If $i = 1$ and $j \in \Phi\, \zeta_{A,1}$, then we have $\Phi \in \overline{P}_j^1$, because $\Phi\, \zeta_{A,1} = ( \quad )(\Phi\, \psi_{A,0}^{0,1})$.

**Definition 2.9.** *The family* $(X_i)$, $i = 0, 1, 2, \ldots$, $X_i \subseteq C_i^0$ *is called metaalgebraic of order* 1 *with respect to* $\hat{C}_0$ *if there exist* $n$ *and a* $\omega$*-polynomial* $P_{n,n}$ *of degree* 1, *such that* $X_i = \overline{P}_1^i$, $i = 0, 1, 2, \ldots$

**Theorem 2.10.** *A metaalgebraic family* $X = (X_i)$, $i = 0, 1, 2, \ldots$ *of order* 1 *with respect to* $\hat{C}_0$ *is the behaviour of any restricted* $VT$*-automaton over* $S_0[\Omega]$.

*Proof.* The $VT$-automaton $(A, t)$ over $S_0[\Omega]$ is called restricted, if $A$ is a restricted $VT$-algebra over $S_0[\Omega]$. Let $P_{n,n}$ be a polynomial of degree 1 such that $X_i = \overline{P}_1^i$, $i = 0, 1, 2, \ldots$ By the Theorem 2.6 there exists a $\omega$-polynomial $T_{n,n}$, such that $\overline{\mathfrak{L}}_{\hat{C}_0} = \overline{\mathfrak{T}}_{\hat{C}_0}$, therefore $X_i = \overline{T}_1^i$, $i = 0, 1, 2, \ldots$ Consider the relational $VT$-algebra $A = (A_i, \alpha_i, \beta_i)$, $A_i = [n]$, $\alpha_i = \beta_i = 1_{[n]}$, $i = 0, 1, 2, \ldots$ corresponding to $T_{n,n}$. By the Theorem 2,8, we have $\overline{T}_j^i = j\, \zeta_{A,i}^{-1}$, $1 \leqslant j \leqslant n$, $i = 0, 1, 2, \ldots$, therefore $X_i = 1\, \zeta_{A,i}^{-1}$. If $t_i = \{1\} \subseteq [n]$, $i = 0, 1, 2, \ldots$, it results that the family $X = (X_i)$, $i = 0, 1, 2, \ldots$ is the behavior of $(A, t)$.

**Theorem 2.11.** *The behavior of the restricted* $VT$*-automaton* $(A, t)$ *over* $S_0[\Omega]$, $A_i =$ *constant*, $\alpha_i = \beta_i = 1_{A_i}$, $t = (t_i)$, $t_i =$ *constant*, $i = 0, 1, 2, \ldots$ *is a metaalgebraic family with respect to* $\hat{C}_0$.

*Proof.* We may assume that $A_i = [n]$, $t_i = [k]$, $k \leqslant n$.

By the Proposition 2.4 there exists a $\omega$-polinomial $P_{n,n}$ of degree 1 corresponding to $A$. We have $\beta((A, t)) = (X_i)$, $i = 0, 1, 2, \ldots$ where $X_i = t_i\, \zeta_{A,i}^{-1} = \bigcup_{h \in t_i = [k]} h\, \zeta_{A,i}^{-1}$. By the Theorem 2.8 $h\, \zeta_{A,i}^{-1} = \overline{P}_h^i$, $1 \leqslant j \leqslant n$, $i = 0, 1, 2, \ldots$ The family $(\overline{P}_1^0, \overline{P}_1^1, \ldots)$ is metaalgebraic with respect to $\hat{C}_0$. By the Proposition 2.2 it results that $X = (X_i)$, $X_i = \bigcup_{h=1}^{k} \overline{P}_h^i$, $i = 0, 1, 2, \ldots$ is metaalgebraic with respect to $\hat{C}^0$.

## REFERENCES


1. Eilenberg, S. Wright, J. — *Automata in General Algebras*, Information and Control, vol. 11, nr. 4, 1967, 452—470.
2. Felea, V. — *Finite Automata with a variable structure in VT-algebras*, to apear in An. şt. Univ. Iaşi.
3. Felea, V. — *Relational VT-algebras over free theories*, to apear.
4. Give'on, Y., Arbib, M.A. — *The Categorical Framework for Dynamic Analysis*, Information and Control, vol. 12, nr. 4, 1968, 364 — 370.


*Received 19.X.1978*

*Faculty of Mathematics*
*University of Iaşi*
*Iaşi, R.S. Romania*

Analele ştiinţifice ale Universităţii „Al. I. Cuza" din Iaşi
Tomul XXVI, s. I a, 1980, fasc. 1

# ON THE COMPLETNESS IN ABSTRACT COMPLEXITY CLASSES OF LANGUAGES

BY

GH. GRIGORAŞ

**1. Introduction.** The importance of polynomial complete languages in concrete computational complexity has been pointed out by Cook [2] and Karp [5]. Book [3] shows the relationships between complexity classes defined by time-(space-) bounded multitape Turing machines with the bounding functions $n$, $n^k$, $2^n$, $2^{nk}$.

The goal of this paper is to show that some results of Book [3], are true in a more general context. We consider an „abstract complexity class" of languages to be a class of languages accepted by (non) deterministic Turing machines which run in time (space) bounded by functions from a class $\mathcal{F}$. We look for the conditions which must be satisfied by $\mathcal{F}$ and by the reductibilities, to obtain the Book-analogous relationships between „abstract complexity classes".

**2. Preliminaries.** Let $\Sigma$ be a finite nonempty set called alphabet. The set of all strings (words) of elements from $\Sigma$ (including the nul string $\Lambda$) will be denoted by $\Sigma^*$ (the free semigroup generated by $\Sigma$). The legth of a string $w \in \Sigma^*$ will be $|w|$ (which means the number of symbols of $w$). If $\sigma \in \Sigma$, the string $\sigma\sigma \ldots \sigma$ of length $m$, will be designed by $\sigma^m$. A language is a subset $L$ of $\Sigma^*$.

We assume that the reader is familiar with the concept of multitape (non) deterministic Turing machine ([1] pag. 25, [4] pag 80). For a Turing machine $M$ with the input alphabet $\Sigma$, we denote by $L(M)$, $L(M) \subseteq \Sigma^*$, the language accepted by $M$.

Let $N$ be the set of positive integers.

**Definition 1.** *A function* $f: N \to N$ *is self computable* ([3]) *if there exists a deterministic Turing machine* $M_1$, *which, upon input* $w$, *runs for precisely* $f(|w|)$ *steps and halts, and a deterministic Turing machine* $M_2$, *which, upon input* $w$, *marks precisely* $f(|w|)$ *consecutive tape squares and halts.*

**Definition 2.** *A function* $f: N \to N$ *is a bounding function if*:

i) $f$ *is nondecreasing*;

ii) $f$ *is self-computable.*

Let $M$ be a multitape Turing machine and $f$ a bounding function. We say that $M$ *operates within time* (*space*) *bound* $f$ if, for each input string $w$, if $w \in L(M)$ there exists an accepting computation of $M$ on $w$ which has no more than $f(|w|)$ steps (visits no more than $f(|w|)$ tape squares of every tape).

Now we make the following notations:

$(N)$ *D T I M E* $(f)$ $= \{L(M)/M$ is a (non)deterministic multitape Turing machine which operates within time bound $f\}$;

$(N)$ *D S P A C E* $(f)$ $= \{L(M)/M$ is a (non) deterministic multitape Turing machine which operates within space bound $f\}$.

Let $\Sigma$ and $\Delta$ be two finite alphabets. We denote $\mathcal{F}(\Sigma, \Delta) = \{f \mid f : \Sigma^* \to \Delta^*\}$ and $\Sigma^* = \mathrm{Dom}(f)$ if $f : \Sigma^* \to \Delta^*$.

**Definition 3.** [3] *Let $f \in \mathcal{F}(\Sigma, \Delta)$. The language $L_1 \subseteq \Sigma^*$ is f-reducible to $L_2 \subseteq \Delta^*$, if for every $w \in \Sigma^*$, $w \in L_1$ if and only if $f(w) \in L_2$.*

**Definition 4.** [3] *Let $\mathfrak{C}$ be a class of functions on strings and $\mathfrak{L}$ a class of languages; a language $L_0$ is $\mathfrak{C}$-complete for $\mathfrak{L}$ if:*

i) $L_0 \in \mathfrak{L}$;

ii) *for each $L \in \mathfrak{L}$ there exists a function $f \in \mathfrak{C}$ such that $L$ is f-reducible to $L_0$.*

C o o k [2] and K a r p [5] have shown the existence of „polynomial complete" languages (problems) (the class $\mathfrak{C}$ is the class of functions computed by deterministic Turing machines in polynomial time) among combinatorial problems. The importance of „polynomial complete" problems derives from the fact that, if any of these problems can be solved in polynomial time, then they all can.

**3. Reducibilities and complete sets in abstract complexity classes of languages.** In this section we define an *abstract complexity class of languages* and we look for the condition to be satisfied in order that a complete set for an abstract complexity class $\mathfrak{L}_1$, remains complete for a class $\mathfrak{L}_2$, $\mathfrak{L}_1 \subset \mathfrak{L}_2$.

**Definition 5.** *Let $\mathcal{T}$ be a class of bounding functions and $\mathfrak{C}$ a class of functions on strings. The class $\mathfrak{C}$ is $\mathcal{T}$-computable, if for each $f \in \mathfrak{C}$ there exists $t \in \mathcal{T}$ and a deterministic Turing machine $M$, such that $M$ computes $f$ within time bound $t$ (for each $w \in \mathrm{Dom}(f)$, $M$ compute $f(w)$ in no more than $t(|w|)$ steps).*

First, let us establish some results concerning $\mathcal{T}$-computable functions.

**Lemma 1.** *Let $\mathfrak{C}$ be a class of functions on strings and $\mathcal{T}$ a class of bounding functions. If:*

i) *$\mathfrak{C}$ is $\mathcal{T}$-computable;*

ii) *$\mathcal{T}$ is closed under composition and*

iii) *for each $t_1, t_2 \in \mathcal{T}$ there exists $t_3 \in \mathcal{T}$ such that $t_1(x) + t_2(x) \leqslant t_3(x)$, $\forall x \in N$, then, for each $t_1, t_2 \in \mathcal{T}$ and $f \in \mathfrak{C}$, there exists $t_3 \in \mathcal{T}$ such that $\forall w \in \mathrm{Dom}(f)$:*

$$t_1(|w|) + t_2(|w|)) \leqslant t_3(|w|).$$

*Proof.* Since $f$ is $\mathcal{T}$-computable, there exists a deterministic Turing machine $M$ and a function $t \in \mathcal{T}$ such that $M$ computes $f$ in time bound $t$. Hence, for each $w \in \mathrm{Dom}(f)$, $|f(w)| \leqslant t(|w|)$. Now, for each $t_1, t_2 \in \mathcal{T}$ we have:

$$t_1(|w|) + t_2(|f(w)|) \leqslant t_1(|w|) + t_2(t(|w|)) = t_1(|w|) + t_2 \circ t(|w|) = t_1(|w|) + $$
$$+ t_2(|w|) \leqslant t_3(|w|),$$

(from the conditions ii) and iii)). q.e.d.



**Corollary 1.** *If $\mathfrak{C}$ is a class of polynomially-computable functions on strings and $\mathcal{T}$ is the class of polynomials, then for every two polynomials $p_1$, $p_2$ and $f \in \mathfrak{C}$, there exists $p_3 \in \mathcal{T}$ such that*

$$\forall w \in \mathrm{Dom}(f),\ p_1(|w|) + p_2(|f(w)|) \leqslant p_3(|w|).$$

*Proof.* The proof results from the fact that the class of polynomials are closed under composition and addition and from the above lemma.

**Lemma 2.** *Let $\mathfrak{C}$ be the class of polynomially-computable functions on strings and $\mathcal{T} = \{2^p \mid p$ is a polynomial$\}$. Then, for each $t_1, t_2 \in \mathcal{T}$ and $f \in \mathfrak{C}$, there exists $t_3 \in \mathcal{T}$ such that:*

$$\forall w \in \mathrm{Dom}(f),\ t_1(|w|) + t_2(|f(w)|) \leqslant t_3(|w|).$$

*Proof.* Let $t_1, t_2 \in \mathcal{T}$ and $f \in \mathfrak{C}$, $t_1(x) = 2^{p_1(x)}$ and $t_2(x) = 2^{p_2(x)}$, $p_1$ and $p_2$ are polynomials. Since $f$ is polynomially-computable, there exists a polynomial $p$ such that $|f(w)| \leqslant p(|w|)$, $\forall w \in \mathrm{Dom}(f)$.
Now:

$$2^{p_1(|w|)} + 2^{p_2(|f(w)|)} \leqslant 2^{p_1(|w|)} + 2^{p_2 \circ p(|w|)} \leqslant 2^{p_1(|w|) + p_2'(|w|)} = 2^{p_3(|w|)} \quad \text{where}$$

$$p_2' = p_2 \circ p \text{ and } p_3 = p_1 + p_2' ;\ \text{q. e. d.}$$

**Lemma 3.** *Let $\mathfrak{C}$ be a class of homomorphisms on strings and $\mathcal{T} = \{2^{cn} \mid c \in N\}$. Then for each $t_1, t_2 \in \mathcal{T}$ and $f \in \mathfrak{C}$ there exists $t_3 \in \mathcal{T}$ such that $t_1(|w|) + t_2(|f(w)|) \leqslant t_3(|w|)$, $\forall w \in \mathrm{Dom}(f)$.*

*Proof.* Let $f \in \mathfrak{C}$ and $\Sigma^* = \mathrm{Dom}(f)$, $\Sigma = \{\sigma_1, \ldots, \sigma_n\}$. Let us consider $k = \max\{|f(\sigma_i)|,\ i = 1, 2, \ldots, n\}$. Then, for each $t_1, t_2 \in \mathcal{T}$, $t_1(x) = 2^{c_1 x}$, $t_2(x) = 2^{c_2 x}$, we have:

$$t_1(|w|) + t_2(|f(w)|) = 2^{c_1|w|} + 2^{c_2|f(w)|} \leqslant 2^{c_1|w|} + 2^{c_2 k|w|} \leqslant 2^{(c_1 + c_2 k)|w|}$$

Then considering $t_3(x) = 2^{(c_1 + c_2 k)x}$, the result follows.

Let $\mathcal{T}$ be a class of bounding functions. We make the following notations:

$(N)$ D T I M E $(\mathcal{T})$ $= \{(N)$D T I M E $(t) \mid t \in \mathcal{T}\}$.

$(N)$ D S P A C E $(\mathcal{T})$ $= \{(N)$ D S P A C E $(t) \mid t \in \mathcal{T}\}$.

**Definition 6.** *An „abstract complexity class" of bound $\mathcal{T}$, denoted $ACC(\mathcal{T})$, is one of the classes $(N)$D T I M E$(\mathcal{T})$ or $(N)$D S P A C E$(\mathcal{T})$.*

If $L \in ACC(\mathcal{T})$, there exists a function $f \in \mathfrak{C}$ and a (non) deterministic Turing machine $M$ such that $M$ accepts $L$ in time (or space) bounded by $t$. We say that the language $L$ (the Turing machine $M$) has the complexity $t$.

**Theorem 1.** *Let $\mathcal{T}$ be a class of bounding functions, and $\mathfrak{C}$ a class of functions on strings. If $L_0 \in ACC(\mathcal{T})$ and $L$ is a language such that for some $f \in \mathfrak{C}$ $L$ is f-reducible to $L_0$ then the following conditions are sufficient for $L \in ACC(\mathcal{T})$:*

i) *$\mathfrak{C}$ is $\mathcal{T}$-computable;*

ii) *for each $t_1, t_2 \in \mathcal{T}$ and $f \in \mathfrak{C}$, there exists $t_3 \in \mathcal{T}$ such that $t_1(|w|) + t_2(|f(w)|) \leqslant t_3(|w|)$, $\forall w \in \mathrm{Dom}(f)$.*

*Proof.* Let $\Sigma$, $\Delta$ be finite alphabets such that $L \subseteq \Sigma^*$ and $L_0 \subseteq \Delta^*$. Since $L_0 \in ACC(\mathcal{T})$ there exists a Turing machine $M_0$ such that $L(M_0) =$

$= L_0$ and $M_0$ has the complexity $t_0 \in \mathfrak{F}$. $L$ is $f$-reducible to $L_0$, hence $w \in L$ iff $f(w) \in L_0$. Since $\mathfrak{C}$ is $\mathfrak{F}$-computable, there exists a deterministic Turing machine $M_1$ such that $M_1$ computes $f(w)$ in time (and, of course, in space) bounded by $t_1(|w|)$, $t_1 \in T$. Let us consider a new Turing machine $M$ which operates in the following manner:

— for the input $w \in \Sigma^*$, $M$ simulates $M_1$'s operations, hence, in time bounded by $t_1(|w|)$ $M$ computes $f(w)$;

— now $M$ simulates the action of $M_0$ upon the string $f(w)$.

It is clear that $L(M) = L$ because $w \in L$ iff $f(w) \in L_0$. Now the amount of time (or space) necessary for the accepting of $w$ is $t_1(|w|) + t_0(|f(w)|)$. From the condition *ii*) we conclude that there exists $t \in \mathfrak{F}$ such that $t_1(|w|) + t_0(|f(w)|) \leqslant t(|w|)$, $\forall w \leqslant \Sigma^*$, hence $L \in ACC(\mathfrak{F})$. q.e.d.

**Corollary 2.** *Let $\mathfrak{L}$ be a class of languages and $L_0$ a language $\mathfrak{C}$-complete for $\mathfrak{L}$. If the conditions of theorem 1 hold, then $\mathfrak{L} \subseteq ACC(\mathfrak{F})$ if and only if $L_0 \in ACC(\mathfrak{F})$.*

*Proof.* If $\mathfrak{L} \subseteq ACC(\mathfrak{F})$, then $L_0 \in ACC(\mathfrak{F})$ due to the completness of $L_0$ for $\mathfrak{L}$.

Conversely, let $L_0 \in ACC(\mathfrak{F})$ and $L_0$ is $\mathfrak{C}$-complete for $\mathfrak{L}$. Then, for every $L \in \mathfrak{L}$, there exists a function $f \in \mathfrak{C}$ such that $L$ is $f$-reducible to $L_0$. From Theorem 1 we conclude $L \in ACC(\mathfrak{F})$, hence $\mathfrak{L} \subseteq ACC(\mathfrak{F})$.

**Remark 1.** There exists classes $\mathfrak{C}$ and $\mathfrak{F}$ which satisfy the conditions of Theorem 1 (see Lemmas 1—3).

**Remark 2.** Lemma 3.1 and Lemma 3.2 from Book [3] can be obtained from Theorem 1 taking some special classes $\mathfrak{C}$, $\mathfrak{F}$ and $ACC(\mathfrak{F})$.

**Definition 7.** *Let $\mathfrak{F}_1$ and $\mathfrak{F}_2$ be two classes of bounding functions. We say that $ACC(\mathfrak{F}_1)$ and $ACC(\mathfrak{F}_2)$ have the same type, if both classes are refered as (non)deterministic Turing machines and time (tape) complexity.*

**Theorem 2.** *Let $\mathfrak{F}_1$ and $\mathfrak{F}_2$ be two classes of bounding functions such that $ACC(\mathfrak{F}_1) \subseteq ACC(\mathfrak{F}_2)$. These classes are supposed to have the same type. Let $\mathfrak{C}_1$ and $\mathfrak{C}_2$ be two classes of functions on strings. If:*

i) *there exists $t_0 \in \mathfrak{F}_1$ such-that $t_0(n) \geqslant 2n$, $\forall n \in N$;*

ii) *$\mathfrak{C}_2$ includes, the class of $\mathfrak{F}_2$-computable functions;*

iii) $\mathfrak{C}_1 \circ \mathfrak{C}_2 \subseteq \mathfrak{C}_2$*).

*then, if $L_0$ is $\mathfrak{C}_1$-complete for $ACC(\mathfrak{F}_1)$ it follows that $L_0$ is $\mathfrak{C}_2$-complete for $ACC(\mathfrak{F})$.*

*Proof.* Let $L_0$ be a language $\mathfrak{C}_1$ - complete for $ACC(\mathfrak{F}_1)$. Let us prove that, for each $L \in ACC(\mathfrak{F}_2)$ there exists $f_2 \in \mathfrak{C}_2$ such that $L$ is $f_2$-reducible to $L_0$. Since $L \in ACC(\mathfrak{F}_2)$, there exists $t_2 \in \mathfrak{F}_2$ and a Turing machine $M$ of complexity $t_2$ with $L(M) = L$. Let $\Sigma$ be the input alphabet for $M$, $L \subseteq \Sigma^*$, and let us consider the function $f: \Sigma^* \to (\Sigma \cup \{c\})^*$, where $c \notin \Sigma$, defined by $f(w) = wc^{t_2(|w|)-|w|}$. We have $|f(w)| = t_2(|w|)$ and it is obvious that $f$ is $\mathfrak{F}_2$-computable. We claim that the language $L_1 = f(L) = \{f(w) | w \in L\}$ is in $ACC(\mathfrak{F}_1)$. To show this we construct a Turing machine $M_1$, which for the input $u = wc^m$ performs the following operations:

— $M_1$ checks whether $|wc^m| = t_2(|w|)$ and, if so

— $M_1$ simulates $M$'s operations upon $w$ to determine whether of not $w \in L$.

*) $\mathfrak{C}_1 \circ \mathfrak{C}_2 = \{g | g = f_1 \circ f_2,\ f_1 \in \mathfrak{C}_1,\ f_2 \in \mathfrak{C}_2,\ f_2: \Sigma^* \to \Delta^*,\ f_1: \Delta^* \to \Gamma^*\}$



Since $t_2$ is a self-computable function, the total amount of time (or space) needed by $M_1$ to accept $u = wc^m$ is $|u| + t_2(|w|) = 2|u|$. Thus, $L(M) \in ACC(t_0) \subseteq ACC(\mathfrak{F}_1)$. Clearly, $L_1 = L(M)$, hence $L_1 \in ACC(\mathfrak{F}_1)$. Now, $L_0$ is $\mathfrak{C}_1$-complete for $ACC(\mathfrak{F}_1)$; therefore there exists $f_1 \in \mathfrak{C}_1$, such that $\mathfrak{L}_1$ is $f_1$-reducible to $L_0$. But $L$ is $f$-reducible to $L_1$, hence $L$ is $f_1 \circ f$-reducible to $L_0$. Let us denote $f_2 = f_1 \circ f$. We have $f_2 \in \mathfrak{C}_2$, taking into account the 3rd condition.

We conclude that $L_0$ is $\mathfrak{C}_2$-complete for $ACC(\mathfrak{F}_2)$.

**Remark 3.** *Lemma 3.3 from Book [3] can be obtained from Theorem 2 by taking appropriate classes $\mathfrak{C}_1$, $\mathfrak{C}_2$, $\mathfrak{F}_1$ and $\mathfrak{F}_2$.*

**Remark 4.** *Let us denote $\mathfrak{C}_{\mathfrak{F}_2}$ the class of functions on strings which are $\mathfrak{F}_2$-computable and $\hat{\mathfrak{C}}_1$ the closure of $\mathfrak{C}_1$ under composition. Then, the smallest class $\mathfrak{C}_2$ which satisfy the conditoons* ii) *and iii) from Theorem 2 is $\mathfrak{C}_2 = \hat{\mathfrak{C}}_1 \circ \mathfrak{C}_{\mathfrak{F}_2} \cup \mathfrak{C}_{\mathfrak{F}_2}$* [6].

## REFERENCES


1. Aho, A. J. Hopcroft, J. Ullman — *The Design and Analysis of Computer Algorithms*, Addison-Wesley, Reading Massachusetts, 1974.
2. Cook S. — *The complexity of theorem proving procedures*, Proc. Third ACM Symposium on Theory of Computing (1971), 151—158.
4. Book R. — *Comparing Complexity Classes*. J. Computer Systems Sci, 9 (1974), 213—229.
3. Hopcroft, J., Ullman J. — *Formal languages and their relations to Automata*. Addison Wesley, Reading Massachusetts, 1969.
5. Karp R. — *Reducibilities among combinatorial Problems*, in Complexity of Computer computation (R. Miller and J. Thatcher, Editors), Plenum, New York, 1972, 85—104.
6. Simovici D. — private communication.


*Received 18.XII.1978*

*Faculty of Mathematics*
*University of Iaşi*
*Iaşi, R.S. Romania*