the infinite set $\{v^k w x^k | k \geqq 0\}$.

Lemma 3. The conditions (properties) in the above two corollaries are independent from one another.

Proof. We consider the languages

$$L_1 = \{a\} \cup \{ab^n a | n \geqq 1\}$$

and

$$L_2 = \{ba^n b, cba^n bc | n \geqq 1\}.$$

The first language fulfils the condition in Corollary 1 (take $p=1$; $\mathrm{Subp}(ab^n b) \cap \cap L_1 = \{a\}$ for all $n \geqq 1$), but not that in Corollary 2 ($\mathrm{Subp}(L_1) \cap L_1 = \{a\}$). The second language fulfils the condition in Corollary 2 ($L_2 \cap \mathrm{Subp}(L_2) = \{ba^n b | n \geqq 1\}$), but not that in Corollary 1 (the strings $ba^n b$, irrespective of their length, have no proper subwords in L_2).

This lemma shows that none of the conditions in Corollaries 1 and 2 is sufficient for a language to be in $\mathcal{FL}_2$; even they together are insufficient for that, as it follows from the next result.

Lemma 4. The condition in Lemma 2 is strictly stronger than the conditions in Corollaries 1 and 2 together.

Proof. We consider the language

$$L = \{b\} \cup \{ba^n b, cba^n bc | n \geqq 1\}.$$

It is easy to see that both conditions in Corollaries 1 and 2 are fulfilled (similarly to languages L_1, L_2 in the above proof), but that in Lemma 2 is not. Indeed, let p and q be two positive integers and take $z = ba^n b$, $|z| > p$ (there are arbitrarily long such strings in L). We must have $z = uvwxy$ such that $v^k w x^k \in L$, $k \geqq 0$, $|vx| > 0$. It follows that $vx \in \{a^n | n \geqq 1\}$, hence $v^k w x^k$ is of the form $a\alpha$ or of the form αa, $\alpha \in \{a, b\}^*$. Such strings cannot be in L, a contradiction.

Lemma 5. The condition in Lemma 2 is not sufficient for a language to be in $\mathcal{FL}_2$.

Proof. Let us consider the language

$$L = \{a^n | n \geqq 0\} \cup \{b^n | n \geqq 0\} \cup \{a^n b^{2^m} | n, m \geqq 1\}.$$

The language L is not context-free; as $\mathcal{FL}_2 \subset \mathcal{L}_2$ [3], it follows that $L \notin \mathcal{FL}_2$. However, this language fulfils the condition in Lemma 2. Take, for instance, $p=1$, $q=1$. For $z=a^n$ or $z=b^n$, we clearly have all conditions in lemma fulfilled. If $z=a^n b^{2^m}$ we take $u=\lambda$, $v=a$, $w=\lambda$, $x=\lambda$, $y=a^{n-1}b^{2^m}$. Obviously, $z=uvwxy$, $|vwx| \leqq q=1$, $|vx|>0$, $uv^k w x^k y = a^k a^{n-1} b^{2^m} \in L$ for all $k \geqq 0$ (for $k=0$, $n=1$ we can obtain $uv^k w x^k y = b^{2^m}$, which is in L too), and $v^k w x^k = a^k \in L$ for all $k \geqq 0$.

Conjecture 1. If L is a context-free language which fulfils the condition in Lemma 2, then $L\in\mathcal{FL}_2$.

We consider now a necessary condition of different type, similar to the one used in the theory of Marcus contextual languages [8].

Definition. For a given language $L\subseteq V^*$, let

$$\mathrm{Min}(L)=\{z\in L|\mathrm{Subp}(z)\cap L=\emptyset\}$$

and define

$$R_1(L)=\mathrm{Min}(L)$$

$$R_i(L)=R_{i-1}(L)\cup\mathrm{Min}(L-R_{i-1}(L)),\quad i\geqq 2.$$

We say that L has *property* R iff all the sets $R_i(L)$, $i\geqq 1$, are finite.

Lemma 6. If $L\in\mathcal{FL}_2$, then L has property R.

Proof. Let $L\in\mathcal{FL}_2$, $L\subseteq V^*$, be a language and take a grammar $G=(V_N, V_T, P)$ such that $L_{\mathrm{in}}(G)=L$ and G does not contain λ-rules and chain rules (Lemma 1). For a string $x\in L$, let $T(x, G)$ be the set of all derivation trees describing derivations of x in G starting from a nonterminal in V_N (which is the root of a tree). Denote by $\mathrm{hei}(T)$ the *height* of a given tree $T\in T(x, G)$, i.e. the maximum of lengths of paths linking the root of T to its leafs (symbols in x). For a given string x we define

$$\mathrm{hei}_G(x)=\max\{\mathrm{hei}(T)|T\in T(x, G)\}.$$

Then we have

$$R_i(L)\subseteq\{x\in L|\mathrm{hei}_G(x)\leqq i\},\quad i\geqq 1.$$

Indeed, let $x\in\mathrm{Min}(L)$ be a string and take a derivation $D\colon A\overset{*}{\Rightarrow}x$ in G corresponding to a tree T. If $\mathrm{hei}(T)\geqq 2$, then the derivation D is of the form D: $A\Rightarrow\alpha_1\alpha_2\dots\alpha_k\overset{*}{\Rightarrow}\beta_1\beta_2\dots\beta_k=x$, $\alpha_i\in V_N\cup V_T$, $\alpha_i\overset{*}{\Rightarrow}\beta_i$, $1\leqq i\leqq k$, $k\geqq 2$, and for some i, $1\leqq i\leqq 2$, $\alpha_i\in V_N$. This implies $\beta_i\in L\cap\mathrm{Subp}(z)$, hence $z\notin\mathrm{Min}(L)$, a contradiction. In conclusion, $\mathrm{hei}(T)=1$, $\mathrm{hei}_G(x)=1$, and the inclusion $R_i(L)\subseteq\{x\in L|\mathrm{hei}_G(x)\leqq i\}$ holds for $i=1$.

Let us assume, this relation is true for $j=1,2,\dots,i$, $i\geqq 1$, and consider $x\in R_{i+1}(L)$. If $x\in R_i(L)$, then $\mathrm{hei}_G(x)\leqq i$ by the induction hypothesis. Assume that $x\in R_{i+1}(L)-R_i(L)$, that is $x\in\mathrm{Min}(L-R_i(L))$. In other terms, $\mathrm{Subp}(x)\cap\cap(L-R_i(L))=\emptyset$. Suppose that $\mathrm{hei}_G(x)>i+1$, and take a derivation tree $T\in T(x, G)$ such that $\mathrm{hei}(T)>i+1$. There is a derivation D, associated with this tree, having the form D: $A\Rightarrow\alpha_1\alpha_2\dots\alpha_k\overset{*}{\Rightarrow}\beta_1\beta_2\dots\beta_k=x$, such that $\alpha_j\in V_N\cup V_T$, $\alpha_j\overset{*}{\Rightarrow}\beta_j$, $1\leqq j\leqq k$ ($\alpha_j=\beta_j$ if $\alpha_j\in V_T$), $k\geqq 2$, and there is an $\alpha_j\in V_N$ for some j, $1\leqq j\leqq k$. All strings β_j, $1\leqq j\leqq k$, belong to $\mathrm{Subp}(x)\cap L$. As $\mathrm{Subp}(x)\cap(L-R_i(L))=\emptyset$, we must have $\beta_j\in R_i(L)$. By the induction hypothesis we get $\mathrm{hei}_G(\beta_j)\leqq i$, $1\leqq j\leqq k$. This implies that the tree T consists of a "root level" describing the rule $A\to\alpha_1\alpha_2\dots\alpha_j$ and of all trees associated with subderivations $\alpha_j\overset{*}{\Rightarrow}\beta_j$, for $\alpha_j\in V_N$. In conclusion, $\mathrm{hei}(T)\leqq\leqq i+1$, a contradiction. We obtain $\mathrm{hei}_G(x)\leqq i+1$, which completes the induction argument.

The sets $\{x\in L|\mathrm{hei}_G(x)\leqq i\}$, $i\geqq 1$, are clearly finite, therefore the sets $R_i(L)$, $i\geqq 1$, are finite too, and the proof is completed.

Lemma 7. The property R implies conditions in Corollaries 1, 2, but there are languages fulfilling both these conditions without having the property R.

Proof. Consider again the language L in the proof of Lemma 4 (it satisfies the conditions in Corollaries 1 and 2). We obtain

$$R_1(L) = \{b\},$$
$$R_2(L) = \{b\}\cup\{ba^n b \mid n \geqq 1\},$$

hence $R_2(L)$ is infinite, L does not have the property R.

Define now, for a given language L,

$$p = \max\{|x| \mid x\in R_1(L)\}$$

If $z\in L$, $|z|>p$, then $z\notin R_1(L)$, hence $\mathrm{Subp}(z)\cap L\neq\emptyset$. The property R implies thus the condition in Corollary 1.

Consider an infinite language L having the property R but not having the property in Corollary 2, that is $L\cap\mathrm{Subp}(L)$ is finite, $\mathrm{card}(L\cap\mathrm{Subp}(L))=t$. As L is infinite, but all sets $R_i(L)$, $i\geqq 1$, are finite, it follows that $R_i(L)\subset R_{i+1}(L)$, $i\geqq 1$ (if $R_j(L)=R_{j+1}(L)$, then $R_j(L)=R_{j+k}(L)$, $k\geqq 1$, hence $L\subseteq R_j(L)$, a contradiction). As $R_{i+1}(L)-R_i(L)=\mathrm{Min}(L-R_i(L))\neq\emptyset$, it follows that $R_{i+1}(L)\cap(L\cap\cap\mathrm{Subp}(L))\neq\emptyset$ and $R_i(L)\cap(L\cap\mathrm{Subp}(L))\subset R_{i+1}(L)\cap(L\cap\mathrm{Subp}(L))$ for all $j\geqq 1$. This implies $\mathrm{card}(R_{t+1}(L)\cap L\cap\mathrm{Subp}(L))\geqq t+1$, therefore $\mathrm{card}(L\cap\cap\mathrm{Subp}(L))\geqq t+1$, a contradiction.

Lemma 8. The condition R is not sufficient for a (context-free) language to be in $\mathscr{FL}_2$.

Proof. We consider the language

$$L = \{a^n \mid n \geqq 1\}\cup\{ab^n a^n \mid n \geqq 1\}.$$

This is a context-free language and we have

$$R_1(L) = \{a\},$$
$$R_i(L) = \{a^j \mid 1 \leqq j \leqq i\}\cup\{ab^j a^j \mid 1 \leqq j \leqq i-1\}, \quad i \geqq 2,$$

therefore the property R is observed.

However, this language is not in $\mathscr{FL}_2$. Assume the contrary, and factorize a long enough $z=ab^n a^n$ in L into $z=uvwxy$ as in Lemma 2. Then we must have $v=b^i$, $x=a^i$, $i>0$, which implies that all $v^k wx^k=b^{ik}wa^{ik}$, $k\geqq 0$, are in L, a contradiction to the form of strings in L.

Remark 1. The above proof shows that if Conjecture 1 were proved then, for context-free languages, the condition in Lemma 2 would be stronger than property R.

Conjecture 2. For arbitrary languages, the condition in Lemma 2 is stronger than property R.

Remark 2. If in condition (ii) of Lemma 2 we take $k\geqq 1$ instead of $k\geqq 0$ (sometimes, the pumping lemma is formulated in this weaker form; see [4], for instance), then the modified condition will be independent of condition R. The language

L in the above proof supports one of the implications; the other one can be proved using the language

$$L = \{ba^n ba^n b^m a | n,\ m \geqq 1\} \cup \{a^n ba^n | n \geqq 1\}.$$

Taking $p=1$, $q=3$ we obtain the modified property in Lemma 2, but we have

$$R_1(L) = \{aba\},$$

$$R_2(L) = \{aba,\ a^2ba^2\} \cup \{babab^m a | m \geqq 1\},$$

hence property (condition) R is not satisfied.

Lemma 2 has some particular forms for right/left linear grammars.

Lemma 9. (i) If $L \in \mathscr{F}\mathscr{L}_{\mathrm{rlin}}$, then there are two positive integers p, q such that, for all $z \in L$, $|z| > p$, we can write $z = uvw$, $0 < |v| \leqq q$ and $uv^i w \in L$, $v^i w \in L$, for all $i \geqq 0$.

(ii) If $L \in \mathscr{F}\mathscr{L}_{\mathrm{llin}}$, then there are two positive integers p, q such that, for all $z \in L$, $|z| > p$, we can write $z = uvw$, $0 < |v| \leqq q$ and $uv^i w \in L$, $uv^i \in L$, for all $i \geqq 0$.

Proof. Obvious particularizations of the proof of Lemma 2 to right/left linear grammars.

4. Fully initial languages in the Chomsky hierarchy

As we have mentioned, in [3] it is proved that $\mathscr{F}\mathscr{L}_2 \subset \mathscr{L}_2$. A more precise (and more general) result is true, namely we have.

Theorem 1. The following diagram holds:

$$\begin{tikzcd}
 & \mathscr{L}_3 \arrow[dr] & & & \\
\mathscr{F}\mathscr{L}_3 \arrow[ur] \arrow[dr] & & \mathscr{L}_2 \arrow[r] & \mathscr{L}_1 = \mathscr{F}\mathscr{L}_1 \arrow[r] & \mathscr{L}_0 = \mathscr{F}\mathscr{L}_0, \\
 & \mathscr{F}\mathscr{L}_2 \arrow[ur] & & &
\end{tikzcd}$$

where $\rightarrow$ indicates a strict inclusion; the families $\mathscr{L}_3$, $\mathscr{F}\mathscr{L}_2$ are incomparable.

Proof. As $\{ba^n b | n \geqq 1\}$ is not in $\mathscr{F}\mathscr{L}_2$ (it fulfils no necessary condition in the previous section), it follows that $\mathscr{L}_3 - \mathscr{F}\mathscr{L}_2 \neq \emptyset$, hence also $\mathscr{L}_2 - \mathscr{F}\mathscr{L}_2 \neq \emptyset$, $\mathscr{L}_3 - \mathscr{F}\mathscr{L}_3 \neq \emptyset$. On the other hand, $\{a^n b^n | n \geqq 1\}$ is in $\mathscr{F}\mathscr{L}_2 - \mathscr{L}_3$, hence $\mathscr{F}\mathscr{L}_2 - \mathscr{F}\mathscr{L}_3 \neq \emptyset$ and $\mathscr{L}_3, \mathscr{F}\mathscr{L}_2$ are incomparable.

Consider now a grammar G of arbitrary type, $G = (V_N, V_T, P)$ and construct the grammar $G' = (V_N \cup \{S'\}, V_T, S', P \cup \{S' \rightarrow A | A \in V_N\})$. Clearly, G' is of the same type as G and $L(G') = L_{\mathrm{in}}(G)$, hence $\mathscr{F}\mathscr{L}_i \subseteq \mathscr{L}_i$, $i = 0, 1, 2, 3$.

In order to complete the proof, we have to prove that $\mathscr{L}_i \subseteq \mathscr{F}\mathscr{L}_i$, $i=0, 1$. Take a language $L \in \mathscr{L}_i$, $L \subseteq V^*$. We can write

$$L = \bigcup_{a \in V} \{a\} \partial_a L \cup \{x \in L \mid |x| \leqq 2\}$$

($\partial_a L$ is the left derivative of L with respect to a). As $\mathscr{L}_i$, $i=0, 1$, are closed under left derivative, $\partial_a L \in \mathscr{L}_i$. Let $G_a = (V_{N,a}, V, S_a, P_a)$ be a type-i grammar for $\partial_a L$. Assume the $V_{N,a}$ are pairwise disjoint and define $G=(V_N, V, S, P)$ with

$$V_N = \bigcup_{a \in V} V_{N,a} \cup \{X_a \mid a \in V\} \cup \{a' \mid a \in V\} \cup \{S\},$$

$$P = \{S \to x \mid x \in L,\ |x| \leqq 2\} \cup \{S \to X_a S_a \mid a \in V\} \cup$$

$$\cup \{\alpha u' \to \alpha v' \mid \alpha \in V_N,\ u \to v \in P_a,\ a \in V\} \cup$$

$$\cup \{X_a b' \to ab \mid a, b \in V\} \cup \{ab' \to ab \mid a, b \in V\}$$

where u' is the string obtained from u by replacing each $a \in V$ by $a' \in V_N$. It is easy to see that no derivation $A \overset{*}{\Rightarrow} w$, $A \in V_N$, is possible in G unless $A=S$, therefore $L(G)=L_{\text{in}}(G)$. Moreover, $L(G)=L$. In conclusion, $L \in \mathscr{F}\mathscr{L}_i$, $i=0, 1$, and the proof is ended.

This theorem shows that families $\mathscr{F}\mathscr{L}_0$ and $\mathscr{F}\mathscr{L}_1$ request no further investigations.

5. Type-3 fully initial languages

First, let us consider some characterizations and representations of languages in $\mathscr{F}\mathscr{L}_{\text{rreg}}$, $\mathscr{F}\mathscr{L}_{\text{lreg}}$, $\mathscr{F}\mathscr{L}_{\text{rlin}}$, $\mathscr{F}\mathscr{L}_{\text{llin}}$.

Lemma 10. (i) $L \in \mathscr{F}\mathscr{L}_{\text{rreg}}$ if and only if $L \in \mathscr{L}_3$ and $L = \text{Fin}(L)$. (ii) $L \in \mathscr{F}\mathscr{L}_{\text{lreg}}$ if and only if $L \in \mathscr{L}_3$ and $L = \text{Init}(L)$. (iii) $L \in \mathscr{F}\mathscr{L}^{\cap}_{\text{reg}}$ if and only if $L \in \mathscr{L}_3$ and $L = \text{Sub}(L)$.

Proof. (i) Let $L \in \mathscr{F}\mathscr{L}_{\text{rreg}}$ be a language such that $L = L_{\text{in}}(G)$, $G=(V_N, V_T, P)$. Clearly, $L \in \mathscr{L}_3$ and $L \subseteq \text{Fin}(L)$. Take a string $w \in \text{Fin}(L)$. There is a $u \in V_T^*$ such that $uw \in L$. Therefore, there is a derivation $A \overset{*}{\Rightarrow} uw$ in G. As G is a right-regular grammar, there is a $B \in V_N$ such that $A \overset{*}{\Rightarrow} uB \overset{*}{\Rightarrow} uw$, which implies $w \in L_{\text{in}}(G) = L$. In conclusion, $w \in L$, $\text{Fin}(L) \subseteq L$.

Conversely, let $L \in \mathscr{L}_3$, $L = \text{Fin}(L)$, and consider a reduced right-regular grammar G, $G=(V_N, V_T, S, P)$, without useless nonterminals, $L=L(G)$, $P \subseteq V_N \times \times (V_T \cup V_T V_N)$. Clearly, $L(G) \subseteq L_{\text{in}}(G)$. Take a string $w \in L_{\text{in}}(G)$. There is a derivation $A \overset{*}{\Rightarrow} w$ in G, $A \in V_N$. As G is reduced, there is a derivation $S \overset{*}{\Rightarrow} uA$, $u \in V_T^*$, therefore $S \overset{*}{\Rightarrow} uA \overset{*}{\Rightarrow} uw$ is possible in G. This implies $w \in \text{Fin}(L(G)) = L$, that is $w \in L$, hence $L_{\text{in}}(G) \subseteq L(G)$. In conclusion, $L_{\text{in}}(G) = L$, $L \in \mathscr{F}\mathscr{L}_{\text{rreg}}$ and (i) is proved.

(ii) Analogously.

(iii) Follows from the definition of $\mathscr{F}\mathscr{L}^{\cap}_{\text{reg}}$, the above parts (i) and (ii) and the relations $\text{Sub}(L) = \text{Fin}(\text{Init}(L)) = \text{Init}(\text{Fin}(L)) = \text{Init}(\text{Sub}(L)) = \text{Fin}(\text{Sub}(L))$.

Denote by $\text{Mi}(w)$ the mirror image of a string w and extend this operation to languages.

Lemma 11. (i) $L\in\mathscr{F}\mathscr{L}_{\rm rreg}$ if and only if $\mathrm{Mi}(L)\in\mathscr{F}\mathscr{L}_{\rm lreg}$. (ii) $L\in\mathscr{F}\mathscr{L}_{\rm rlin}$ if and only if $\mathrm{Mi}(L)\in\mathscr{F}\mathscr{L}_{\rm llin}$.

Proof. (i) Take a language $L\in\mathscr{F}\mathscr{L}_{\rm rreg}$, generated by $G=(V_N, V_T, P)$ and define $G'=(V_N, V_T, \{A\to\mathrm{Mi}(x)|A\to x\in P\})$. Clearly, $L_{\rm in}(G')=\mathrm{Mi}(L(G))=\mathrm{Mi}(L)$, hence $\mathrm{Mi}(L)\in\mathscr{F}\mathscr{L}_{\rm lreg}$. The converse implication is analogous.
(ii) Similar.

Lemma 12. (i) Each language in $\mathscr{F}\mathscr{L}_{\rm rlin}$ is a homomorphic image of a language in $\mathscr{F}\mathscr{L}_{\rm rreg}$. (i) Each language in $\mathscr{F}\mathscr{L}_{\rm llin}$ is a homomorphic image of a language in $\mathscr{F}\mathscr{L}_{\rm lreg}$.

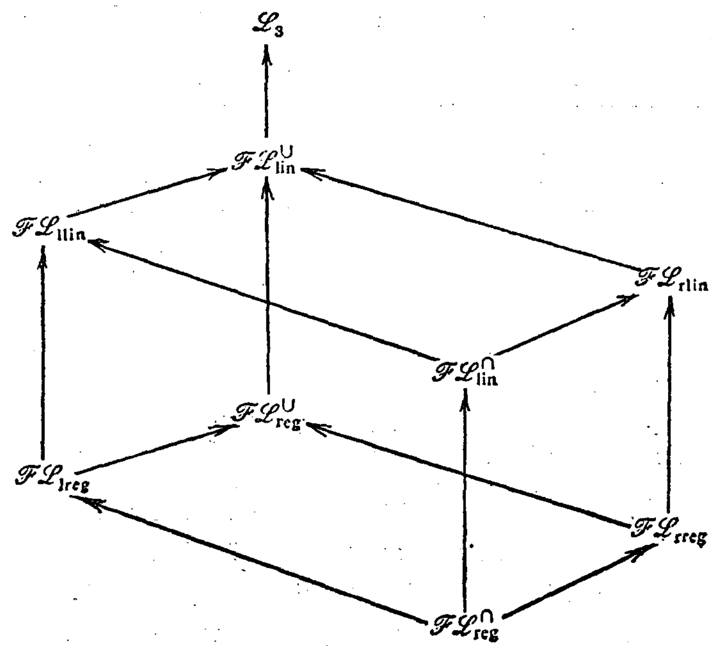
Proof. (i) Let $L\subseteq V^*$, $L\in\mathscr{F}\mathscr{F}_{\rm rlin}$, be a language generated by the grammar $G=(V_N, V, P)$. We define the grammar $G'=(V_N, V', P')$ by

$$V' = \{[\alpha]|X\to\alpha Y \text{ or } X\to\alpha \text{ is in } P,\ \alpha\in V^*,\ X, Y\in V_N\},$$

$$P' = \{X\to[\alpha]Y|X\to\alpha Y\in P\}\cup\{X\to[\alpha]|X\to\alpha\in P\}.$$

Consider also the homomorphism $h\colon V'^*\to V^*$ defined by $h([\alpha])=\alpha$, $[\alpha]\in V'$. Clearly, G' is a right-regular grammar and $h(L_{\rm in}(G'))=L$.
(ii) Analogously.

Theorem 2. The inclusion relations between the above discussed families of type-3 fully initial languages are those in the next diagram ($\rightarrow$ indicates a strict inclusion; the unlinked families are incomparable).



Proof. All inclusions are obvious. Moreover, we have: $ba^*\in\mathcal{FL}_{\rm lreg}-\mathcal{FL}_{\rm rreg}$, $a^*b\in\mathcal{FL}_{\rm rreg}-\mathcal{FL}_{\rm lreg}$ (use Lemma 10, parts (i), (ii)). This settles the relations on the bottom face of the "cube" in the diagram. Moreover, $c(ab)^*\in\mathcal{FL}_{\rm llin}-(\mathcal{FL}_{\rm rlin}\cup\cup\mathcal{FL}_{\rm lreg})$ and $(ab)^*c\in\mathcal{FL}_{\rm rlin}-(\mathcal{FL}_{\rm llin}\cup\mathcal{FL}_{\rm rreg})$. This settles the relations on the upper face of the "cube", as well as those indicated by the vertical edges, except $\mathcal{FL}^{\cap}_{\rm reg}\subset\mathcal{FL}^{\cap}_{\rm lin}$. This, however, follows from $(ab)^*\in\mathcal{FL}^{\cap}_{\rm lin}-\mathcal{FL}^{\cap}_{\rm reg}$ (use condition (iii) in Lemma 10). The inclusion $\mathcal{FL}^{\cup}_{\rm lin}=\mathcal{FL}_3\subset\mathcal{L}_3$ was shown in Theorem 1.

Theorem 3. The closure properties of the above discussed families of type-3 fully initial languages are as presented in Table 1 (Y indicates a positive closure result, N points out a negative closure result).

Table 1.

	$\mathcal{FL}^{\cup}_{\rm lin}$	$\mathcal{FL}_{\rm llin}$	$\mathcal{FL}_{\rm rlin}$	$\mathcal{FL}^{\cap}_{\rm lin}$	$\mathcal{FL}^{\cup}_{\rm reg}$	$\mathcal{FL}_{\rm lreg}$	$\mathcal{FL}_{\rm rreg}$	$\mathcal{FL}^{\cap}_{\rm reg}$
Union	N	Y	Y	Y	N	Y	Y	Y
Complementation	N	N	N	N	N	N	N	N
Intersection	N	N	N	N	N	Y	Y	Y
Concatenation	N	N	N	N	N	N	N	N
Kleene closure	Y	Y	Y	Y	Y	Y	Y	Y
Homomorphism	Y	Y	Y	Y	N	N	N	N
Inverse homomorphism	N	N	N	N	Y	Y	Y	Y
Mirror image	Y	N	N	Y	Y	N	N	Y
Right quotient	N	N	Y	N	N	N	Y	N
Left quotient	N	Y	N	N	N	Y	N	N
Init, Fin, Sub	Y	Y	Y	Y	Y	Y	Y	Y
gsm mapping	N	N	N	N	N	N	N	N
Inverse gsm mapping	N	N	N	N	N	N	N	N
Intersection with regular sets	N	N	N	N	N	N	N	N

Proof. Union. If L_1, L_2 are in $\mathcal{FL}_{\rm rreg}$, $\mathcal{FL}_{\rm lreg}$ or $\mathcal{FL}^{\cap}_{\rm reg}$ then $L_1\cup L_2$ belongs to the same families, as it easily follows from Lemma 10 ($L_1\cup L_2\in\mathcal{L}_3$ and $L_1\cup L_2==\mathrm{Fin}(L_1\cup L_2)$, $L_1\cup L_2=\mathrm{Init}(L_1\cup L_2)$, $L_1\cup L_2=\mathrm{Sub}(L_1\cup L_2)$, respectively). $\mathcal{FL}^{\cup}_{\rm reg}$ is not closed under union, because, for instance, $L_1=a^*b$, $L_2=ba^*$ are in $\mathcal{FL}^{\cup}_{\rm reg}$, but $L_1\cup L_2$ is not in $\mathcal{FL}^{\cup}_{\rm lin}$ ($L_1\cup L_2$ is neither in $\mathcal{FL}_{\rm rlin}$ nor in $\mathcal{FL}_{\rm llin}$: use Lemma 9). The closure of $\mathcal{FL}_{\rm rlin}$, $\mathcal{FL}_{\rm llin}$, $\mathcal{FL}^{\cap}_{\rm lin}$ can be proved by direct, standard constructions.

Complementation. The language $L=a^*b^*$ is in $\mathcal{FL}^{\cap}_{\rm reg}$, but $\{a, b\}^*-L$ is not in $\mathcal{FL}^{\cup}_{\rm lin}$ (use Lemma 9).

Intersection. The closure of $\mathcal{FL}_{\rm rreg}$, $\mathcal{FL}_{\rm lreg}$, $\mathcal{FL}^{\cap}_{\rm reg}$ can again be proved using Lemma 10 ($\mathrm{Fin}(L_1\cap L_2)\subseteq\mathrm{Fin}(L_1)\cap\mathrm{Fin}(L_2)=L_1\cap L_2$ hence $L_1\cap L_2=\mathrm{Fin}(L_1\cap L_2)$, $L_1\cap L_2\in\mathcal{L}_3$ etc.). For $\mathcal{FL}^{\cup}_{\rm reg}$ take $L_1=a^*b^+$, $L_2=a^+b^*$, both in this family; $L_1\cap L_2=a^+b^+$ does not belong to $\mathcal{FL}^{\cup}_{\rm reg}$. For the other families take

$$L_1 = c(aab)^*c\cup(aab)^*c\cup c(aab)^*\cup(aab)^*,$$

$$L_2 = ca(aba)^*abc\cup(aba)^*abc\cup ca(aba)^*\cup(aba)^*.$$

They belong to $\mathcal{FL}^{\cap}_{\rm lin}$, but $L_1\cap L_2=ca(aba)^*abc$ is not in $\mathcal{FL}^{\cup}_{\rm lin}$.

Concatenation. The languages $L_1=a^+$, $L_2=b^+$ are in $\mathcal{FL}_{\mathrm{reg}}^{\cap}$, but $L_1L_2=$ $=a^+b^+$ is not in $\mathcal{FL}_{\mathrm{lin}}^{\cup}$, which settles all cases.

Kleene closure. Given a right-regular or a right-linear grammar $G=(V_N, V_T, P)$, construct the grammar $G'=(V_N, V, P')$ with $P'=P\cup\{X\to\alpha Y|X\to\alpha\in P,\ \alpha\in V_T^*,$ $X, Y\in V_N\}$. It is easy to see that $L_{\mathrm{in}}(G')=L(G)^+$. The left-regular and left-linear cases can be treated similarly.

Homomorphism. A standard construction proves the positive results. For regular families take $L=a^+$ (it belongs to $\mathcal{FL}_{\mathrm{reg}}^{\cap}$) and the homomorphism $h\colon a^*\to\{a, b\}^*$ defined by $h(a)=ab$. The language $h(L)=(ab)^+$ is not $\mathcal{FL}_{\mathrm{reg}}^{\cup}$, which implies the nonclosure cases in Table 1.

Inverse homomorphism. Let $h\colon V^*\to V'^*$ be a homomorphism and $L\subseteq V^*$ a language in $\mathcal{FL}_{\mathrm{reg}}^{\cup}$. According to Lemma 10, $L\in\mathcal{L}_3$ and $L=\mathrm{Sub}(L)$. Clearly, $h^{-1}(L)\in\mathcal{L}_3$ and $h^{-1}(L)\subseteq\mathrm{Sub}(h^{-1}(L))$. Consider now a string u in $\mathrm{Sub}(h^{-1}(L))$. There are v, $w\in V'^*$ such that $vuw\in h^{-1}(L)$, hence $h(v)h(u)h(w)\in L$. This implies $h(u)\in\mathrm{Sub}(L)=L$, hence $h(u)\in L$, that is $u\in h^{-1}(L)$. In conclusion, $\mathrm{Sub}(h^{-1}(L))\subseteq$ $\subseteq h^{-1}(L)$, which shows that $\mathrm{Sub}(h^{-1}(L)=h^{-1}(L)$, hence $h^{-1}(L)\in\mathcal{FL}_{\mathrm{reg}}^{\cap}$ (Lemma 10, part (iii)). Similar arguments hold for $\mathcal{FL}_{\mathrm{rreg}}$, $\mathcal{FL}_{\mathrm{lreg}}$, $\mathcal{FL}_{\mathrm{reg}}^{\cup}$.

Consider now the language $L=(ab)^*c\cup c(ab)^*\cup(ab)^*$. It belongs to $\mathcal{FL}_{\mathrm{lin}}^{\cap}$, but $h^{-1}(L)=ab^*c$, for h defined by $h(a)=a$, $h(b)=ba$, $h(c)=bc$; this language is not in $\mathcal{FL}_{\mathrm{lin}}^{\cup}$, which implies nonclosure under inverse homomorphism for $\mathcal{FL}_{\mathrm{rlin}}$, $\mathcal{FL}_{\mathrm{llin}}$, $\mathcal{FL}_{\mathrm{lin}}^{\cup}$, $\mathcal{FL}_{\mathrm{lin}}^{\cap}$.

Mirror image. The closure cases follow from Lemma 11, the nonclosure ones are settled by examples of the form: $a^+b\in\mathcal{FL}_{\mathrm{rreg}}$, $\mathrm{Mi}(a^+b)=ba^+\notin\mathcal{FL}_{\mathrm{llin}}$.

Right quotient. We have $L=\{abc, ab, bc, a, b, c\}\in\mathcal{FL}_{\mathrm{reg}}^{\cap}$, but $L/\{c\}=\{ab, b\}\notin$ $\notin\mathcal{FL}_{\mathrm{lreg}}$, hence these families are not closed under right quotient. Similarly, $L=\{abc, ab, a\}\in\mathcal{FL}_{\mathrm{reg}}^{\cup}$, but $L/\{c\}=\{ab\}\notin\mathcal{FL}_{\mathrm{reg}}^{\cup}$. Similar languages can be constructed for $\mathcal{FL}_{\mathrm{lin}}^{\cap}$, $\mathcal{FL}_{\mathrm{llin}}$, $\mathcal{FL}_{\mathrm{lin}}^{\cup}$ (take $L=a^+bc\cup a^+b\cup bc\cup a^+\cup\{b, c\}\in\mathcal{FL}_{\mathrm{lin}}^{\cap}$, respectively, $L=ba^+bc\cup ba^+\in\mathcal{FL}_{\mathrm{lin}}^{\cup}$).

Consider now $L\in\mathcal{FL}_{\mathrm{rreg}}$ and an arbitrary language L'. According to Lemma 10, we have $L=\mathrm{Fin}(L)$. As L/L' is a regular language, we have only to prove that $\mathrm{Fin}(L/L')=L/L'$. Let $u\in\mathrm{Fin}(L/L')$ be an arbitrary string. There is a v such that $vu\in L/L'$, hence there is a $w\in L'$ such that $vuw\in L$. Therefore $uw\in\mathrm{Fin}\,(L)=L$, that is $u\in L/L'$. In conclusion, $\mathrm{Fin}(L/L')\subseteq L/L'$, hence $\mathrm{Fin}(L/L')=L/L'$, and $\mathcal{FL}_{\mathrm{rreg}}$ is closed under right quotient (with arbitrary languages).

Finally, consider a language $L\in\mathcal{FL}_{\mathrm{rlin}}$, $L=L_{\mathrm{in}}(G)$, $G=(V_N, V, P)$; let L' be an arbitrary language. For $X\in V_N$ set $L_X=L(G_X)$, $G_X=(V_N, V, X, P)$. We define the grammar $G'=(V_N, V, P')$ by

$$P' = \big(P-\{X\to\alpha|\alpha\in V^*,\ X\in V_N\}\big)\cup$$

$$\cup\{X\to\alpha|X\to\alpha\beta\in P,\ \text{for some } \alpha, \beta\in V^*,\ \beta\in L',\ X\in V_N\}$$

$$\cup\{X\to\alpha|X\to\alpha\beta Y\in P,\ \text{for some } \alpha, \beta\in V^*,\ X\in V_N,\ \{\beta\}L_Y\cap L'\neq\emptyset\}.$$

It is easy to see that $L_{\mathrm{in}}(G')=L/L'$, which completes the proof.

Left quotient. Simmetrically.

Init, Fin, Sub. Let $L \in \mathcal{FL}_{\text{rreg}}$; in view of Lemma 10, we have $\text{Fin}(L) = L$, $L \in \mathcal{L}_3$. Clearly, $\text{Init}(L)$, $\text{Fin}(L)$, $\text{Sub}(L)$ are regular languages. As $L = \text{Fin}(L)$, we have $\text{Fin}(\text{Init}(L)) = \text{Init}(\text{Fin}(L)) = \text{Init}(L)$, $\text{Fin}(\text{Fin}(L)) = \text{Fin}(L)$, $\text{Fin}(\text{Sub}(L)) = \text{Sub}(L)$. This implies that $\text{Init}(L)$, $\text{Fin}(L)$, $\text{Sub}(L)$ are in $\mathcal{FL}_{\text{rreg}}$, too. Similarly for $\mathcal{FL}_{\text{lreg}}$, hence also $\mathcal{FL}^{\cup}_{\text{reg}}$, $\mathcal{FL}^{\cap}_{\text{reg}}$ are closed. The family $\mathcal{FL}_{\text{rlin}}$ is closed under right quotient; as $\text{Init}(L) = L/V^*$, we obtain the closure under Init. Consider now $L \in \mathcal{FL}_{\text{rlin}}$, $L = L_{\text{in}}(G)$, $G = (V_N, V, P)$, and define the grammar $G' = (V'_N, V, P')$ by $V'_N = V_N \cup V''_N$, $P' = P \cup P''$, where, for each production r: $X \to a_1 a_2 \ldots a_n Y \in P$, $a_i \in V$, $1 \leqq i \leqq n$, $Y \in V_N \cup \{\lambda\}$, we introduce in P'' all productions $[X, r, j] \to a_{j+1} \ldots a_n Y$, $1 \leqq j \leqq n-1$, simultaneously introducing the new symbols $[X, r, j]$ in V''_N. Clearly, $L_{\text{in}}(G') = \text{Fin}\,(L)$, hence $\mathcal{FL}_{\text{rlin}}$ is closed under Fin. Now the closure under Sub follows from the closure under Init.

Similar arguments show that $\mathcal{FL}_{\text{llin}}$, hence also $\mathcal{FL}^{\cup}_{\text{lin}}$ and $\mathcal{FL}^{\cap}_{\text{lin}}$ are closed under Init, Fin, Sub.

Gsm mapping. $L = a^+$ is in $\mathcal{FL}^{\cup}_{\text{reg}}$; it is easy to construct a gms g such that $g(L) = ba^+b$. This language is not in $\mathcal{FL}_2$ (Corollary 1), hence none of the above families is closed under gsm mappings.

Inverse gsm mapping. Consider the gsm $g = (\{q_0, q_1, q_2\}, \{a, b\}, \{a\}, q_0, \{q_2\}, \{q_0 b \to a q_1, q_1 a \to a q_1, q_1 b \to a q_2\})$. We have $g^{-1}(a^+) = ba^*b \notin \mathcal{FL}_2$ (Corollary 1), hence none of the above families is closed under inverse gsm mappings.

Intersection with regular sets. As $V^* \in \mathcal{FL}^{\cap}_{\text{reg}}$, for each V, but $\mathcal{L}_3 - \mathcal{FL}^{\cup}_{\text{lin}} \neq \emptyset$, the assertion is obvious.

6. Further questions

In the proof of inclusions $\mathcal{FL}_i \subseteqq \mathcal{L}_i$, $i = 0, 1, 2, 3$, in Theorem 1, starting from the grammar G, used in fully initial manner, we constructed a grammar G' such that $\text{Prod}(G') = \text{Prod}(G) + \text{Var}(G)$. (For an arbitrary grammar $G = (V_N, V_T, S, P)$ we denote, as in [5], $\text{Prod}(G) = \text{card}\,P$, $\text{Var}(G) = \text{card}\,V_N$.) Can the difference between $\text{Prod}(G')$ and $\text{Prod}(G)$ be diminished? More generally, given a language $L \in \mathcal{FL}_i$, define

$$\text{Prod}(L) = \inf\{\text{Prod}(G) | L = L(G)\},$$

$$\text{Prod}_{\text{in}}(L) = \inf\{\text{Prod}(G) | L = L_{\text{in}}(G)\}.$$

What is the relation between $\text{Prod}(L)$ and $\text{Prod}_{\text{in}}(L)$? The construction in the proof of Theorem 1 (used also in [3]) shows that $\text{Prod}(L) \leqq \text{Prod}_{\text{in}}(L) + \text{Var}_{\text{in}}(L)$. We shall prove that this relation cannot be essentially improved (which shows that, in some sense, the fully initial mode of generating a language is more economical than the usual mode, at least for certain languages).

Indeed, consider the context-free grammar $G = (\{A_1, A_2, \ldots, A_n\}, \{a_1, a_2, \ldots, a_n, b\}, P)$ with

$$P = \{A_i \to a_i A_i a_i | 1 \leqq i \leqq n\} \cup$$

$$\cup \{A_i \to a_i A_{i+1} a_i | 1 \leqq i \leqq n-1\} \cup \{A_n \to a_n b a_n\}.$$

We have

$$L_{\text{in}}(G) = \{a_i^{k_i}a_{i+1}^{k_{i+1}} \dots a_n^{k_n}ba_n^{k_n} \dots a_{i+1}^{k_{i+1}}a_i^{k_i} | 1 \leqq i \leqq n,\ k_j \geqq 1,\ 1 \leqq j \leqq n\}.$$

Consequently, $\text{Prod}_{\text{in}}(L_{\text{in}}(G)) \leqq 2n$, $\text{Var}_{\text{in}}(L_{\text{in}}(G)) \leqq n$. It is easy to see that, in fact, we have $\text{Var}_{\text{in}}(L_{\text{in}}(G)) = n$ (for each i we need a derivation $X_i \overset{*}{\Rightarrow} a_i^j X_i a_i^j$, $j \geqq 1$), hence also $\text{Prod}_{\text{in}}(L_{\text{in}}(G)) = 2n$.

Consider now a usual context-free grammar $G' = (V_N, V_T, S, P')$ such that $L(G') = L_{\text{in}}(G)$. Again, for each i, $1 \leqq i \leqq n$, we need a derivation $X_i \overset{*}{\Rightarrow} a_i^j X_i a_i^j$, $j \geqq 1$, one of the form $X_i \overset{*}{\Rightarrow} a_i^j a_{i+1}^k X_{i+1} a_{i+1}^m a_i^p$, $j, k, m, p \geqq 0$, as well as one of the form $S \overset{*}{\Rightarrow} a_i^j X_i a_i^k$, $j, k \geqq 0$. Two symbols X_i, X_j cannot be identical when $i \neq j$ (otherwise strings containing both substrings $a_i a_j$, $a_j a_i$ on the same side of b could be obtained). Moreover, the axiom S must differ from every X_i, $i \geqq 2$. In conclusion, $\text{Prod}\,(G') \geqq 3n-1 = \text{Prod}(G) + \text{Var}(G) - 1$, therefore $\text{Prod}\,(L_{\text{in}}(G)) \geqq \text{Prod}_{\text{in}}(L_{\text{in}}(G)) + \text{Var}_{\text{in}}(L_{\text{in}}(G)) - 1$.

Consider now another question. Given a language L and a grammar G for it, $L = L(G)$, what one can say about $L_{\text{in}}(G)$? For example, taking $L = \{a^n b^n | n \geqq 1\} \cdot \{a, b\}^*$ and the grammar $G = (\{S, A, B\}, \{a, b\}, S, \{S \to AB, A \to aAb, A \to ab, B \to aB, B \to bB, B \to \lambda\})$ we obtain $L(G) = L \in \mathcal{L}_2 - \mathcal{L}_3$, $L_{\text{in}}(G) = \{a, b\}^* \in \mathcal{L}_3$.

Are there languages L for which this is not possible (no grammar G, $L = L(G)$ with $L_{\text{in}}(G)$ regular)? The answer is affirmative: take $L = \{a^n b^n | n \geqq 1\}$ and consider a context-free grammar $G = (V_N, \{a, b\}, S, P)$ such that $L = L(G)$ and G is reduced. Clearly, each recursive derivation $X \overset{*}{\Rightarrow} \alpha X \beta$, $\alpha, \beta \in \{a, b\}^*$ must have $\alpha = a^i$, $\beta = b^i$, $i \geqq 1$. For each symbol $A \in V_N$, consider the set $L_A = \{w \in \{a, b\}^* | A \overset{*}{\Rightarrow} w \text{ in } G\}$. If L_A is finite for some A, then, replacing each occurrence of A in the right-hand sides of rules in P by a string in L_A (and removing all rules $A \to \gamma$), we obtain a grammar G', $L(G) = L(G')$, $L_{\text{in}}(G) - L_{\text{in}}(G')$ is finite. The grammar G' obtained in this way be removing all $A \in V_N$ with finite L_A is linear. (If rule $X \to x_1 Y x_2 Z x_3$ is in G', then L_Y, L_Z must be infinite, hence must involve recursive derivations in the generation of their strings, hence L_X contains strings of the form $z_1 a^i b^i z_2 a^j b^j z$, i, $j \geqq 1$, a contradiction.) If $L_{\text{in}}(G)$ is regular, then $L_{\text{in}}(G')$ is regular too (it differs from $L_{\text{in}}(G)$ by a finite set). However, each derivation in G', besides its maximal recursive subderivations, contains at most card V_N further steps. These steps introduce at most $\pi = \text{card}\, V_N \cdot \max\{|x|\,|A \to x \in P\}$ occurrences of a and of b. In conclusion, each string in $L_{\text{in}}(G')$ is of the form $a^{n+p}b^{n+q}$, $n \geqq 1$, $p \leqq \pi$, $q \leqq \pi$. This implies $L_{\text{in}}(G') \notin \mathcal{L}_3$, a contradiction.

A further situation which can be looked for is the following. Are there languages $L \in \mathcal{L}_2 - \mathcal{L}_3$ such that each context-free grammar $G, L = L(G)$, has $L_{\text{in}}(G) \in \mathcal{L}_3$? (Such a language can be called *inherently fully initial regular*, whereas the above $L = \{a^n b^n | n \geqq 1\}$ can be called *inherently fully initial context-free*.) This last problem remains *open*.

References

[1] ADJ. — J. A. GOGUEN, J. W. THATCHER, E. G. WAGNER, J. B. WRIGHT, Initial algebra semantics and continuous algebra, *Journal of ACM*, 24, 1 (1977), 68—95.
[2] J. ALBERT, L. WEGNER, Languages with homomorphic replacements, *Lect. Notes Computer Sci.*, 85, Automata, Languages and Programming, Ed. J. W. de Bakker, J. van Leeuwen, Springer-Verlag, 1980, 19—29.
[3] J. DASSOW, On fully initial context-free languages, *Papers on Automata and Languages* (Ed. I. Peák), X (1988), 3—6.
[4] S. GINSBURG, *The mathematical theory of context-free languages*, McGraw Hill Book Comp., New York, 1966.
[5] J. GRUSKA, Descriptional complexity of context-free languages, *Proc. Symp. Math. Found. Computer Sci.*, High Tatras, 1973.
[6] G. T. HERMAN, A biologically motivated extension of ALGOL-like languages, *Inform. Control*, 22, 5 (1973), 487—502.
[7] H. A. MAURER, A. SALOMAA, D. WOOD, Pure grammars, *Inform. Control*, 49 (1980), 47—72.
[8] GH. PĂUN, *Contextual grammars*, The Publ. House of the Romanian Acad. of Sci., Bucureşti, 1982 (in Roumanian).
[9] GH. PĂUN, A note on fully initial context-free languages, *Papers on Automata and Languages* (Ed. I. Peák), X (1988), 7—11.
[10] A. SALOMAA, *Formal languages*, Academic Press, New York, London, 1973.
[11] A. VAN WIJNGAARDEN, *Orthogonal design and description of formal languages*, Math. Centrum, Amsterdam, 1965.

(Received June 21, 1988)

On fully initial grammars with regulated rewriting

T. Bălănescu, M. Gheorghe, Gh. Păun

The Computer Centre of the University of Bucharest, Str. Academiei 14, București, 70109 ROMANIA

We investigate the fully initial version of context-free grammars added with various control devices: regular control, matrices, programming, random context, Indian parallelism and ordering, each of them with or without λ-rules and (when appropriate) appearance checking. It is shown that the fully initial feature decreases the generative power of programmed, random context λ-free grammars with or without appearance checking, and of ordered and Indian parallel ones. In all remaining cases the generative capacity is not modified. On the other hand, regulated rewriting increases the generative capacity of fully initial context-free grammars.

1. Definitions and notations

The fully initial (fi, for short) variant of context-free grammars was defined by S. Horváth and investigated in [2], [3]. Such a grammar is a usual context-free grammar (cfg, for short) having no distinguished start symbol. The language generated in this way by a grammar $G=(V_N, V_T, P)$ is $L(G)=\{x\in V_T^*|A\overset{*}{\Rightarrow} x$ for some $A\in V_N\}$. (As usual, V_N is the nonterminal vocabulary, V_T is the terminal vocabulary and P is the set of rewriting rules; V^* denotes the free monoid generated by V under the operation of concatenation and λ is the null element.) Inclusion and strict inclusion are denoted by $\subseteq$ and $\subset$, respectively.

Similar to regulated rewriting for context-free grammars [1], [4], we consider here the languages generated by fi regular control, matrix, programmed, random context, Indian parallel and ordered cfg's. We give only informal definitions and refer to [1], [4] for details.

Given a grammar G as above, Lab (P) denotes the set of labels of rules in G (each rule has a distinct label).

A fi regular control (fic, for short) grammar $G=(V_N, V_T, P, K, F)$ consists of a fi cfg (V_N, V_T, P), a regular control language K over Lab (P) and a set F of labels. We write $A \stackrel{*}{\Rightarrow} y$ in G if there exists a string $p_1 p_2 \ldots p_n \in K$, $p_i \in \text{Lab}(P)$, such that $A=x_0 \underset{p_1}{\Rightarrow} x_1 \ldots \underset{p_n}{\Rightarrow} x_n=y$, and for each i we have either $x_{i-1} \underset{p_i}{\Rightarrow} x_i$ or $x_{i-1}=x_i$, the rule p_i is not applicable to x_{i-1} and $p_i \in F$.

A fi matrix (fim, for short) grammar $G=(V_N, V_T, P, M, F)$ consists of a cfg (V_N, V_T, P), a finite set M of matrices and a finite set F of occurrences of productions in matrices of M. A matrix is a sequence $m=(A_1 \to u_1, \ldots, A_n \to u_n)$, $n \geqq 1$, of productions in P. We write $x \underset{m}{\Rightarrow} y$ for a matrix m as above if there are $x_1=x$, $x_2, \ldots$ $\ldots, x_n=y$ such that either $x_j=x_{j+1}$, the rule r_j: $A_j \to u_j$ is in F and it is not applicable to x_j or $x_j \underset{r_j}{\Rightarrow} x_{j+1}$.

In a programmed (fip, for short) grammar $G=(V_N, V_T, P)$ the rules are of the form $(b\colon A \to u, S(b), F(b))$, where b is the label of the production, $S(b)$ and $F(b)$ are sets of labels referred to as the success and the failure field. If $A \to u$ is applicable to a string x, then, after applying it, we continue the derivation with a rule having the label in $S(b)$; if $A \to u$ is not applicable to x, then we pass to a rule with its label in $F(b)$ (the string x remains unchanged).

A fi random context (firc, for short) grammar $G=(V_N, V_T, P)$ has the rules of the form $(A \to u, Q, R)$, where Q, R are subsets of V_N, referred to as permitting and forbidding sets of symbols, respectively. Such a rule is applicable to a string x iff x contains all nonterminals of Q and contains no nonterminal in R.

A fi Indian parallel (fiip, for short) grammar is a cfg grammar in which each rule $A \to w$ is used in a derivation $u \Rightarrow v$ for rewriting all occurrences of A in w, thus obtaining v.

A fi ordered (fio, for short) grammar $(G, >)$ consists of a fi cfg G and a partial order $>$ on P. A rule $A \to u$ is applicable to a string x iff no rule $B \to v$ is applicable to x and $B \to v > A \to u$.

We denote by FI_λ, $\text{FIC}_{ac,\lambda}$, $\text{FIM}_{ac,\lambda}$, $\text{FIP}_{ac,\lambda}$, $\text{FIRC}_{ac,\lambda}$, FIIP_λ, and FIO_λ the families of languages generated by fi, fic, fim, fip, firc, fiip and fio grammars, respectively. The corresponding families generated in the usual mode are denoted by $\text{C}_{ac,\lambda}$, $\text{M}_{ac,\lambda}$, $\text{P}_{ac,\lambda}$, $\text{RC}_{ac,\lambda}$, IP_λ, O_λ, respectively. When the appearance checking feature is not present, that is when $F=\emptyset$ for fic and fim, $F(b)=\emptyset$ for fip and $R=\emptyset$ for firc grammars, we erase the subscript ac; when no λ-rules are allowed we erase also the subscript λ. As usual, the families of recursively enumerable, context sensitive, context-free and regular languages are denoted by $\mathcal{L}_0$, $\mathcal{L}_1$, $\mathcal{L}_2$, $\mathcal{L}_3$, respectively.

Two languages are identified if they differ by at most the empty string.

2. The generative capacity of fully initial regulated grammars

Lemma 1. $\text{FIC}_{ac,\lambda}=\text{C}_{ac,\lambda}$, $\text{FIC}_\lambda = \text{C}_\lambda$, $\text{FIC}_{ac}=\text{C}_{ac}$, $\text{FIC}=\text{C}$.

Proof. Let $G=(V_N, V_T, S, P, K, F)$ be a regular control grammar. We consider the fic grammar $G'=(V_N, V_T, P, K', F)$, where $K'=K \cap I \cdot \text{Lab}(P)^*$, I being the set of labels of rules of the form $S \to u$ in P. Clearly, $L(G)=L(G')$ and G' is of the same type as G.

Conversely, for a fic grammar $G=(V_N, V_T, P, K, F)$ we consider the regular control grammar $G'=(V_N\cup\{S\}, V_T, S, P', K', F)$, where S is a new nonterminal, $P'=P\cup\{S\to A|A\in V_N\}$, $K'=I\cdot K$ and I is the set of labels of rules $S\to A$, $A\in V_N$. Obviously, $L(G)=L(G')$.

Lemma 2. $\mathrm{FIM}_{ac,\lambda}=\mathrm{M}_{ac,\lambda}$, $\mathrm{FIM}_{\lambda}=\mathrm{M}_{\lambda}$, $\mathrm{FIM}_{ac}=\mathrm{M}_{ac}$, $\mathrm{FIM}=\mathrm{M}$.

Proof. Let $G=(V_N, V_T, P, M, F)$ be a fim grammar. We construct the grammar $G'=(V_N\cup\{S\}, V_T, S, P, M', F)$, where S is a new symbol and $M'=M\cup\{(S\to A)| A\in V_N\}$. Clearly, $L(G)=L(G')$, hence we have the inclusions $\subseteqq$.

Conversely, let $L\subseteqq V^*$ be a matrix language in a family $\mathrm{M}_{\alpha,\beta}$, $\alpha=ac$ or it is empty, $\beta=\lambda$ or it is empty. We write

$$L = \bigcup_{a\in V} \{a\}\partial_a(L)\cup\{x\in L|\,|x|\leqq 1\}$$

($\partial_a(L)$ is the left derivative of L with respect to a). Each language $\partial_a(L)$ is a matrix language of the same type as L; let $G_a=(V_{N,a}, V, S_a, P_a, M_a, F_a)$ be a matrix grammar for each. Without loss of generality we may suppose that the vocabularies $V_{N,a}$ are pairwise disjoint and that each M_a contains matrices $m=(r_1, \ldots, r_n)$ with at least one occurrence of productions not in F_a (otherwise we remove m and the corresponding occurrences of rules from M_a and F_a, respectively, and we introduce all matrices $m_i=(r_1, \ldots, r_n)$, $1\leqq i\leqq n$, containing the same rules as m but with the rule occurring on the position i not in F_a).

A fim grammar generating L is $G'=(V_N', V, P', M', F')$, where

$$V_N' = \bigcup_{a\in V} (V_{N,a}\cup\{[\alpha]\})\cup\{S\}, \quad S \text{ is a new symbol,}$$

$$P' = \bigcup_{a\in V} (P_a\cup\{S\to[a]S_a, [a]\to[a], [a]\to a\})\cup\{S\to x|x\in L,\ |x|\leqq 1\},$$

and M' is constructed as follows:

a) $(S\to x)$, $x\in L$, $|x|\leqq 1$, is in M',

b) for each $a\in V$ we introduce in M' the matrices

b.1) $(S\to[a]S_a)$,

b.2) $([a]\to[a],\ r_1, \ldots, r_n)$, for $(r_1, \ldots, r_n)\in M_a$,

b.3) $([a]\to a,\ r_1, \ldots, r_n)$, for $(r_1, \ldots, r_n)\in M_a$.

Finally, $F'=\bigcup_{a\in V} F_a$.

It is easy to see that in each derivation of a string $x\in L$, $|x|>1$, all sentential forms are of the form $[a]w$; moreover, no derivation can start from a symbol different from S (remember that for all $a\in V$, each matrix in M_a contains a rule not in F_a). In conclusion, $L(G')=L$, hence $\mathrm{M}_{\alpha,\beta}\subseteqq\mathrm{FIM}_{\alpha,\beta}$, α, β as above.

Lemma 3. $\mathrm{FIP}_{ac,\lambda}=\mathrm{P}_{ac,\lambda}$, $\mathrm{FIP}_{\lambda}=\mathrm{P}_{\lambda}$, $\mathrm{FIP}_{ac}\subseteqq\mathrm{P}_{ac}$, $\mathrm{FIP}\subseteqq\mathrm{P}$.

Proof. Let $G=(V_N, V_T, P)$ be a fip grammar and consider the programmed grammar $G'=(V_N\cup\{S\}, V_T, S, P')$, where S is a new symbol and $P'=P\cup\{(r_A\colon S\to A, \mathrm{Lab}(P), \emptyset)|A\in V_N\}$. We have $L(G)=L(G')$, hence $\mathrm{FIP}_{\alpha,\beta}\subseteqq\mathrm{P}_{\alpha,\beta}$, $\alpha=ac$ or it is empty, $\beta=\lambda$ or it is empty.

Conversely, let $G=(V_N, V_T, S, P)$ be a programmed grammar. We construct the fip grammar $G'=(V'_N, V_T, P)$, where
$V'_N=V_N\cup\{X, Y, N\}$, X, Y, N are new symbols, and P' contains the next rules:

a) $(s\colon X\to SY,\ S(s),\ \emptyset)$, $s\notin \mathrm{Lab}(P)$, $S(s)=\{i|(i\colon S\to u,\ S(i),\ F(i))\in P\}$,

b) $(r\colon A\to uN,\ S(r)\cup\{f\},\ F(r))$, $f\notin \mathrm{Lab}(P)$ and

$(r\colon A\to u,\ S(r),\ F(r))\in P$,

c) $(f\colon Y\to\lambda, \{f_N\}, \emptyset)$,

d) $(f_N\colon N\to\lambda, \{f_N\}, \emptyset)$, $f_N\notin \mathrm{Lab}(P)$.

It is easy to see that the symbol N cannot be erased without erasing first symbol Y. Therefore, no rule in group b) can be successfully used without starting the derivation by the rule of type a). In consequence, $L(G)=L(G')$, hence $P_{\alpha,\lambda}\subseteq \mathrm{FIP}_{\alpha,\lambda}$, where α is as above.

Lemma 4. $\mathrm{FIRC}_{ac,\lambda}=\mathrm{RC}_{ac,\lambda}$, $\mathrm{FIRC}_\lambda=\mathrm{RC}_\lambda$, $\mathrm{FIRC}_{ac}\subseteq R_{ac}$, $\mathrm{FIRC}\subseteq\mathrm{RC}$.

Proof. Given a firc grammar $G=(V_N, V_T, P)$, we construct the random context grammar $G'=(V_N\cup\{S\}, V_T, S, P')$, where S is a new symbol and $P'=P\cup\{(S\to A, \emptyset, \emptyset)|A\in V_N\}$. We have $L(G)=L(G')$, hence $\mathrm{FIRC}_{\alpha,\beta}\subseteq\mathrm{RC}_{\alpha,\beta}$, $\alpha=ac$ or it is empty, $\beta=\lambda$ or it is empty.

Conversely, for a random context grammar $G=(V_N, V_T, S, P)$, we construct the firc grammar $G'=(V_N\cup\{X, Y\}, V_T, P')$, where X, Y are new symbols and P' contains the following rules:

a) $(X\to SY, \emptyset, \emptyset)$,

b) $(Y\to\lambda, \emptyset, \emptyset)$,

c) $(A\to u,\ Q\cup\{Y\},\ R)$, for $(A\to u, Q, R)\in P$.

Obviously, $L(G)=L(G')$, which completes the proof.

Lemma 5. $\mathscr{L}_3$ — $\mathrm{FIO}_\lambda\neq\emptyset$.

Proof. Let us consider the regular language $L=\{ab^na|n\geqq 0\}$ and suppose that L is generated by the fio grammar $(G, >)$, $G=(V_N, \{a, b\}, P)$. Define $k=\max\{|u|\ |A\to u\in P\}$ and consider a derivation $A=u_0\Rightarrow u_1\Rightarrow\ldots\Rightarrow u_p=ab^ka$ in $(G, >)$, $A\in V_N$. As $|ab^ka|>k$, we have $p\geqq 2$. Let i be the greatest index such that $u_i=u'_iBu''_i$ and $u'_i\overset{*}{\Rightarrow}\lambda$, $u''_i\overset{*}{\Rightarrow}\lambda$ and $B\overset{*}{\Rightarrow}ab^ka$ in $(G, >)$. It follows that $B\Rightarrow uCv\overset{*}{\Rightarrow}ab^ka$, $ab^ka=$ $=xyz$, $u\overset{*}{\Rightarrow}x$, $C\overset{*}{\Rightarrow}y$, $v\overset{*}{\Rightarrow}z$ and $y\neq\lambda$. Clearly, $y\neq ab^ka$, hence $y\in L(G, >)$ and y is a proper subword of ab^ka, contradiction.

Corollary. $\mathrm{FIO}_\lambda\subset\mathrm{O}_\lambda$, and $\mathrm{FIO}\subset\mathrm{O}$.

Lemma 6. $\mathscr{L}_3-(\mathrm{FIP}_{ac}\cup\mathrm{FIRC}_{ac})\neq\emptyset$.

Proof. Let us consider the language $L=\{ab^na|n\geqq 0\}$ as above and suppose that L is generated by a fip (firc) grammar $G=(V_N, V_T, P)$ without λ-rules. Let $k=$ $=\max\{|u||A\to u\in P\}$ and take $x=ab^ka\in L(G)$. There exists a derivation $A\Rightarrow x_1\Rightarrow\ldots$

$\ldots \Rightarrow x_n = ab^k a$, $A \in V_N$. The lastly used rule is $B \to u$, with $u = ab^t$, or $u = b^q a$, or $u = b^s$, $0 \leqq t$, $q < k$, $1 \leqq s \leqq k$. It follows that $u \in L(G)$, a contradiction.

Corollary. (i) $\mathscr{L}_2 - (\mathrm{FIP}_{ac} \cup \mathrm{FIRC}_{ac}) \neq \emptyset$, (ii) $\mathrm{FIP}_{ac} \subset \mathrm{P}_{ac}$, $\mathrm{FIP} \subset \mathrm{P}$, $\mathrm{FIRC}_{ac} \subset \mathrm{RC}_{ac}$, $\mathrm{FIRC} \subset \mathrm{RC}$.

Lemma 7. Let L be a language over a vocabulary V and let c be a symbol not in V. a) If $L \in \mathrm{P}_{ac}$ ($L \in \mathrm{P}$), then $L\{c\} \cup V \cup \{c\} \in \mathrm{FIP}_{ac}$ (FIP, respectively). b) If $L \in \mathrm{RC}_{ac}$ ($L \in \mathrm{RC}$), then $L\{c\} \cup \{c\} \in \mathrm{FIRC}_{ac}$ (FIRC, respectively).

Proof. a) For a programmed λ-free grammar $G = (V_N, V, S, P)$ generating L, we construct the fip grammar $G' = (V'_N, V \cup \{c\}, P')$ with $V'_N = V_N \cup \{a' \mid a \in V\} \cup \{X, Y\}$ where X, Y are new symbols, and with P' containing the next productions:

a) $(s\colon X \to SY, S(s), \emptyset)$, with $s \notin \mathrm{Lab}(P)$, $S(s) = \{i \mid (i\colon S \to u, S(i), F(i)) \in P\}$

b) $(r\colon A \to u', S(r) \cup \{f\}, F(r))$, for each $(r\colon A \to u, S(r), F(r)) \in P$; $f \notin \mathrm{Lab}(P)$ and u' is obtained from u by replacing each $a \in V$ by $a' \in V'_N$ in u,

c) $(f\colon Y \to c, \{f_a \mid a \in V\}, \emptyset)$,

d) $(f_a\colon a' \to a, \{f_b \mid b \in V\}, \emptyset)$, for all $a \in V$; $f_a \notin \mathrm{Lab}(P)$.

The equality $L(G') = L\{c\} \cup V \cup \{c\}$ is obvious, hence we have proved the first part of the lemma.

b) If $G = (V_N, V, S, P)$ is a random context grammar generating L, then we construct the firc grammar $G' = (V'_N, V \cup \{c\}, P')$, where

$$V'_N = V_N \cup \{X, Y\}, \quad \text{with new symbols } X \text{ and } Y,$$

$$P' = \{(X \to SY, \emptyset, \emptyset), (Y \to c, \emptyset, \emptyset)\} \cup$$

$$\cup \{(A \to u, Q \cup \{Y\}, R) \mid (A \to u, Q, R) \in P\}.$$

We obviously have $L(G') = L\{c\} \cup \{c\}$, which completes the proof.

Corollary 1. $\mathrm{FIP} - \mathscr{L}_2 \neq \emptyset$, $\mathrm{FIRC} - \mathscr{L}_2 \neq \emptyset$.

Proof. Follows from $\mathrm{P} - \mathscr{L}_2 \neq \emptyset$, $\mathrm{RC} - \mathscr{L}_2 \neq \emptyset$, the above lemma and the closure properties of $\mathscr{L}_2$.

Corollary 2. $\mathrm{FIRC} - \mathrm{FIP}_{ac} \neq \emptyset$.

Proof. The language $L = \{ab^n a \mid n \geqq 0\}\{c\} \cup \{c\}$ is in FIRC, but not in FIP_{ac} (this follows as in the proof of Lemma 6).

Lemma 8. $\mathrm{FIP} - \mathrm{FIO}_\lambda \neq \emptyset$.

Proof. The language $L = \{ab^n ac \mid n \geqq 0\} \cup \{a, b, c\} \in \mathrm{FIP} - \mathrm{FIO}_\lambda$. The relation $L \in \mathrm{FIP}$ follows from Lemma 7, and $L \notin \mathrm{FIO}_\lambda$ can be proved as in the proof of Lemma 5.

Corollary. $\mathrm{FIRC} - \mathrm{FIO}_\lambda \neq \emptyset$, $\mathrm{FIP}_{ac} - \mathrm{FIO} \neq \emptyset$.

Lemma 9. $\mathrm{FIO} \subset \mathrm{FIP}_{ac}$.

Proof. Let $(G, >)$, $G = (V_N, V_T, P)$, be a fio grammar. Without loss of generality we may assume that whenever $A \to u$ and $A \to v$ are both in P, then these rules are incomparable. We construct the fip grammar $G' = (V_N \cup \{X\}, V_T, P')$, where X is a new symbol and P' is constructed as follows. For any rule $r\colon A \to u \in P$ write $g(r) = \{A_1 \to u_1, \ldots, A_n \to u_n\}$, where $A_i \to u_i > A \to u$, $1 \leqq i \leqq n$.

For every rule r: $A \to u$ in P, introduce in P' all the rules $(r^{(i)}: A_i \to u_i X, \emptyset, \{r^{(i+1)}\})$, $1 \leq i \leq n-1$, as well as the rule $(r^{(n)}: A_n \to u_n X, \emptyset, \{r'\})$; then, add also to P' the rule $(r': A \to u, E, \emptyset)$, with $E = \{p^{(1)} | p: B \to v \in P, g(p) \neq \emptyset\} \cup \{p' | p: B \to v \in P, g(p) = \emptyset\}$.

A derivation in G' develops as follows: the use of a rule $(r': A \to u, E, \emptyset)$ is preceeded by the application with appearance checking of all the rules $r^{(i)}$, $1 \leq i \leq \text{card}(g(r))$; if such a rule $r^{(i)}$ can be applied, then the derivation is blocked. Therefore $L(G, >) = L(G')$, hence $\text{FIO} \subseteq \text{FIP}_{ac}$. The proper inclusion follows from the corollary to Lemma 8.

Lemma 10. $\text{FIP}_{ac} \subset \text{FIRC}_{ac}$.

Proof. Let $G = (V_N, V_T, P)$ be a fip grammar. We construct the firc grammar $G' = (V'_N, V_T, P')$, where

$$V'_N = \{[A, r] | A \in V_N, r \in \text{Lab}(P)\} \cup \{(u, r) | (r: A \to u, S(r), F(r)) \in P\}$$

and, for every rule $(r: A \to u, S(r), F(r)) \in P$, the set P' contains the following random context rules:

a) $([A, r] \to (u, r), \emptyset, C_r)$, for any $s \in S(r)$,

b) $([B, r] \to [B, s], \{(u, s)\}, C_{r,s} - \{(u, s)\})$, for any $s \in S(r)$ and $B \in V_N$,

c) $((u, s) \to [u, s], \emptyset, C_s - \{(u, s)\})$, for any $s \in S(r)$,

d) $([B, r] \to [B, f], \emptyset, C_{r,f} \cup \{[A, r]\})$, for any $f \in F(r)$, $B \in V_N$ and $B \neq A$,

with $C_r = V'_N - \{[X, r]\} | X \in V_N\}$, $C_{r,s} = C_r - \{[X, s] | X \in V_N\}$ and if $u = x_1 A_1 x_2 \ldots x_n A_n x_{n+1}$, $x_i \in V_T^*$, $A_i \in V_N$, $n \geq 0$, then $[u, s] = x_1 [A_1, s] \ldots x_n [A_n, s] x_{n+1}$.

An arbitrary derivation $v \underset{r}{\Rightarrow} w$ in G is simulated in G' as follows. If r is not applicable to v, then simply apply the rules of the form d) and continue according to the failure field $F(r)$. Otherwise, a rule of the form a) is applied, provided all nonterminals are marked with the label r. The new by introduced nonterminal, (u, s), enables us to continue the derivation according to the success field $S(r)$; it assists the application of the rules of the form b) until all nonterminals are marked by s. Next, the rewriting of A by u is simply accomplished by a rule of the form c); note that all nonterminals of the sentential form must be marked by s. The process continues with the rules derived from the rule $s \in S(r)$. Obviously, $L(G) \subseteq L(G')$. Similarly, each derivation in G' corresponds to one in G, hence $L(G') \subseteq L(G)$, hence $\text{FIP}_{ac} \subseteq \text{FIRC}_{ac}$. The inclusion is proper, as it follows from Corollary 2 of Lemma 7.

Let us investigate now the Indian parallel grammars.

Similarly to the equality $\text{IP} = \text{IP}_\lambda$, we also have $\text{FIIP} = \text{FIIP}_\lambda$.

Lemma 11. $\text{FIIP} \subset \text{IP}$ and $\text{FIIP} \subset \text{FIP}_{ac}$.

Proof. If $G = (V_N, V_T, P)$ is a fiip grammar, we construct $G' = (V_N \cup \{S\}, V_T, S, P')$, S a new symbol, $P' = P \cup \{S \to A | A \in V_N\}$, for proving $\text{FIIP} \subseteq \text{IP}$, and

$$G'' = (V_N \cup \{A' | A \in V_N\}, V_T, P''),$$

$$P'' = \{(r: A \to x_A, \{r\}, \{r_A\}) | r: A \to x \in P\} \cup \{(r_A: A' \to A, \{r_A\}, \text{Lab}(P)) | A \in V_N\},$$

for proving $\mathrm{FIIP} \subseteq \mathrm{FIP}_{ac}$ (x_A is the string obtained by replacing each occurrence of A in x by A').

As $\{ab^n a \mid n \geqq 0\} \in \mathrm{IP} - \mathrm{FIP}_{ac}$ and the Dyck language over $\{a, b\}$ is in FI (it is generated by the cfg $(\{S\}, \{a, b\}, \{S \to SS, S \to \lambda, S \to aSb\})$), but not in IP, both inclusions above are proper.

Corollary 1. $\mathscr{L} - \mathrm{FIIP} \neq \emptyset$ for all families $\mathscr{L} \in \{\mathrm{FI}, \mathscr{L}_2, \mathrm{FIO}\}$.

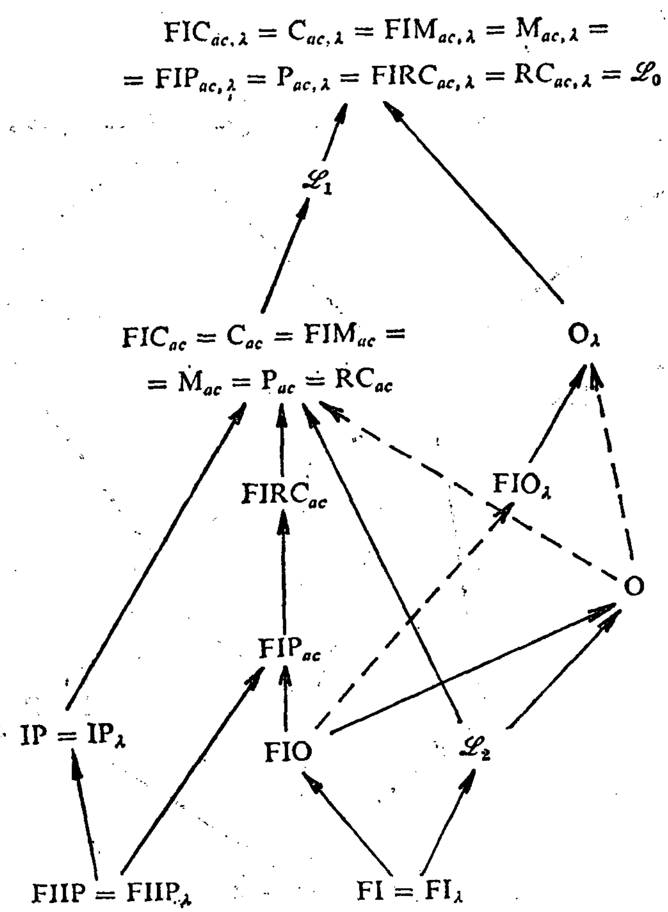
Corollary 2. IP is incomparable with all families FI, $\mathscr{L}_2$, FIO, FIP, FIP_{ac}, FIRC, FIRC_{ac}.

Lemma 12. $\mathrm{FIIP} - \mathscr{L}_2 \neq \emptyset$, $\mathrm{FIIP} - \mathrm{FI} \neq \emptyset$.

Proof. $\mathrm{L} = \{a^{2^n} \mid n \geqq 0\}$ is in FIIP as it is generated by the grammar $(\{S\}, \{a\}, \{S \to SS, S \to a\})$, but it is not context-free, hence nor is it in FI.

Summarizing the results in the previous lemmas, we obtain:

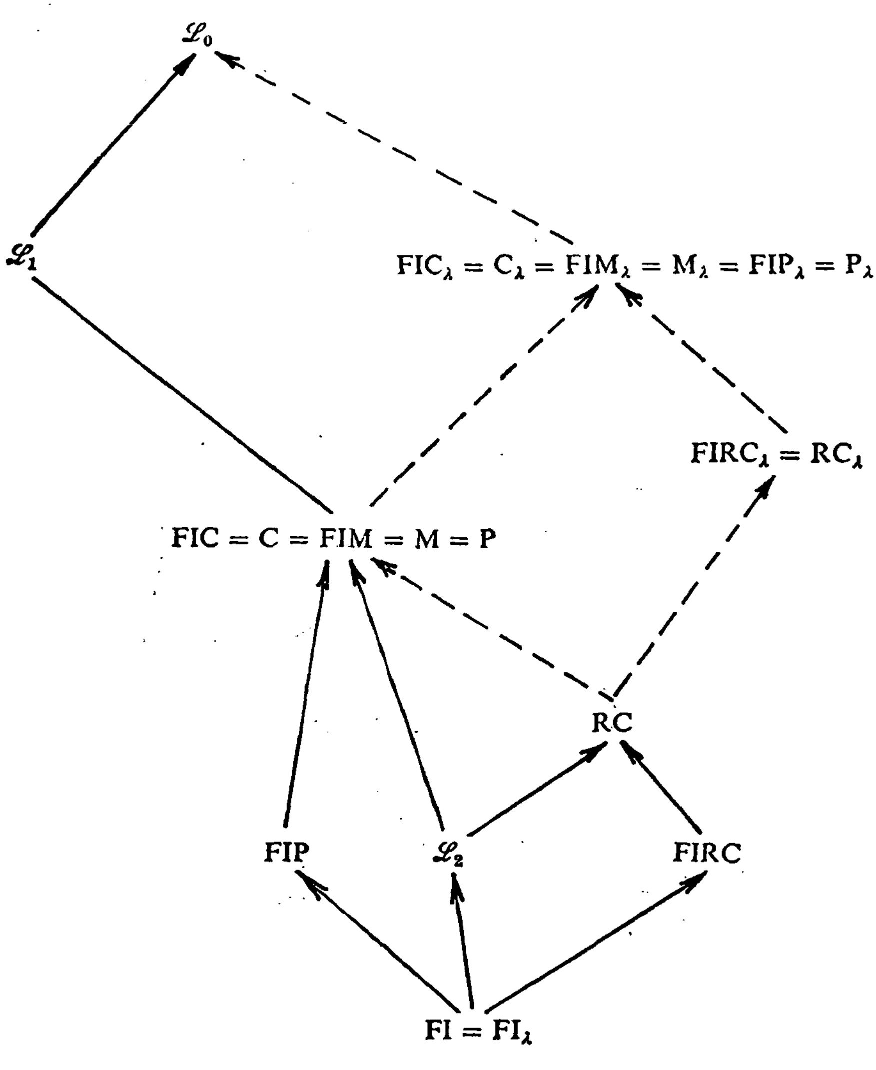
Theorem 1. The following diagram holds:



where ⟶ indicates strict inclusion and ⇢ points out an inclusion which is not known to be strict.

Theorem 2. a) The families in the next pairs are incomparable: ($\mathscr{L}_2$, FIO), ($\mathscr{L}_2$, FIO_λ), ($\mathscr{L}_2$, FIP_{ac}), ($\mathscr{L}_2$, FIRC_{ac}), ($\mathscr{L}_2$, FIIP), (FI, FIIP), (IP, FI), (IP, FIO), (IP, FIP), (IP, FIP_{ac}), (IP, FIRC), (IP, FIRC_{ac}), (IP, $\mathscr{L}_2$). b) The following relations hold: $\mathrm{FIP}_{ac}-\mathrm{FIO}_\lambda\neq\emptyset$, $\mathrm{FIRC}_{ac}-\mathrm{FIO}_\lambda\neq\emptyset$, $\mathscr{L}_3-\mathrm{FIRC}_{ac}\neq\emptyset$, $\mathscr{L}_3-\mathrm{FIP}_{ac}\neq\emptyset$, $\mathscr{L}_3-\mathrm{FIO}_\lambda\neq\emptyset$, $\mathrm{FIO}-\mathrm{FIIP}\neq\emptyset$.

Theorem 3. The following diagram holds



Theorem 4. a) The families in the next pairs are incomparable: $(\mathcal{L}_2, \mathrm{FIP})$ $(\mathcal{L}_2, \mathrm{FIRC})$. b) The following relations hold: $\mathrm{FIRC}-\mathrm{FIP}_{ac}\neq\emptyset$, $\mathrm{FIP}-\mathrm{FIO}_\lambda\neq\emptyset$ $\mathrm{FIRC}-\mathrm{FIO}_\lambda\neq\emptyset$.

3. Final remarks and open problems

As it may be noticed from the previous results, any recursively enumerable set can be generated by fully initial context-free grammars with the following regulated rewriting: matrices, programming, regular control and random context, provided that the appearance checking mode of derivation is present. If λ-rules are not allowed, then the fully initial regular control and matrix grammars are weaker than the context sensitive grammars and they are stronger than the context-free ones. Moreover, the fully initial context-free ordered, programmed, random context and matrix λ-free grammars give a hierarchy of languages (appearance checking is supposed). The family of context-free languages strictly includes the fully initial corresponding family, but it is strictly contained in the family of fully initial regular control and matrix languages. Both the families of regular and context-free languages are incomparable with the families of fully initial ordered and of Indian parallel languages, as well as, with the families of fully initial λ-free programmed and random context languages. The incomparability of the fully initial ordered family (with λ-rules) with the fully initial random context and programmed families is only partially solved: we said nothing about $\mathrm{FIO}_\lambda-\mathrm{FIP}_{ac}$ and $\mathrm{FIO}_\lambda-\mathrm{FIRC}_{ac}$. Without appearance checking but with λ-rules, it seems that the fully initial random context grammars are weaker than the regular control, the matrix and the programmed grammars. Moreover, in the λ-free case, the fully initial programmed and random context grammars are stronger than the fully initial context-free grammars, but the relation between them remains open (we know only that $\mathrm{FIRC}-\mathrm{FIP}\neq\emptyset$). As these open problems correspond to some unsettled questions about usual regulated grammars, the answers are not expected to be easy.

Similarly to the usual case, the Indian parallel family has a "lateral" position (incomparable with FI, $\mathcal{L}_2$ etc.).

References

[1] J. Dassow, Gh. Păun, *Regulated rewriting in formal language theory*, Akademie Verlag, Berlin, 1989.
[2] J. Dassow, On fully initial context-free languages, *Papers on automata and languages* (Ed. I. Peák), X (1988), 3—6.
[3] Gh. Păun, A note on fully initial context-free languages, *Papers on automata and languages* (Ed. I. Peák), X (1988), 7—11.
[4] A. Salomaa, *Formal languages*, Academic Press, New York, London, 1973.

(Received June 21, 1988)

On star-products of automata

F. GÉCSEG, B. IMREH

Bolyai Institute, Aradi vértanúk tere 1, H-6720 Szeged, Hungary

The study of complete systems of automata was initiated by V. M. Gluškov in [3]. In this work he characterized isomorphically complete systems with respect to the Gluškov-type product. Further characterizations of isomoprhically complete systems with respect to different kinds of products were presented in the works [1], [2] and [5]. In this paper we deal with star-products which have been deeply investigated in [6] and [7], and study isomorphic completeness for this kind of products. It will turn out that there exists no finite isomorphically complete system, however, as shown in [6], there are finite isomorphically S-complete systems with respect to it.

1. Definitions

By an *automaton* we mean a system $\mathbf{A}=(X, A, \delta)$, where A and X are finite nonvoid sets, and $\delta\colon A\times X^*\to A$ is the transition function. (Here and in the sequel X^* denotes the free monoid generated by X.) The concepts of *subautomaton* and *isomorphism* will be used in the usual sense.

Let $A_t=(X_t, A_t, \delta_t)$ $(t=1, \ldots, k)$ be a system of automata. Moreover, let X be a finite nonvoid set and φ a mapping of $A_1\times\ldots\times A_k\times X$ into $X_1\times\ldots\times X_k$ such that φ can be given in the form

$$\varphi(a_1, \ldots, a_k, x) = \big(\varphi_1(a_1, \ldots, a_k, x), \varphi_2(a_1, a_2, x), \ldots, \varphi_k(a_1, a_k, x)\big).$$

We say that

$$\mathbf{A} = (X, A, \delta)$$

is a *star-product* of $\mathbf{A}_t$ $(t=1, \ldots, k)$ with respect to X and φ if $A=A_1\times\ldots\times A_k$ and for arbitrary $(a_1, \ldots, a_k)\in A$ and $x\in X$

$$\delta((a_1, \ldots, a_k), x) = \big(\delta_1(a_1, \varphi_1(a_1, \ldots, a_k, x)), \delta_2(a_2, \varphi_2(a_1, a_2, x)), \ldots, \\ \ldots, \delta_k(a_k, \varphi_k(a_1, a_k, x))\big).$$

For this product we use the notation

$$\prod_{t=1}^{k} \mathbf{A}_t(X, \varphi).$$

As regards the introduced composition, let us observe the following: if the product-automaton is in the state $(a_1, \ldots, a_k)$ and receives an input sign x, then the automaton $\mathbf{A}_1$ receives the input sign $x_1 = \varphi_1(a_1, \ldots, a_k, x)$ which depends on x and all the actual states, and for every index $2 \leqq j \leqq k$ the automaton $\mathbf{A}_j$ receives the input sign $x_j = \varphi_j(a_1, a_j, x)$ which depends on the actual states a_1, a_j and x. Therefore, at a given moment the working of $\mathbf{A}_1$ depends on all component automata, while the working of $\mathbf{A}_j$ $(2 \leqq j \leqq k)$ depends on $\mathbf{A}_1$ and $\mathbf{A}_j$ only. This connection can be realized if the automaton $\mathbf{A}_1$ is placed in the centre and it is connected directly to each $\mathbf{A}_j$ $(2 \leqq j \leqq k)$, as illustrated in Fig. 1. This network of automata corresponds to the simplest computer network.

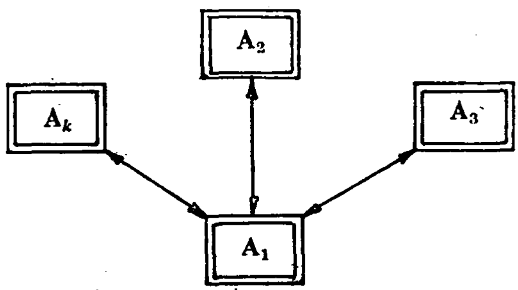


Fig. 1. Schematic diagram for the star-product of $\mathbf{A}_1, \ldots, \mathbf{A}_k$

2. Isomorphic realization

Let Σ be a system of automata. Σ is called *isomorphically comptete* with respect to the star-product if every automaton can be embedded isomorphically into a star-product of automata from Σ. Furthermore, Σ is a *minimal* isomorphically complete system if Σ is isomorphically complete and for arbitrary $\mathbf{A} \in \Sigma$, the system $\Sigma \setminus \{\mathbf{A}\}$ is not isomorphically complete.

For arbitrary positive integer n, let us denote by

$$\mathbf{D}_n = (\{x_{rs}\colon 1 \leqq r, s \leqq n\}, \{1, \ldots, n\}, \delta_n)$$

the automaton, where δ_n is determined in the following way: for arbitrary $i \in \{1, \ldots, n\}$ and input sign x_{rs} $(1 \leqq r, s \leqq n)$,

$$\delta_n(i, x_{rs}) = \begin{cases} s & \text{if } \ i = r, \\ i & \text{otherwise.} \end{cases}$$

Now we present a necessary and sufficient condition for the isomorphic completeness.

Theorem 1. A system Σ of automata is isomorphically complete with respect to the star-product if and only if for every positive integer n, there exists an automaton $\mathbf{A}\in\Sigma$ such that $\mathbf{D}_n$ can be embedded isomorphically into a star-product of $\mathbf{A}$ with a single factor.

Proof. First we show that $\mathbf{D}_n$ $(n>1)$ can be embedded isomorphically into a star-product of automata from Σ with at most two factors if $\mathbf{D}_n$ can be embedded isomorphically into a star-product of automata from Σ. For this, suppose that $\mathbf{D}_n$ can be embedded isomorphically into the star-product

$$\prod_{t=1}^{k} \mathbf{A}_t(\{x_{rs}\colon\ 1 \leqq r, s \leqq n\}, \varphi),$$

where $\mathbf{A}_t\in\Sigma$ $(t=1, ..., k)$ and $k>2$. Let us denote by μ such an isomorphism, and for arbitrary $i\in\{1, ..., n\}$ let $(a_{i1}, ..., a_{ik})$ be the image of i under μ. Now take an m $(2\leqq m\leqq k)$, and assume that $a_{i1}=a_{j1}$ and $a_{im}=a_{jm}$ hold for some indices $i\neq j$ $(1\leqq i,j\leqq n)$. Moreover, let $v\in\{1, ..., n\}$ be arbitrary. Then $\delta_n(i, x_{iv})=v$, $\delta_n(j, x_{iv})=j$, and since μ is an isomorphism, we obtain

$$\delta_m(a_{im}, \varphi_m(a_{i1}, a_{im}, x_{iv}))=a_{vm},$$
$$\delta_m(a_{jm}, \varphi_m(a_{j1}, a_{jm}, x_{iv}))=a_{jm}.$$

From this, by our assumption $a_{i1}=a_{j1}$ and $a_{im}=a_{jm}$, it follows that $a_{vm}=a_{jm}$. Since v is arbitrary, $a_{jm}=a_{vm}$ $(v=1, ..., n)$. Therefore, there is an index $(2\leqq m\leqq k)$ such that the pairs (a_{i1}, a_{im}) $(i=1, ..., n)$ are pairwise different. But then the automaton $\mathbf{D}_n$ can be embedded isomorphically into a star-product of $\mathbf{A}_1$ and $\mathbf{A}_m$, which yields the validity of our statement.

Now in order to prove the necessity, let us assume that Σ is isomoprhically complete with respect to the star-product. Let n be an arbitrary positive integer. The case $n=1$ being obvious, we may assume that $n>1$. Let $w=n^2$. Since Σ is isomorphically complete, $\mathbf{D}_w$ can be embedded isomorphically into a star-product

$$\prod_{t=1}^{k} \mathbf{A}_t(\{x_{rs}\colon\ 1 \leqq r, s \leqq w\}, \varphi)$$

of automata from Σ. From this, by the above assertion, it follows that $\mathbf{D}_w$ can be embedded isomorphically into a star-product of $\mathbf{A}_1$ and $\mathbf{A}_m$ for some $2\leqq m\leqq k$. But in this case it is easy to see that $\mathbf{D}_n$ can be embedded isomorphically into a star-product of one of the automata $\mathbf{A}_1$ and $\mathbf{A}_m$ with a single factor, which results the necessity of the condition.

To prove the sufficiency, it is enough to show that arbitrary automaton with n states can be embedded isomorphically into a star-product of $\mathbf{D}_n$ with a single factor, which is obvious. This ends the proof of Theorem 1.

Corollary. There exists no system of automata which is isomorphically complete with respect to the star-product and minimal.

Proof. Let Σ be isomorphically complete with respect to the star-product, and take an $\mathbf{A}\in\Sigma$ with $|A|=n$. Let $m>n$ be a fixed positive integer. Then $\mathbf{A}$ can be embedded isomorphically into a star-product of $\mathbf{D}_m$ with a single factor. On the other hand, by Theorem 1, there exists an $\mathbf{A}^* \in \Sigma$ such that $\mathbf{D}_m$ can be embedded isomorphically into a star-product of $\mathbf{A}^*$ with a single factor. But then $\mathbf{A}$ can also be embedded into a star-product of $\mathbf{A}^*$ with a single factor. This results that $\Sigma \setminus \{\mathbf{A}\}$ is isomorphically complete with respect to the star-product. Therefore, Σ is not minimal.

3. Isomoprhic simulation

In [2] products are generalized in such a way that feedback functions take their values from the set of input words of the factors. Moreover, in homomorphic and isomorphic representations the words are permitted as counter images of input signs. It turned out that these new concepts are more powerful than the old ones. Under these new concepts completeness results for α_i-products are presented in [2], while [1] is dealing with the corresponding problems concerning v_i-products. The representation of automata by isomorphic simulation and generalized products corresponds to the computation of functions on networks of automata. Going on this line, we introduce the concept of the generalized star-product, and study complete systems with respect to such products and isomorphic simulation.

We start with the definition of the generalized star-product. Let $\mathbf{A}_t = (X_t, A_t, \delta_t)$ $(t=1, \ldots, k)$ be a system of automata. Moreover, let X be a finite nonviod set and φ a mapping of $A_1 \times \ldots \times A_k \times X$ into $X_1^* \times \ldots \times X_k^*$ such that φ can be given in the form

$$\varphi(a_1, \ldots, a_k, x) = (\varphi_1(a_1, \ldots, a_k, x), \varphi_2(a_1, a_2, x), \ldots, \varphi_k(a_1, a_k, x)).$$

It is said that the automaton

$$\mathbf{A} = (X, \prod_{t=1}^{k} A_t, \delta)$$

is a *generalized star-product* of $\mathbf{A}_t$ $(t=1, \ldots, k)$ with respect to X and φ is for arbitrary $(a_1, \ldots, a_k) \in \prod_{t=1}^{k} A_t$, and $x \in X$,

$$\delta((a_1, \ldots, a_k), x) = (\delta_1(a_1, \varphi_1(a_1, \ldots, a_k, x)),$$
$$\delta_2(a_2, \varphi_2(a_1, a_2, x)), \ldots, \delta_k(a_k, \varphi_k(a_1, a_k, x))).$$

Obviously, if for each automaton $\mathbf{A}_t$ its characteristic semigroup is equal to X_t, then the generalized star-product is simply the star-product.

Let $\mathbf{A} = (X, A, \delta)$ and $\mathbf{B} = (Y, B, \delta')$ be arbitrary automata. We say that $\mathbf{A}$ *isomorphically simulates* $\mathbf{B}$ if there exist one-to-one mappings $\mu: B \to A$ and $\tau: Y \to X^*$ such that $\mu(\delta'(b, y)) = \delta(\mu(b), \tau(y))$ for arbitrary state $b \in B$ and input sign $y \in Y$.

As far as the isomorphic simulation is concerned, we have

Lemma 1. If $\mathbf{A}$ isomorphically simulates $\mathbf{B}$ and $\mathbf{B}$ isomorphically simulates $\mathbf{C}$, then $\mathbf{C}$ can be simulated isomorphically by $\mathbf{A}$, too.

Now we define isomorphic S-completeness.

A system Σ of automata is *isomorphically* S-*complete* with respect to the generalized star-product if every automaton can be simulated isomorphically by a generalized star-product of automata from Σ.

We shall use the following special automata. For arbitrary $n \geqq 1$, let us denote by $\mathbf{T}_n = (T_n, N, \delta_n)$ the automaton for which $N = \{1, \ldots, n\}$, T_n is the set of all transformations of N and $\delta_n(i, t) = t(i)$ for all $i \in N$ and $t \in T_n$.

Now we are ready to prove the following result giving necessary and sufficient conditions for S-completeness.

Theorem 2. A system Σ of automata is isomorphically S-complete with respect to the generalized star-product if and only if Σ contains an automaton $\mathbf{A} = (X, A, \delta)$ which has two different states a, b and two (not necessarily different) words $p, q \in X^*$ with $\delta(a, p) = b$ and $\delta(b, q) = a$.

Proof. The necessity of the conditions is obvious. The sufficiency can be derived from Theorem 1 in [6]. Here, using a different approach, we present a constructive proof. For this let us suppose that the conditions are satisfied by $\mathbf{A} \in \Sigma$ under the states 0, 1 and words p, q. Let $s = qp$ and $r = pq$. Then $\delta(0, r) = 0$ and $\delta(1, s) = 1$.

From the definition of $\mathbf{T}_n$ it follows that every automaton $\mathbf{B} = (X, B, \delta)$ can be embedded isomorphically into $\mathbf{T}_n$ if $n \geqq |B|$. Therefore, by Lemma 1, it is enough to show that for arbitrary $n \geqq 1$, $\mathbf{T}_n$ can be simulated isomorphically by a generalized star-product of automata from Σ. On the other hand, in [4] it is proved that the mappings t_1, t_2, t_3 generate the full transformation semigroup over N, where t_1, t_2, t_3 are determined as follows:

$$t_1(i) = i+1 \quad \text{if} \quad 1 \leqq i < n \quad \text{and} \quad t_1(n) = 1,$$

$$t_2(1) = 2, \quad t_2(2) = 1 \quad \text{and} \quad t_2(i) = i \quad \text{if} \quad 3 \leqq i \leqq n,$$

$$t_3(1) = t_3(2) = 1, \quad \text{and} \quad t_3(i) = i \quad \text{if} \quad 3 \leqq i \leqq n.$$

Therefore, the automaton $\mathbf{T}_n$ can be simulated isomorphically by the subautomaton $\mathbf{T}'_n = (\{t_1, t_2, t_3\}, N, \delta'_n)$ of the automaton $\mathbf{T}_n$. Therefore, again by Lemma 1, we obtain that if for every n the automaton $\mathbf{T}'_n$ can be simulated isomorphically by a generalized star-product of automata from Σ, then Σ is isomorphically S-complete with respect to the generalized star-product.

Obviously, if $n \leqq 2$, then $\mathbf{T}'_n$ can be simulated isomorphically by a generalized star-product of $\mathbf{A}$ with a single factor. Thus, suppose that $n > 2$ is an arbitrarily fixed integer. To obtain a simulation of $\mathbf{T}'_n$ by a generalized star-product of automata from Σ, consider the generalized star-power $\mathbf{A}^n(Y, \varphi)$, where $Y = \{y_j \colon 1 \leqq j \leqq n\}$, and using a function $\psi \colon \{0, 1\} \to \{r, s\}$, the mappings φ_j are defined in the following way: for arbitrary $a, b, a_1, \ldots, a_k \in \{0, 1\}$,

$$\psi(a) = \begin{cases} s, & \text{if} \quad a = 1, \\ r, & \text{if} \quad a = 0, \end{cases}$$

$$\varphi_1(a_1, \ldots, a_n, y_1) = \begin{cases} p, & \text{if} \quad a_1 = 0, \ a_2 = 1, \\ \psi(a_1) & \text{otherwise}, \end{cases}$$

$$\varphi_2(a, b, y_1) = \begin{cases} q, & \text{if} \quad a = 0, \ b = 1, \\ \psi(b) & \text{otherwise}, \end{cases}$$

$$\varphi_j(a, b, y_1) = \psi(b) \, (j = 3, \ldots, n),$$

$$\varphi_1(a_1, \ldots, a_n, y_i) = \begin{cases} q, & \text{if } a_1 = 1, \\ p, & \text{if } a_1 = 0,\ a_i = 1,\ (i = 2, \ldots, n) \\ \psi(a_1) & \text{otherwise,} \end{cases}$$

$$\varphi_j(a, b, y_j) = \begin{cases} p, & \text{if } a = 1, \\ q, & \text{if } a = 0,\ b = 1,\ (j = 2, \ldots, n) \\ \psi(b) & \text{otherwise,} \end{cases}$$

$$\varphi_j(a, b, y_i) = \psi(b) \quad (2 \leqq j \leqq n,\ 2 \leqq i \leqq n,\ i \neq j).$$

Take the mappings

$$\mu: \begin{cases} 1 \to (1, 0, \ldots, 0, 0), \\ 2 \to (0, 1, \ldots, 0, 0), \\ \vdots \\ n \to (0, 0, \ldots, 0, 1), \end{cases}$$

and

$$\tau: \begin{cases} t_1 \to y_2 \ldots y_n, \\ t_2 \to y_2, \\ t_3 \to y_1. \end{cases}$$

The validity of the equalities $\mu(\delta_n'(i, t_j)) = \delta_{A^n}(\mu(i), \tau(t_j))$ $(j = 1, 2, 3)$ can be checked in a trivial way. This completes the proof of Theorem 2.

Remark. Let us consider the automaton $A_2 = (\{x, y\}, \{0, 1\}, \delta)$ with the transition function $\delta(0, x) = \delta(1, y) = 1$, $\delta(1, x) = \delta(0, y) = 0$. From the above constructive proof it follows that $\Sigma = \{A_2\}$ is isomorphically S-complete with respect to the star-product.

Acknowledgement. The authors are grateful to Z. Ésik for calling their attention to the papers [6] and [7], and for suggesting a simplification in the original proof of Theorem 2.

References

[1] DÖMÖSI, P. and B. IMREH, On ν_i-products of automata, Acta Cybernet., v. 6, 1983, pp. 149—162
[2] GÉCSEG, F., On products of abstract automata, Acta Sci. Math. (Szeged), v. 38, 1976, pp. 21—43
[3] GLUŠKOV, V. M., Abstract theory of automata (Russian), Uspehi matematičeskih nauk, 16:5 101 (1961), pp. 3—62.
[4] GLUŠKOV, V. M., On complete systems of operations in computers (Russian), Kibernetika, v. 2, 1968, pp. 1—5.
[5] IMREH, B., On α_i-products of automata, Acta Cybernet., v. 3, 1978, pp. 301—307.
[6] TCHUENTE, M., Computation on binary tree-networks, Discrete Applied Mathematics, v. 14, 1986, pp. 295—310.
[7] TCHUENTE, M., Computations on finite networks of automata, Lecture Notes in Computer Science, v. 316, 1988, pp. 53—67.

(Received February 20, 1989)

On characteristic semigroups of Mealy automata

G. TANAKA

Hiroshima Shudo University Numata-cho, Asaminami-ku Hiroshima 731—31, JAPAN

Dedicated to Professor Miyuki Yamada on his 60th birthday.

Abstract

The purpose of this paper is to investigate the characteristic semigroup of a Mealy automaton. We show that there exists a bijection from the set of regular $\mathcal{D}$-classes of a characteristic semigroup $S'(M)$ of a Mealy automaton M onto the set of regular $\mathcal{D}$-classes of the semigroup $S(M^*)$ of the projection M^*.

1. Introduction

For a set I, the cardinality of I is denoted by $|I|$. I^* is the free monoid with an identity ε generated by I, and $I^+=I^*-\{\varepsilon\}$. If $w\in I^+$ is a nonempty word, then we denote by $\bar{w}$ the last letter of w. We use the symbol $\emptyset$ for the empty set.

Let $\delta\colon S\to S_1$ and $\lambda\colon S_1\to S_2$ be mappings of S and S_1, respectively. We read a product $\delta\lambda$ from left to right: $(s)\delta\lambda=((s)\delta)\lambda$, $s\in S$. The set $(S)\delta$ is called the image of δ and it is denoted by $\operatorname{Im}\delta$. The equivalence relation $\operatorname{Ker}\delta$ defined on S by $(s_1, s_2)\in\operatorname{Ker}\delta$ if and only if $(s_1)\delta=(s_2)\delta$ is called the kernel of δ.

An *automaton A* is a triple $A=(S, I, \delta)$, where S is a nonempty set of states, I is a nonempty set of inputs, δ is a state transition function such that $\delta(s, xy)=$ $=\delta(\delta(s, x), y)$ and $\delta(s, \varepsilon)=s$ for all $s\in S$ and all $x, y\in I^*$.

A *Mealy automaton M* is a quintuple $M=(S, I, U, \delta, \lambda)$, where $M^*=(S, I, \delta)$ is an automaton, U is a nonempty set of outputs, $\lambda\colon S\times I\to U$ is an output function. The output function is also used in the extended sence; for $s\in S$ and $xy\in I^*$ such that $x\in I^*$ and $y\in I$, $\lambda(s, \varepsilon)=\varepsilon$ and $\lambda(s, xy)=\lambda(s, x)\lambda(\delta(s, x), y)$.

The automaton M^* mentioned above is called the *projection* of the Mealy automaton M.

Let $M=(S, I, U, \delta, \lambda)$ be a Mealy automaton. To each $x\in I^+$ we assign the transformation δ_x on S, where $\delta_x\colon s\to\delta(s, x)$, $s\in S$. Let $S(M^*)=\{\delta_x|x\in I^+\}$.

Then $S(M^*)$ is a subsemigroup of the full transformation semigroup on S. To each $x \in I^+$ we assign the mapping λ_x: $s \to \lambda(s, x)$, $s \in S$. If xy is an element of I^+ such that both x and y are in I^+, then $(s)\lambda_{xy} = (s)\delta_x\lambda_y$.

The congruence ϱ on I^+ is defined by $x\varrho y$ if and only if $\delta_x = \delta_y$ and $\lambda_x = \lambda_y$. Put $S'(M) = \{(\lambda_x, \delta_x) | x \in I^+\}$. In $S'(M)$ we introduce the multiplication as follows:

$$(\lambda_x, \delta_x)(\lambda_y, \delta_y) = (\delta_x \lambda_y, \delta_x \delta_y).$$

Since $(\delta_x\lambda_y, \delta_x\delta_y) = (\lambda_{xy}, \delta_{xy}) \in S'(M)$, the set $S'(M)$ forms a semigroup which is isomorphic to I^+/ϱ. In this paper $S'(M)$ is called the *characteristic semigroup* of M. We note that if $\lambda_x = \lambda_y$ and $\delta_x = \delta_z$ $(x, y, z \in I^+)$, then $(\lambda_y, \delta_z) = (\lambda_x, \delta_x)$ as a pair of mappings and $(\lambda_y, \delta_z) \in S'(M)$.

We shall remark on another aspect of the characteristic semigroup of a finite Mealy automaton.

Remark. Assume that S is a finite set. On the output set U we define a multiplication by $ab = b$, $(a, b \in U)$. In such a way we obtain a right zero semigroup U. To each (λ_x, δ_x) in $S'(M)$ we define the $|S| \times |S|$ row-monomial matrix $M(\lambda_x, \delta_x)$ by

$$M(\lambda_x, \delta_x)_{st} = \begin{cases} (s)\lambda_x & \text{if } (s)\delta_x = t, \\ 0 & \text{otherwise.} \end{cases}$$

Two matrices are multiplied in the obvious way, and the set of all matrices forms a semigroup. Since the mapping $(\lambda_x, \delta_x) \to M(\lambda_x, \delta_x)$ is an isomorphism, $S'(M)$ is isomorphic to a subsemigroup of the wreath product $U \mathrm{wr} S(M^*)$ of U and $S(M^*)$ (see [7]).

2. Regular $\mathscr{D}$-class

On a semigroup T Green's relations are defined by

$$a\mathscr{R}b \Leftrightarrow aT^1 = bT^1, \quad a\mathscr{L}b \Leftrightarrow T^1 a = T^1 b,$$

$$a\mathscr{D}b \Leftrightarrow a\mathscr{L}c \quad \text{and} \quad c\mathscr{R}b \quad \text{for some } c \in T.$$

The intersection of two equivalences $\mathscr{R}$ and $\mathscr{L}$ is denoted by $\mathscr{H}$. An element x of a semigroup T is called *regular* if there exists y in T with $xyx = x$. If D is a $\mathscr{D}$-class, then either every element of D is regular or no element of D is regular. Therefore we call a $\mathscr{D}$-class regular if all its elements are regular. In a regular $\mathscr{D}$-class each $\mathscr{R}$-class and each $\mathscr{L}$-class contains at least one idempotent.

Let T be a subsemigroup of the full transformation semigroup on a set S, and let D be a regular $\mathscr{D}$-class of T. If $x, y \in D$, then we have $x\mathscr{L}y$ in $T \Leftrightarrow \mathrm{Im}\, x = \mathrm{Im}\, y$, and $x\mathscr{R}y$ in $T \Leftrightarrow \mathrm{Ker}\, x = \mathrm{Ker}\, y$ (see [2, p 39]).

The proof of the next lemma is omitted.

Lemma 1. Let δ be a transformation on a set S_1 such that $\delta^2 = \delta$, and let λ be a mapping from S_1 to S_2. Then $\delta\lambda = \lambda$ if and only if $\mathrm{Ker}\, \delta \subseteq \mathrm{Ker}\, \lambda$.

In what follows M means a Mealy automaton such that $M = (S, I, U, \delta, \lambda)$.

Theorem 1. $(\lambda_x, \delta_x) \in S'(M)$ is a regular element if and only if δ_x is a regular element of $S(M^*)$ and $\mathrm{Ker}\,\delta_x \subseteqq \mathrm{Ker}\,\lambda_x$.

Proof. "only if" part. Since (λ_x, δ_x) is a regular element, there exists some (λ_y, δ_y) in $S'(M)$ such that $\delta_x\delta_y\delta_x = \delta_x$ and $\delta_x\delta_y\lambda_x = \lambda_x$. This implies that $\mathrm{Ker}\,\delta_x \subseteqq \mathrm{Ker}\,\delta_x\delta_y\lambda_x = \mathrm{Ker}\,\lambda_x$. "if" part. Since δ_x is a regular element, $\delta_x\delta_y\delta_x = \delta_x$ for some δ_y in $S(M^*)$. From $\delta_x\delta_y \mathscr{R} \delta_x$ we have $\mathrm{Ker}\,\delta_{xy} = \mathrm{Ker}\,\delta_x \subseteqq \mathrm{Ker}\,\lambda_x$.

Since δ_{xy} is an idempotent, by Lemma 1, $\delta_{xy}\lambda_x = \lambda_x$. Therefore we have $(\lambda_x, \delta_x) \cdot (\lambda_y, \delta_y)(\lambda_x, \delta_x) = (\lambda_x, \delta_x)$. Q.E.D.

For a subset H of $S'(M)$ we define the sets of mappings by

$$H^{(1)} = \{\lambda_x | (\lambda_x, \delta_x) \in H\}, \quad H^{(2)} = \{\delta_x | (\lambda_x, \delta_x) \in H\}.$$

Theorem 2. If L is an $\mathscr{L}$-class contained in a regular $\mathscr{D}$-class of $S'(M)$, then $L^{(2)}$ is an $\mathscr{L}$-class of $S(M^*)$.

Proof. It is clear that there exists some regular $\mathscr{L}$-class L^* of $S(M^*)$ such that $L^{(2)} \subseteqq L^*$. Now we show the validity of the reverse inclusion. Let $(\lambda_e, \delta_e) \in L$ be an idempotent of $S'(M)$. Then δ_e is an idempotent of L^* and δ_e is a right identity for L^*. Hence for every δ_x in L^* we have $\delta_x\delta_e = \delta_x$ and $\delta_p\delta_x = \delta_e$ for some δ_p in $S(M^*)$. Consequently, $(\delta_x\lambda_e, \delta_x) = (\lambda_{xe}, \delta_{xe}) \in S'(M)$ and $(\delta_x\lambda_e, \delta_x)(\lambda_e, \delta_e) = (\delta_x\lambda_e, \delta_x)$. Moreover, we have $(\lambda_p, \delta_p)(\delta_x\lambda_e, \delta_x) = (\lambda_e, \delta_e)$. This yields that $(\lambda_e, \delta_e) \mathscr{L} (\delta_x\lambda_e, \delta_x)$ in $S'(M)$, and therefore $\delta_x \in L^{(2)}$. Q.E.D.

Theorem 3. If L is an $\mathscr{L}$-class contained in a regular $\mathscr{D}$-class of $S'(M)$, then $(\lambda_x, \delta_x) \to \delta_x$ is a bijection from L onto $L^{(2)}$.

Proof. An idempotent (λ_e, δ_e) in L is a right identity for L. If $(\lambda_p, \delta_x), (\lambda_q, \delta_x) \in L$, then

$$(\lambda_p, \delta_x) = (\lambda_p, \delta_x)(\lambda_e, \delta_e) = (\delta_x\lambda_e, \delta_x) = (\lambda_q, \delta_x)(\lambda_e, \delta_e) = (\lambda_q, \delta_x).$$

Q.E.D.

Let H_1 and H_2 be $\mathscr{H}$-classes contained in the same $\mathscr{D}$-class of $S'(M)$. Then, using Green's lemma, it can be seen that $|H_1^{(2)}| = |H_2^{(2)}|$ holds (see [5]). However, there are examples that show that in general the equality $|H_1^{(1)}| = |H_2^{(1)}|$ does not hold. Therefore, in the next theorem, the condition that boht H_1 and H_2 are in the same $\mathscr{L}$-class is indispensable.

Theorem 4. Let L be an $\mathscr{L}$-class in a regular $\mathscr{D}$-class of $S'(M)$. If H_1 and H_2 are two $\mathscr{H}$-classes contained in L, then $|H_1^{(1)}| = |H_2^{(1)}|$.

Proof. Let (λ_e, δ_e) be an idempotent of L, and let H be an $\mathscr{H}$-class of (λ_e, δ_e). If $\lambda_z \in H^{(1)}$, then $\delta_e\lambda_z = \lambda_z$ since (λ_e, δ_e) is an identity of H. Let $\lambda_x, \lambda_y \in H^{(1)}$ and $\lambda_x \neq \lambda_y$. Then $(s)\delta_e\lambda_x \neq (s)\delta_e\lambda_y$ for some $s \in S$, therefore λ_x and λ_y are distinct mappings on $\mathrm{Im}\,\delta_e$. Let H_1 be an arbitrary $\mathscr{H}$-class in L. Then $(\lambda_p, \delta_p)H = H_1$ for some (λ_p, δ_p) in $S'(M)$. Thus $H_1^{(1)} = \{\delta_p\lambda_w | \lambda_w \in H^{(1)}\}$. Assume that $\delta_p\lambda_x = \delta_p\lambda_y$ for some $\lambda_x, \lambda_y \in H^{(1)}$, $(\lambda_x \neq \lambda_y)$. Then $\delta_p\delta_e\lambda_x = \delta_p\delta_e\lambda_y$. Since $\delta_p\delta_e \in H_1^{(2)}$, we have that $\delta_p\delta_e \mathscr{L} \delta_e$, and so, $\mathrm{Im}\,\delta_p\delta_e = \mathrm{Im}\,\delta_e$. Therefore for every $s \in \mathrm{Im}\,\delta_e$ there exists some $t \in S$ with $(t)\delta_p\delta_e = s$. Then $(s)\lambda_x = (t)\delta_p\delta_e\lambda_x = (t)\delta_p\delta_e\lambda_y = (s)\lambda_y$ holds for every s in $\mathrm{Im}\,\delta_e$, which is a contradiction. Hence $\lambda_x \neq \lambda_y$ implies $\delta_p\lambda_x \neq \delta_p\lambda_y$. This shows that the mapping $\theta\colon H^{(1)} \to H_1^{(1)}$ defined by $(\lambda_w)\theta = \delta_p\lambda_w$ is a bijection from $H^{(1)}$ onto $H_1^{(1)}$. Q.E.D.

Theorem 5. If R is an $\mathscr{R}$-class contained in a regular $\mathscr{D}$-class of $S'(M)$, then $R^{(2)}$ is an $\mathscr{R}$-class of $S(M^*)$.

Proof. It is clear that there exists an $\mathscr{R}$-class R^* of $S(M^*)$ such that $R^{(2)} \subseteq R^*$. We shall show that the reverse inclusion holds, too. Let $(\lambda_e, \delta_e) \in R$ be an idempontent. Then δ_e is an idempotent in R^*, and therefore, $\delta_e \delta_x = \delta_x$ for every $\delta_x \in R^*$. For the word $ex \in I^+$ we have $(\lambda_{ex}, \delta_{ex}) = (\delta_e \lambda_x, \delta_x) \in S'(M)$. Since $\delta_x \mathscr{R} \delta_e$, there exists some $\delta_p \in S(M^*)$ such that $\delta_x \delta_p = \delta_e$. In this case $(\delta_e \lambda_x, \delta_x)(\lambda_{pe}, \delta_{pe}) = (\lambda_e, \delta_e)$ and $(\lambda_e, \delta_e)(\delta_e \lambda_x, \delta_x) = (\delta_e \lambda_x, \delta_x)$. Therefore $(\delta_e \lambda_x, \delta_x) \in R$ and $\delta_x \in R^{(2)}$. Q.E.D.

Theorem 6. ([6]). Let D be a regular $\mathscr{D}$-class of $S'(M)$ and (λ_x, δ_x), $(\lambda_y, \delta_y) \in D$. Then $(\lambda_x, \delta_x) \mathscr{R} (\lambda_y, \delta_y)$ if and only if $\operatorname{Ker} \delta_x = \operatorname{Ker} \delta_y \subseteq (\operatorname{Ker} \lambda_x \cap \operatorname{Ker} \lambda_y)$.

Theorem 7. If R_1 and R_2 are distinct $\mathscr{R}$-classes in the same regular $\mathscr{D}$-class of $S'(M)$, then $R_1^{(2)} \cap R_2^{(2)} = \emptyset$.

Proof. If $R_1^{(2)} \cap R_2^{(2)} \neq \emptyset$ then, by Theorem 5, we have $R_1^{(2)} = R_2^{(2)}$. If $(\lambda_x, \delta_x) \in R_1$ and $(\lambda_y, \delta_y) \in R_2$, then δ_x and δ_y are in $R_1^{(2)}$, thus $\operatorname{Ker} \delta_x = \operatorname{Ker} \delta_y$. By Theorem 1, $\operatorname{Ker} \delta_x \subseteq \operatorname{Ker} \lambda_x$ and $\operatorname{Ker} \delta_y \subseteq \operatorname{Ker} \lambda_y$. Therefore, by Theorem 6, we have that $(\lambda_x, \delta_x) \mathscr{R} (\lambda_y, \delta_y)$, and so $R_1 = R_2$, which is a contradiction. Q.E.D.

Theorem 8. If D is a regular $\mathscr{D}$-class of $S'(M)$, then $D^{(2)}$ is a regular $\mathscr{D}$-class of $S(M^*)$.

Proof. It is obvious that there exists a regular $\mathscr{D}$-class D^* such that $D^{(2)} \subseteq D^*$. We show that the reverse inclusion holds. Let $\delta_x \in D^*$ and let L^* be an $\mathscr{L}$-class of D^* containing δ_x. If R is an $\mathscr{R}$-class of D then, by Theorem 5, $R^{(2)}$ is an $\mathscr{R}$-class of D^*. Hence $R^{(2)} \cap L^* \neq \emptyset$. If $\delta_y \in R^{(2)} \cap L^*$, then $(\lambda_p, \delta_y) \in D$ for some λ_p. Let L be an $\mathscr{L}$-class containing (λ_p, δ_y). Then $\delta_y \in L^{(2)} \cap L^*$. Thus, by Theorem 2, $L^{(2)} = L^*$. This means that $\delta_x \in L^{(2)} \subseteq D^{(2)}$, and so $D^* \subseteq D^{(2)}$. Q.E.D.

Theorem 9. Let D be a regular $\mathscr{D}$-class of $S'(M)$, and let D_R and $D_R^{(2)}$ be sets of $\mathscr{R}$-classes of D and $D^{(2)}$, respectively. Then $|D_R| = |D_R^{(2)}|$.

Proof. By Theorems 7 and 8, the mapping $R \to R^{(2)}$ is a bijection from the set of $\mathscr{R}$-classes of D onto the set of $\mathscr{R}$-classes of $D^{(2)}$. Q.E.D.

If D is a finite regular $\mathscr{D}$-class, then D and $D^{(2)}$ consists of the same number of $\mathscr{R}$-classes. However, note that we cannot in general assert that D and $D^{(2)}$ have the same number of $\mathscr{L}$-classes.

Lemma 2. If (λ_w, δ_e) is a regular element of $S'(M)$ such that δ_e is an idempotent, then (λ_w, δ_e) is an idempotent and $\lambda_w = \delta_e \lambda_w$.

Proof. There exists some idempotent (λ_f, δ_f) such that $(\lambda_w, \delta_e) \mathscr{L} (\lambda_f, \delta_f)$. Since (λ_f, δ_f) is a right identity in its $\mathscr{L}$-class, we obtain that $(\lambda_w, \delta_e)(\lambda_f, \delta_f) = (\delta_e \lambda_f, \delta_e \delta_f) = (\lambda_w, \delta_e)$. Thus $\delta_e \lambda_f = \lambda_w$. From this we have that (λ_w, δ_e) is an idempotent and $\lambda_w = \delta_e \lambda_w$. Q.E.D.

Theorem 10. If D^* is a regular $\mathscr{D}$-class of $S(M^*)$, then there exists a unique regular $\mathscr{D}$-class D of $S'(M)$ such that $D^{(2)} = D^*$.

Subscription information:

For Albania, Bulgaria, China, Cuba, Czechoslovakia, German Democratic Republic, Korean People's Republic, Mongolia, Poland, Romania, USSR, Vietnam
orders should be addressed to:

Kultura
Hungarian Foreign Trading Co.
H—1389 Budapest 62
P. O. Box 149
Hungary

For all other countries orders should be adressed to:

J. C. Baltzer AG
Scientific Publishing Company
Wettsteinplatz 10
CH—4058 Basel
Switzerland

Mailing address for editorial correspondence:

Acta Cybernetica
Aradi vértanúk tere 1.
Szeged
H—6720 Hungary

CONTENTS

ISSN 0324—721 X

Felelős szerkesztő és kiadó: Gécseg Ferenc
A kézirat a nyomdába érkezett: 1989. április
Terjedelem: 8,05 (A/5) ív
Készült monószedéssel, íves magasnyomással
az MSZ 6601 és az MSZ 5602—55 szabvány szerint
88-1143 — Szegedi Nyomda — Felelős vezető: Surányi Tibor igazgató