ANNALES
UNIVERSITATIS MARIAE CURIE-SKŁODOWSKA
LUBLIN-POLONIA
VOL. XLIX, 6 SECTIO A 1995

Zdzisław GRODZKI and Jerzy MYCKA (Lublin)

The Equivalence of Some Classes of Algorithms

ABSTRACT. This paper is a starting point of investigations on uniform transposition of well known notions of formal algorithms (Turing machines, Markov normal algorithms, unlimited register machines), formal grammars, as well as programming languages like PASCAL, by means of iterative systems [6]. Using the same idea we are able to introduce new classes of Markov-like k-algorithms. In this paper only two classes $\mathcal{MA}_k$ and $\overline{\mathcal{MA}_k}$ of Markov-like k-algorithms are introduced and briefly characterized. The equivalence of each of the above classes and the class $\mathcal{MNA}$ of Markov normal algorithms is shown. This equivalence implies the closure properties of $\mathcal{MA}_k$ and $\overline{\mathcal{MA}_k}$ under the same operations as those in $\mathcal{MNA}$.

1. Introduction. Iterative systems have been used by Pawlak to define some classes of computing machines [6] and by Mazurkiewicz [4] to define a very general class of programming languages. We extend this idea to unify the well known notions of formal algorithms (Turing machines [5], Markov normal algorithms [3], unlimited register machines [1]), formal grammars, as well as programming languages like PASCAL. We are also able to define easily new classes of Markov-like k-algorithms which are equivalent to the class $\mathcal{MNA}$ of Markov normal algorithms. The unified definition of classes of algorithms mentioned above allows us to prove easily their equivalence.

In this paper only two classes $\mathcal{MA}_k$ and $\overline{\mathcal{MA}_k}$ of left-hand side Markov-like k-algorithms are introduced and briefly characterized (the "symmetric" classes $\mathcal{RMA}_k$ and $\overline{\mathcal{RMA}_k}$ of right-hand side

Markov-like k-algorithms will be introduced in the next authors' paper). The equivalence of the classes $\mathcal{MA}_k$ and $\overline{\mathcal{MA}_k}$ and the class $\mathcal{MNA}$ of Markov normal algorithms is shown. It follows immediately from this equivalence that both classes are closed under composition, ramification, propagation and iteration.

Let us summarize reasons motivating research on uniform formalization of different kinds of objects (formal grammars, effective algorithms and programming languages):

(1) Uniform formalization of different kinds of objects allows us to distinguish some class of problems which are essential for all objects; these problems can be easily reformulated from one formulation into another;

(2) It is possible to compare solutions of identical problems by means of different kinds of objects with respect to complexity of these solutions;

(3) Uniform formalization of the majority of algorithms allows us to prove the equivalence of particular classes;

(4) The classes $\mathcal{MA}_k$ and $\overline{\mathcal{MA}_k}$ of k-algorithms form an infinite sequence with respect to increasing k. In order to solve some problems we choose an algorithm A of $\mathcal{MA}_k \cup \overline{\mathcal{MA}_k}$ for some $k \geq 1$;

(5) The used formalism allows us to simulate the algorithms of the above mentioned classes by means of computing machines.

2. Algorithms.

A subclass $\mathcal{A}$ of algorithms of class $\mathcal{IS}$ of the iterative systems will be considered.

An iterative system IS is a pair (T, ϕ), where T is a nonempty set (finite or infinite) and ϕ is a partial function, $\phi : T \mapsto T$. Then T is called the set of states and ϕ the transition function of IS.

An iterative system $IS = (T, \phi)$ is said to be *an algorithm* iff the following conditions hold:
$T = P \times V$, where V is a set of objects and P is an indexed set $\{(x_i, y_i) \in V^2 : 1 \leq i \leq n\}$ whose elements are called *the productions*, $\phi = (Contr, Tr)$, where $Contr : P \times V \mapsto P$, called *control*, is a partial function and $Tr : P \times V \mapsto V$, called *transformation*, is a total function.

An algorithm $A = (P, V, Contr, Tr)$ is called *empty* and denoted by A^0 iff $\mathrm{Dom}(Contr) = \emptyset$; otherwise A is called nonempty.

An algorithm $A = (P, V, Contr, Tr)$ is called *total* iff $\mathrm{Dom}(Contr) = \mathrm{Dom}(Tr) = P \times V$.[1]

Let $\mathcal{A}$ denote the class of all algorithms including the empty algorithm A^0 which are defined in a common set of objects V.

A sequence $\mathbf{v} = v_1, v_2, \ldots \in V^\infty$ is said to be a computation of an algorithm $A = (P, V, Contr, Tr)$ iff there exists a sequence of productions (*trace*) of P of the form $\mathbf{p} = (x_{i_1}, y_{i_1}), (x_{i_2}, y_{i_2}), \ldots$ satisfying the conditions:

(2.1) $l(\mathbf{v}) = l(\mathbf{p})$, where $l(\mathbf{v})$, $l(\mathbf{p})$ denote the lengths of $\mathbf{v}$ and $\mathbf{p}$;

(2.2) For every j, $1 \leq j < l(\mathbf{v})$, we have $(x_{i_{j+1}}, y_{i_{j+1}}) = Contr((x_{i_j}, y_{i_j}), v_j)$ and $v_{j+1} = Tr((x_{i_j}, y_{i_j}), v_j)$;

(2.3) For every $m > 1$, if $l(\mathbf{v}) = m$ then $((x_{i_m}, y_{i_m}), v_m) \notin \mathrm{Dom}(Contr)$.

The set of all computations of an algorithm A is called its *computation set* and denoted by $C(A)$.

Let $\mathcal{C}(\mathcal{A})$ denote the class of all computation sets of all algorithms of $\mathcal{A}$.

Remark 2.1 Let us assign to every computation $\mathbf{v} = v_1, v_2, \ldots \in V^\infty$ of an algorithm A a sequence $\mathbf{d} = ((x_{i_1}, y_{i_1}), v_1), ((x_{i_2}, y_{i_2}), v_2), \ldots$ such that $l(\mathbf{v}) = l(\mathbf{d})$ and $(x_{i_1}, y_{i_1}), (x_{i_2}, y_{i_2}), \ldots$ is a trace of $\mathbf{v}$.

A sequence $\mathbf{d}$ of the above form is called *a derivation according to* A. The set of all derivations according to A is called its *derivation set* and denoted by $D(A)$.

3. The designated algorithms.

The subclass $\mathcal{DA}$ of $\mathcal{A}$ of the designated algorithms will be considered here. The algorithms of $\mathcal{DA}$ can be the models of programming languages like PASCAL or PROLOG. It will be shown in the next section that $\mathcal{DA}$ contains the class $\mathcal{MNA}$ of Markov normal algorithms, as well as two new classes $\mathcal{MA}_k$ and $\overline{\mathcal{MA}_k}$ of Markov-like k-algorithms.

By *a designated algorithm* of $\mathcal{DA}$ we mean an algorithm $A = (P, V, Contr, Tr)$ for which two nonempty subsets P_i and P_f of P of the initial and final productions are distinguished.

[1] $\mathrm{Dom}(\phi)$ denotes the domain of ϕ.

A designated algorithm will be written in the form:

$$DA = (P, P_i, P_f, V, Contr, Tr).$$

An activity of a designated algorithm $DA = (P, P_i, P_f, V, Contr, Tr)$ can be described by a set of sequences (finite or infinite) whose elements are the configurations. By *a configuration* we mean an element of $P \times V$.

A configuration $((x_i, y_i), v)$ *transforms directly* a configuration $((x_j, y_j), u)$ according to DA, in symbols $((x_i, y_i), v) \Rightarrow_{DA} ((x_j, y_j), u)$, iff $(x_j, y_j) = Contr((x_i, y_i), v)$ and $u = Tr((x_i, y_i), v)$, where (x_m, y_m) is an m-th element of the set of productions P, called the m-th production.

A sequence of configurations (finite or infinite) of the form:

$$\mathbf{d} = ((x_{i_1}, y_{i_1}), v_1), ((x_{i_2}, y_{i_2}), v_2), \ldots \in (P \times V)^\infty$$

is said to be *a derivation of DA* iff $(x_{i_1}, y_{i_1}) \in P_i$ and $((x_{i_j}, y_{i_j}), v_j) \Rightarrow_{DA} ((x_{i_{j+1}}, y_{i_{j+1}}), v_{j+1})$ for all $1 \le j < l(\mathbf{d})$.

A derivation is called *successful* if it is finite and its last element $((x_{i_p}, y_{i_p}), v_p)$ has the property: $(x_{i_p}, y_{i_p}) \in P_f$ and $Contr((x_{i_p}, y_{i_p}), v_p)$ is undefined.

A partial function $\pi : V \to V$ is said to be computable by an algorithm DA iff for every $v \in \mathrm{Dom}(\pi)$ there exists a successful derivation of the form:

$$\mathbf{d} = ((x_{i_1}, y_{i_1}), v_1), \ldots, ((x_{i_p}, y_{i_p}), v_p)$$

such that $v = v_1$ and $\pi(v) = Tr((x_{i_p}, y_{i_p}), v_p)$.

Remark 3.1 The iterative Mazurkiewicz's algorithms [4] which are models of programs without procedures can be easily reformulated as the designated algorithms. We omit here this reformulation but we give only some examples.

Example 3.2. Let us consider a program in PASCAL computing the function $\pi : N \times N \mapsto N \times N$ such that $\pi(0, n) = (n!, 0)$

```
s := n;
while n > 1 do
```

```
begin
$n := n - 1;$
$s := s * n;$
end;
```

Now we define the designated algorithm $DA = (P, P_i, P_f, V, Contr, Tr)$ which computes the function π:
$V = N \times N$,
$Tr(((m^i{}_1, m^i{}_2), (n^i{}_1, n^i{}_2)), (x_1, x_2)) = (y_1, y_2)$
and

$$(y_1, y_2) = \begin{cases} (x_1 - n^i{}_1, x_2 - n^i{}_2) & \text{if } m^i_1 > m^i_2 \\ (x_1 + n^i{}_1, x_2 + n^i{}_2) & \text{if } m^i_1 = m^i_2 \\ (x_1 * x_2 * n^i{}_1, x_2 * n^i{}_2) & \text{otherwise} \end{cases}$$

$Contr(((m^i{}_1, m^i{}_2), (n^i{}_1, n^i{}_2)), (x_1, x_2)) = ((m^j{}_1, m^j{}_2), (n^j{}_1, n^j{}_2))$,
where

$$j = \begin{cases} i+1 & \text{if } m^i_1 > m^i_2 \text{ and } x_2 = 1 \\ i-1 & \text{otherwise,} \end{cases}$$

$P = \{P_1 = ((0,0),(1,0)), P_2 = ((0,1),(1,1)), P_3 = ((1,0),(0,1)),$
$P_4 = ((0,0),(0,0))\}, P_i = \{P_1\}, P_f = \{P_4\}$.
One can easily see that DA computes the function π.

4. Markov-like k-algorithms. It will be shown that the class of designated algorithms $\mathcal{DA}$ contains the class $\mathcal{MNA}$ of Markov normal algorithms, as well as two new classes $\mathcal{MA}_k$ and $\overline{\mathcal{MA}_k}$ of Markov-like k-algorithms. The algorithms of both classes are defined analogously as algorithms of $\mathcal{MNA}$, by means of an indexed set of productions but the manner of use of the productions to the transformed words is different for every one of these classes. The succession of use of the productions for the algorithms of $\mathcal{MA}_k$ (resp. $\overline{\mathcal{MA}_k}$) is the same (resp. different) as for algorithms of $\mathcal{MNA}$.

Let us introduce at the beginning some notations.

For an alphabet Σ let Σ^* denote the set of all finite sequences (words) over Σ including the empty word ε.

Let $\mathbf{u}$ and $\mathbf{v}$ be arbitrary words of Σ^* of the lengths p and $m (p \leq m)$, respectively.

A word $\mathbf{u}$ is said to be a subword of $\mathbf{v}$, $\mathbf{u} \preceq \mathbf{v}$, iff there are the words $\mathbf{z}, \mathbf{w} \in \Sigma^*$ (possibly empty) such that $\mathbf{v} = \mathbf{zuw}$.

If $\mathbf{v} = v_1 \ldots v_m$ then for all $1 \leq i \leq j \leq m$ $\mathbf{v}_{[i,j]}$ will denote a sequence $v_i \ldots v_j$. For brevity we shall write $\mathbf{v}_{[i]}$ instead of $\mathbf{v}_{[i,i]}$. We assume that if $i > j$ then $\mathbf{v}_{[i,j]} = \varepsilon$.

Let us define the set of the initial positions of all occurrences of $\mathbf{u}$ in $\mathbf{v}$ as follows

$$N_{\mathbf{v};\mathbf{u}} = \{j \in \mathbb{N} : \mathbf{v}_{[j,j+p-1]} = \mathbf{u}\} .$$

Let us set

$$N^q_{\mathbf{v};\mathbf{u}} = \{j \in N_{\mathbf{v};\mathbf{u}} : j \leq q\} .$$

A sequence $\mathbf{v}_{[r,r+p-1]}$ is said to be *the i-th occurrence of* $\mathbf{u}$ *in* $\mathbf{v}$, $\mathbf{u} \preceq_i \mathbf{v}$, iff $r = \min_q(|N^q_{\mathbf{v};\mathbf{u}}| = i)$.

Example 4.1 Let $\mathbf{u} = 01$ and $\mathbf{v} = 1010111011$. Then $N_{\mathbf{v};\mathbf{u}} = \{2,4,8\}$ and $N^0_{\mathbf{v};\mathbf{u}} = N^1_{\mathbf{v};\mathbf{u}} = \emptyset; N^2_{\mathbf{v};\mathbf{u}} = N^3_{\mathbf{v};\mathbf{u}} = \{2\}; N^i_{\mathbf{v};\mathbf{u}}$ for $i = 4,5,6,7$ is equal to $\{2,4\}$ and for $j \geq 8$ $N^j_{\mathbf{v};\mathbf{u}} = \{2,4,8\}$.

Therefore the 2-nd occurrence of $\mathbf{u}$ in $\mathbf{v}$ is equal to $\mathbf{v}_{[4,5]}$, because $\min_q(|N^q_{\mathbf{v};\mathbf{u}}| = 2) = 4$.

We say that a sequence $\mathbf{v}_{[j,j+m-1]}$ is *at most i-th occurrence of* $\mathbf{u}$ *in* $\mathbf{v}$, $\mathbf{u} \preceq_{\leq i} \mathbf{v}$, iff $\mathbf{u} \preceq_i \mathbf{v}$ or there exists $1 \leq l < i$ such that $\mathbf{u} \preceq_l \mathbf{v}$ and $\neg(\mathbf{u} \preceq_{l+1} \mathbf{v})$.

First, it will be shown that Markov normal algorithms can be defined by means of the designated algorithms.

A designated algorithm

$$DA = (P, P_i, P_f, V, Contr_1, Tr_1)$$

is said to be Markov normal algorithm in the alphabet Σ iff the following conditions hold:
$V = \Sigma^\star$, P is an indexed set $\{(x_i, y_i) : 1 \leq i \leq n\}$ of productions, where $x_i, y_i \in \Sigma^\star$ for all $i (1 \leq i \leq n)$ and $(x_n, y_n) \in P_f$ with $x_n = y_n = \varepsilon$, $P_i = \{(x_1, y_1)\}$[2];
$Contr_1$ is the partial function, Tr_1 is the total function of $P \times V$ into P and V, respectively, which are defined as follows[3]:

$$Contr_1((x_i, y_i), \mathbf{v}) = \begin{cases} (x_1, y_1) & \text{if } x_i \preceq_1 \mathbf{v} \text{ and } (x_i, y_i) \notin P_f \\ (x_{i+1}, y_{i+1}) & \text{if } \neg(x_i \preceq_1 \mathbf{v}) \\ \text{undefined} & \text{if } x_i \preceq_1 \mathbf{v} \text{ and } (x_i, y_i) \in P_f \end{cases}$$

[2] We do not assume that $P_i \cap P_f = \emptyset$.

[3] In the definition of $Contr_1$ in the second case $\neg(x_i \preceq_1 v)$ implies the inequality $i < |P|$, because otherwise ($i = |P|$) the third case holds.

$$Tr_1((x_i,y_i),\mathbf{v}) = \begin{cases} \mathbf{v}_{[1,j-1]}y_i\mathbf{v}_{[j+m,l(\mathbf{v})]} & \text{if } \mathbf{v}_{[j,j+m-1]} \text{ is first} \\ & \text{occurence of } x_i \text{ in } \mathbf{v} \\ \mathbf{v} & \text{otherwise} \end{cases}$$

Markov normal algorithm DA is said to be an algorithm over an alphabet Σ iff it is an algorithm in some alphabet Σ' such that $\Sigma \subset \Sigma'$.

It follows immediately from the above definition that every derivation of DA is finite iff the last used production $(x_{i_p}, y_{i_p}) \in P_f$ and $Contr((x_{i_p}, y_{i_p}), v_p)$ is undefined.

Let us make some comments. For the whole class $\mathcal{MNA}$ of Markov normal algorithms the controls and transformations are the same. Therefore to define an algorithm of $\mathcal{MNA}$ it is sufficient to construct an indexed set P of productions, in which the final productions are indicated (first production is always initial). In Markov [3] the final productions are denoted by $x_i \longrightarrow \cdot y_i$ but the remaining ones by $x_i \longrightarrow y_i$ and the author said that last production of the form $\varepsilon \longrightarrow \cdot\varepsilon$ can be ommited.

Example 4.2. Let us define a Markov normal algorithm DA in the alphabet $\Sigma = \{0,1\}$ by means of the following set of productions: $P = \{(01,11),(111,10),(\varepsilon,\varepsilon)\}$ and the following set of final productions: $P_f = \{(01,11),(\varepsilon,\varepsilon)\}$.

For $\mathbf{v} = 1111$ the derivation of DA has the form:

$$\mathbf{d} = ((01,11),1111),((111,10),1111),((01,11),101)\ .$$

The result of the application of DA to the word 1111 is equal to $Tr_1((01,11),101) = 111$.

Let us observe that the first production is final but a derivation does not stop when this production is used first time, because $\neg(01 \preceq_1 1111)$.

Before giving formal definitions of the k-algorithms of $\mathcal{MA}_k$ and $\overline{\mathcal{MA}_k}$ let us give some intuitive remarks.

Every algorithm of both classes is defined by means of an indexed set P of productions, which is called *a schema of productions*. We additionally assume that P consists of the empty production $(\varepsilon, \varepsilon)$ which is an element of a subset P_f of P of the final productions and this production has the maximal index. An algorithm A of $\mathcal{MA}_k$ works in the following way. Given a word $\mathbf{v}_0 \in \Sigma^*$ we choose a production (x_i, y_i), with the least index i, such that x_i is a subword of $\mathbf{v}_0$. If such a production does not exist then we stop, otherwise we put a word y_i instead of the m-th left-hand side occurrence of x_i in $\mathbf{v}_0$ for the maximal $m \leq k$. If the above production is in P_f then we stop, otherwise we follow analogously with the new obtained word $\mathbf{v}_1$ as with $\mathbf{v}_0$.

If $A \in \overline{\mathcal{MA}_k}$ then we follow in a slightly different way. Given a word $\mathbf{v}_0 \in \Sigma^*$ we choose a production (x_j, y_j), with the least index j, such that x_j occurs in $\mathbf{v}_0$ at least k times. If such a production does not exist we choose a production (x_m, y_m) with the least index m such that x_m occurs in $\mathbf{v}_0$ $(k-1)$ times and so on. Let $p \leq k$ be the maximal number for which there exist a production (x_l, y_l) with the least index l such that x_l occurs in $\mathbf{v}_0$ p times. Then we put y_l instead of the p-th left-hand side occurence of x_l in $\mathbf{v}_0$. If this production is in P_f then we stop, otherwise we analogously follow with the new obtained word $\mathbf{v}_1$ as with $\mathbf{v}_0$.

Let us give at the end a remark. The algorithms of both classes $\mathcal{MA}_k$ and $\overline{\mathcal{MA}_k}$ finish their computations in such a moment when the last effectively used production is finished, i.e. element of P_f.

Let us formalize the above considerations.

Definition 4.3. By *a Markov-like k-algorithm $A \in \mathcal{MA}_k$ in the alphabet Σ* we mean a designated algorithm

$$A = (P, P_i, P_f, V, Contr_2, Tr_2)$$

such that $V = \Sigma^*$, $P = \{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$, where $x_i, y_i \in \Sigma^*$, for all i $(1 \leq i \leq n)$ $x_n = \varepsilon$, $y_n = \varepsilon$ and $(x_n, y_n) \in P_f$, $P_i = \{(x_1, y_1)\}$ and $Contr_2, Tr_2$ are defined as follows:

$$Contr_2((x_i, y_i), \mathbf{v}) = \begin{cases} (x_1, y_1) & \text{if } x_i \preceq_{\leq k} \mathbf{v} \text{ and } (x_i, y_i) \notin P_f \\ (x_{i+1}, y_{i+1}) & \text{if } \neg(x_i \preceq_{\leq k} \mathbf{v}) \\ \text{undefined} & \text{if } x_i \preceq_{\leq k} \mathbf{v} \text{ and } (x_i, y_i) \in P_f \end{cases}$$

$$Tr_2((x_i,y_i),\mathbf{v}) = \begin{cases} \mathbf{v}_{[1,j-1]}y_i\mathbf{v}_{[j+m,l(\mathbf{v})]} & \text{if } \mathbf{v}_{[j,j+m-1]} \text{ is at most} \\ & k \text{ occurence of } x_i \text{ in } \mathbf{v} \\ \mathbf{v} & \text{otherwise} \end{cases}$$

Definition 4.4. By *a Markov-like k-algorithm* $A \in \overline{\mathcal{MA}_k}$ *in the alphabet* Σ we mean a designated algorithm

$$A = (P, P_i, P_f, V, Contr_3, Tr_3)$$

such that $V = \Sigma^*$, $P = \{(x_1,y_1),(x_2,y_2),\dots,(x_{kn+1},y_{kn+1})\}$, where for all $i, 1 \le i \le kn+1, x_i, y_i \in \Sigma^*$, and for all $j(n < j < kn+1)$ $x_j = x_{j-n}$, $y_j = y_{j-n}$ and $x_{kn+1} = \varepsilon, y_{kn+1} = \varepsilon$, $(x_{kn+1}, y_{kn+1}) \in P_f$ and $P_i = \{(x_1,y_1)\}$ and the functions $Contr_3$ and Tr_3 are equal

$$Contr_3((x_i,y_i),\mathbf{v}) = \begin{cases} (x_1,y_1) & \text{if } x_i \preceq_l \mathbf{v}, l = k-(i-1)/n \\ & \text{and } (x_i,y_i) \notin P_f \\ (x_{i+1},y_{i+1}) & \text{if } \neg(x_i \preceq_l \mathbf{v}), l = k-(i-1)/n \\ \text{undefined} & \text{if } x_i \preceq_l \mathbf{v}, l = k-(i-1)/n \\ & \text{and } (x_i,y_i) \in P_f \end{cases}$$

$$Tr_3((x_i,y_i),\mathbf{v}) = \begin{cases} \mathbf{v}_{[1,j-1]}y_i\mathbf{v}_{[j+m,l(\mathbf{v})]} & \text{if } \mathbf{v}_{[j,j+m-1]} \text{ is } l \text{ occurence} \\ & \text{of } x_i \text{ in } \mathbf{v} \\ & \text{where } l = k-(i-1)/n \\ \mathbf{v} & \text{otherwise} \end{cases}$$

The notions of direct transformation of configurations as well as a derivation can be analogously defined for algorithms of the classes $\mathcal{MA}_k$ and $\overline{\mathcal{MA}_k}$ as for designated algorithms in Section 3 by replacing only $Contr$ and Tr by $Contr_i$ and Tr_i, $i = 2,3$ respectively.

Let us give some comments on Definition 4.4. The schema of productions P of an algorithm A is, informally speaking, a sequence $B_1, B_2, \dots, B_k, (\varepsilon,\varepsilon)$ where each $B_i, (1 \le i \le k)$ is a sequence of productions $(x_{(i-1)n+1}, y_{(i-1)n+1})$, $(x_{(i-1)n+2}, y_{(i-1)n+2})$, $\dots$,

$(x_{(i-1)n+n}, y_{(i-1)n+n})$. The j-th production of B_i has identical left and right sides (but different indexes) to left and right side (respectively) of the j-th production of B_l ($1 \leq i, l \leq k$). For brevity, the schema of productions of A will be written in the form: $B_1, (\varepsilon, \varepsilon)$.

The j-th production of B_i is applied to a transformed word $\mathbf{v}$ in such a way that $k - i + 1$ occurence of the left side of the production $(x_{(i-1)n+j}, y_{(i-1)n+j})$ of B_i is replaced by the right side ($=y_{(i-1)n+j}$), if such an occurence there exists. Otherwise we go to the next production.

Remark 4.5. Let us see that each of the classes $\mathcal{MA}_1$ and $\overline{\mathcal{MA}_1}$ are equal to the class $\mathcal{MNA}$ of Markov normal algorithms.

Remark 4.6. If the class of Markov-like k-algorithms is known then every algorithm of this class can be defined by means of a schema of productions.

Example 4.7. Let us consider the following mapping $f_k : N^m \mapsto N^{m-1}$

$$f_k(x_1, x_2, \dots, x_m) = \begin{cases} (x_1, \dots, x_k + x_{k+1}, \dots, x_m) & k < m \\ (x_1, \dots, x_{m-1} + x_m) & k \geq m \end{cases}$$

Let us assume the convention that every m-tuple $(x_1, \dots, x_m)$ will be written in the form:

$$\underbrace{1 \dots 1}_{x_1} | \underbrace{1 \dots 1}_{x_2} | \dots | \underbrace{1 \dots 1}_{x_m} .$$

Now we define the algorithm $A \in \mathcal{MNA}$ over the alphabet $\Sigma = \{1, |\}$ which computes the fuction f_k. We define new alphabet $\Sigma' = \Sigma \cup \{\gamma_1, \dots, \gamma_k, \lambda\}$. Schema of productions P has the form:
$P = ((\gamma_1 |, |\gamma_2), (\gamma_1 1, 1\gamma_1),$
$(\gamma_2 |, |\gamma_3), (\gamma_2 1, 1\gamma_2),$
$\vdots$
$(\gamma_{k-1} |, |\gamma_k), (\gamma_{k-1} 1, 1\gamma_{k-1}),$
$(\gamma_k |, \varepsilon), (\gamma_k 1, 1\gamma_k),$
$(\gamma_1 \varepsilon, \lambda), \dots, (\gamma_k \varepsilon, \lambda),$

$(1\lambda, \lambda 1), (|\lambda, \varepsilon),$
$(\varepsilon, \varepsilon));$
$P_i = \{(\gamma_1|, |\gamma_2)\}$, $\quad P_f = \{(\gamma_k|, \varepsilon), (|\lambda, \varepsilon), (\varepsilon, \varepsilon)\}$.

Now we define the algorithm $A' \in \overline{\mathcal{MA}_k}$ in the alphabet $\Sigma = \{1, |\}$ which computes this mapping. In this case $P = \{(|, \varepsilon), (\varepsilon, \varepsilon)\}, P_i = \{(|, \varepsilon)\}, P_f = \{(|, \varepsilon), (\varepsilon, \varepsilon)\}$.

One can see that algorithm $A' \in \overline{\mathcal{MA}_k}$ has fewer productions than $A \in \mathcal{MNA}$.

Example 4.8. Let us consider three 3-algorithms $A \in \mathcal{MA}_3$, $A' \in \overline{\mathcal{MA}_3}$ and $A'' \in \mathcal{MNA}$ in the alphabet $V = \{0, 1\}$ with the same schema of productions of the form: $P = \{(0, 1), (10, 01), (1, 0)\}$ with $(1, 0) \in P_f$.
Then the sequences

$$\begin{aligned} \mathbf{c} &= 1010, 1011, 1111, 1101 \\ \mathbf{c}' &= 1010, 1011, 1010 \\ \mathbf{c}'' &= 1010, 1110, 1111, 0111 \end{aligned}$$

are the computations of A, A' and A'', respectively.

5. Equivalence of the classes $\mathcal{MA}_k$, $\overline{\mathcal{MA}_k}$, and $\mathcal{MNA}$ of algorithms.

It will be shown that every class of functions computable by the algorithms of $\mathcal{MA}_k$ and $\overline{\mathcal{MA}_k}$ is equal to the class of functions computable by Markov normal algorithms.

At the beginning let us introduce some notations.

For $\mathbf{x} = x_1 x_2 \ldots x_n \in \Sigma^*$ let $In(\mathbf{x}, a)$ denote a word $x_1 a x_2 \ldots x_n$. A schema of productions: $(xa_1y, xy), (xa_2y, xy), \ldots, (xa_ny, xy)$, where $\Sigma = \{a_1, \ldots, a_n\}$ will be briefly denoted by $(x\alpha y, xy), \alpha \in \Sigma$. The additional symbols ($\notin \Sigma$) will be denoted by $\gamma, \xi, \chi, \psi, \lambda$ (with or without subscripts and superscripts).

Now we give two theorems on equivalence of the classes $\mathcal{MA}_k$, $\overline{\mathcal{MA}_k}$ and $\mathcal{MNA}$.

Theorem 5.1. (1) *For every k-algorithm $A \in \mathcal{MA}_k$ in an alphabet Σ there exists an equivalent algorithm $M \in \mathcal{MNA}$ over an alphabet Σ such that $A(\mathbf{v}) = M(\mathbf{v})$ for every $\mathbf{v} \in \Sigma^*$;*

(2) *For every algorithm $M \in \mathcal{MNA}$ in an alphabet Σ there exists an equivalent algorithm $A_1 \in \mathcal{MA}_k$ over an alphabet Σ such that $A_1(\mathbf{v}) = M(\mathbf{v})$ for every $\mathbf{v} \in \Sigma^*$.*

Proof. First we prove (1).

Let $\Sigma' = \Sigma \cup \{{\gamma_j}^i, \lambda^i\})$ where $1 \le i \le n$ and $1 \le j \le k$. For an algorithm $A \in \mathcal{MA}_k$ with the schema of productions $(x_1, y_1), \ldots, (x_n, y_n)$ we create for each production (x_i, y_i) an auxiliary block of productions Sr_i of the form:

$({\gamma_j}^i x_i, In(x_i, {\gamma_{j+1}}^i))$ for $j < k$

$({\gamma_k}^i x_i, y_i)$ this production is final iff (x_i, y_i) is final

$({\gamma_j}^i \alpha, \alpha {\gamma_j}^i)$ for $j < k$

$({\gamma_j}^i \varepsilon, \lambda^i)$ for $j < k$

$(\lambda^i x_i, y_i)$

$(\alpha\lambda^i, \lambda^i\alpha)$

$(\varepsilon\lambda^i, {\gamma_1}^{i+1})$.

This determines a Markov normal algorithm $M = (Sr_1, Sr_2, \ldots, Sr_n, ({\gamma_1}^{n+1}, \varepsilon), (\varepsilon, {\gamma_1}^1))$ (the production $({\gamma_1}^{n+1}, \varepsilon)$ is final) over Σ such that $M(\mathbf{v}) = A(\mathbf{v})$, for arbitrary $\mathbf{v} \in \Sigma^*$.

To prove (2) let us consider a Markov normal algorithm M in the alphabet Σ with a schema of productions of the form: $(x_1, y_1), \ldots, (x_n, y_n)$.

Let us assign to each j-th production $(1 \le j \le n)$ a schema of productions Sr_j (in the alphabet $\Sigma' = \Sigma \cup \{\xi_1, \ldots, \xi_{n+1}\} \cup \{\gamma_1, \ldots, \gamma_n\})$ of the form:

$(\xi_j x_j, y_j)$ this production is final iff (x_j, y_j) is final

$(\xi_j \alpha, \alpha \xi_j)$ $(\alpha \in \Sigma)$

$(\xi_j \varepsilon, \gamma_j)$

$(\alpha\gamma_j, \gamma_j\alpha)$

$(\varepsilon\gamma_j, \xi_{j+1})$

Let $A_1 \in \mathcal{MA}_k$ be an algorithm with a schema of productions $(Sr_1, Sr_2, \ldots, Sr_k, (\xi_{k+1}, \varepsilon), (\varepsilon, \xi_1))$ $((\xi_{k+1}, \varepsilon) \in P_f)$. One can easily see that $A_1(v) = M(\mathbf{v})$, for arbitrary $\mathbf{v} \in \Sigma^*$. ■

Theorem 5.2. (1) *For every k-algorithm $A \in \overline{\mathcal{MA}_k}$ in an alphabet Σ there exists an equivalent algorithm $M \in \mathcal{MNA}$ over an alphabet Σ such that $A(\mathbf{v}) = M(\mathbf{v})$ for every $\mathbf{v} \in \Sigma^*$;*

(2) *For every algorithm $M \in \mathcal{MNA}$ in an alphabet Σ there exists an equivalent algorithm $A_2 \in \overline{\mathcal{MA}_k}$ over an alphabet Σ such that $A_2(\mathbf{v}) = M(\mathbf{v})$ for every $\mathbf{v} \in \Sigma^*$.*

Proof. First we prove (1).

Let $\Sigma' = \Sigma \cup \{\gamma_j{}^{l,i}, \lambda^i\}\}$ where $1 \leq i \leq n$ and $1 \leq j, k \leq k$. For an algorithm $A \in \overline{\mathcal{MA}_k}$ with the schema of productions $(x_1, y_1), \dots, (x_n, y_n)$ (productions are without repetition, see comments after Definition 4.4) we create for each production (x_i, y_i) an auxiliary block of productions $Sr_i{}^l$ of the form:

$(\gamma_j{}^{l,i} x_i, In(x_i, \gamma_{j+1}{}^{l,i}))$	for $j < k$
$(\gamma_k{}^{l,i} x_i, y_i)$	this production is final iff (x_i, y_i) is final
$(\gamma_j{}^{l,i} \alpha, \alpha \gamma_j{}^{l,i})$	for $j < k$
$(\gamma_j{}^{l,i} \varepsilon, \lambda^{l,i})$	for $j < k$
$(\alpha \lambda^{l,i}, \lambda^{l,i} \alpha)$	
$(\varepsilon \lambda^{l,i}, \gamma_1{}^{l,i+1})$	if $i + 1 = n$ then right side $= \gamma_1{}^{l-1,1}$.

Let $M \in \mathcal{MNA}$ be an algorithm with a schema of productions $(Sr_1{}^k, Sr_2{}^k, \dots, Sr_n{}^k, Sr_1{}^{k-1}, Sr_2{}^{k-1}, \dots, Sr_n{}^{k-1}, \dots, Sr_1{}^1, Sr_2{}^1, \dots, Sr_n{}^1, (\gamma_1{}^{0,1}, \varepsilon), (\varepsilon, \gamma_1{}^{k,1})), ((\gamma_1{}^{0,1}, \varepsilon) \in P_f)$. One can easily see that $A(v) = M(\mathbf{v})$, for arbitrary $\mathbf{v} \in \Sigma^*$.

To prove (2) let us consider an algorithm M of $\mathcal{MNA}$ in the alphabet Σ with the schema of productions of the form: $(x_1, y_1), \dots, (x_n, y_n)$.

Let us construct an algorithm $A_2 \in \overline{\mathcal{MA}_k}$ in the alphabet $(\Sigma' = \Sigma \cup \{\xi_1, \dots, \xi_{k+1}\} \cup \{\gamma_1, \dots, \gamma_k\})$ as follows.

Let us assign to the j-th production (x_j, y_j) $(1 \leq j \leq n)$ the following schema of productions Sr_j:

$(\xi_j \alpha, \alpha \xi_j)$
$(\beta_j \alpha, \alpha \beta_j)$
$(\xi_j \varepsilon, \gamma_j)$ $\quad (\alpha \in \Sigma)$
$(\beta_j \varepsilon, \varepsilon)$
$(\alpha \gamma_j, \gamma_j \alpha)$
$(\varepsilon \gamma_j, \xi_{j+1})$.

Let $A_2 \in \overline{\mathcal{MA}_k}$ be an algorithm with a schema of productions (without repetition) of the form: $((\xi_1 x_1, \zeta_1), \ldots, (\xi_n x_n, \zeta_n), Sr_1, Sr_2, \ldots, Sr_n, (\varepsilon\gamma_n, \varepsilon) \in P_f, (\varepsilon, \xi_1))$, where $\zeta_i = y_i\beta_i$ if $i \notin P_f$ and otherwise $\zeta_i = y_i$ and this production is final.

One can easily see that $A_2(\mathbf{v}) = M(\mathbf{v})$, for arbitrary $\mathbf{v} \in \Sigma^*$. ■

6. Closure properties of the classes $\mathcal{MA}_k$ and $\overline{\mathcal{MA}_k}$.

Analogously as for Markov normal algorithms one can define the operations on the k-algorithms of respective class, such as composition, ramification, propagation and iteration.

Only for illustration the operation of composition will be recalled after Mendelson [5], the remaining operations can be found in [5].

Let A_1 and A_2 be two k-algorithms in the alphabet Σ of $\mathcal{MA}_k \cup \overline{\mathcal{MA}_k}$. A k-algorithm A in the alphabet Σ is said to be obtained from A_1 and A_2 by operation of ***composition*** $(A_1 \circ A_2)$ iff $A(\mathbf{v}) = A_1(A_2(\mathbf{v}))$ for arbitrary $\mathbf{v} \in \Sigma^*$.

Lemma 6.1. *For every $k \geq 1$, $(\mathcal{MA}_k, \circ)$ and $(\overline{\mathcal{MA}_k}, \circ)$ form the semigroup with unity. The empty algorithm is the unity of these semigroups.*

Theorem 6.2. *The classes $\mathcal{MA}_k$ and $\overline{\mathcal{MA}_k}$ are closed under the following operations:*
(1) *composition;*
(2) *ramification;*
(3) *propagation;*
(4) *iteration.*

Proof immediately follows from the fact that $\mathcal{MNA}$ is closed under the above operations (see [5], pp. 214-218) and from Theorems 6.1 and 6.2.

7. Final remarks.

Only one aspect of the equivalence problem of the classes $\mathcal{MA}_k$ and $\overline{\mathcal{MA}_k}$ and $\mathcal{MNA}$ has been examined. The same aspect of the equivalence problem will be continued for the other classes of algorithms.

The authors' next paper will be devoted to the equivalence of the classes $\mathcal{RMA}_k$ and $\overline{\mathcal{RMA}_k}$ of right-hand side Markov-like k-algorithms and the class $\mathcal{MNA}$ of Markov normal algorithms.

The complexity problem of algorithms and their computations remains open till now. It would be interesting to compare the ability of algorithms of particular classes for computations of Boolean functions.

References

[1] Cutland, N. J., *Computability and introduction to recursive function theory*, Cambridge University Press,, Cambridge, London, New York, Sydney, Melbourne, 1980.

[2] Golomb, S.W., *Shift-register sequences*, Aegen Park Press, Laguna Hills,, 1982.

[3] Markov, A., *The Theory of Algorithms (in Russian)*, Trudy Mat. Inst. Steklov. **XLII** (1954).

[4] Mazurkiewicz, A., *Foundation of programming theory*, Problemy Przetwarzania Informacji, Wydz. Naukowo-Tech., Warszawa 1974,, 39–94.

[5] Mendelson, E., *Introduction to Mathematical Logic*, The University Series in Mathematics, Princeton, 1964.

[6] Pawlak, Z., *Stored program computers*, Algorytmy **10** (1969), 7–22.

Authors' addresses:
Institute of Management and Foundation of Technics
Department of Applied Mathematics
Technical University of Lublin
ul. Bernardyńska 13
20-950 Lublin, Poland

Jerzy Mycka
Institute of Mathematics
Mariae Curie-Sklodowska University
pl. M. Curie-Skłodowskiej 1
20-031 Lublin, Poland

received March 16, 1995