RAIRO
INFORMATIQUE THÉORIQUE

HANS-JÜRGEN STENGER

Algebraic caractérisations of NTIME(F) and NTIME(F, A)

RAIRO – Informatique théorique, tome 18, nº 4 (1984), p. 365-385.
<http://www.numdam.org/item?id=ITA_1984__18_4_365_0>

R.A.I.R.O. Informatique théorique/Theoretical Informatics
(vol. 18, n° 4, 1984, p. 365 à 385)

ALGEBRAIC CHARACTERISATIONS OF NTIME(F) AND NTIME(F, A) (*)

by Hans-Jürgen STENGER (1)

Communicated by R. BOOK

Abstract. — *If F is a class of time bounds and A is a language, then NTIME(F) (NTIME(F, A)) is the class of languages accepted by nondeterministic Turing machines (by oracle machines with oracle set A) that are time bounded by a function of F. Each of these two classes is characterized algebraically through a uniform representation of its languages.*

As an application, several classes of formal languages, each with its relativized counterpart, are characterized by specification of F: the class of recursively enumerable (recursive, primitive recursive) sets, for each $k \geq 3$ the class E_k of sets whose characteristic function is in the Grzegorczyk class $\mathscr{E}_k$ and the class NP.

Résumé. — *Pour une classe F de bornes de temps et un langage A, NTIME (F) (NTIME (F, A)) est la classe qui se compose de tous les langages acceptés par des machines de Turing non déterministes (avec l'oracle A) dont le temps de calcul est borné par une fonction de F. Chacune des deux classes est caractérisée d'une manière algébraique au moyen d'une représentation uniforme de ses langages.*

Comme application de ces résultats, plusieurs classes de langages formels sont caractérisées en spécifiant F : la classe des ensembles récursivement énumérables (récursifs, primitifs-récursifs), pour chaque $k \geq 3$, la classe E_k des ensembles dont la fonction caractéristique est dans la classe $\mathscr{E}_k$ de Grzegorczyk et la classe NP, ainsi que pour chacune des classes mentionnées le pendant rélativisé.

INTRODUCTION

In recent years, many classes of formal languages have been algebraically characterized [2, 3, 5]. Generally, these characterisations were obtained by use of the regular languages or other classes of languages, e. g. the class of linear-context-free languages as a basis and allowing closure under elementary

(*) Received in August 1983, revised in January 1984.
This paper is an abridged version of the author's diploma thesis at the Faculty of Mathematics, University of Heidelberg.
(1) German Cancer Research Center, Department of Documentation, Information and Statistics, Section Epidemiology, Im Neuenheimer Feld 280, D-6900 Heidelberg.

operations. Using the fact that 'homomorphic replication', a generalisation of the concept of homomorphism, does not preserve the class of regular languages, many classes (e. g. NP, the class of recursively enumerable sets, etc.) could be characterized as the smallest class containing the regular languages and having certain closure properties.

Whereas in [2, 3, 5], each characterisation is proven as a single result, most of these characterisations are obtained in this paper as special cases of the characterisations of two families of classes of formal languages.

If F is a class of time bounds and A is a language, then NTIME(F) (NTIME(F, A)) is the class of languages accepted by nondeterministic Turing machines (oracle machines with oracle set A) that are time bounded by a function $f \in F$. Defining the 'F-erasing homomorphism', erasing-properties of homomorphisms can be related to the class of time bounding functions. In two theorems, each of these two families is characterized as the smallest class containing the regular languages (and one other language with information about the oracle set A, if NTIME(F, A) is concerned) that is closed under certain operations and F-erasing homomorphic duplication; homomorphic duplication is a simple form of homomorphic replication which does not use reversal.

By specification of the class F of time bounds, characterisations of several special classes are obtained in two corollaries. So many characterisations proven or stated in [2, 3, 5] are obtained as special cases of two general characterisations. Furthermore, two uniform representations for the languages of the two families of classes of languages are provided by the proofs of the theorems. So, besides the characterisations of NTIME(F) and NTIME(F, A) stated without proof in [4], we have two uniform representations for the languages of these classes.

In order to perform the characterisation of NTIME(F) in section 1, first a new representation for recursively enumerable (r.e.) languages is given which is based on regular languages and uses the operations length-preserving homomorphic duplication, homomorphism and intersection. After that, closure of NTIME(F, A) under F-erasing homomorphic replication is shown and then we are able to describe NTIME(F) in terms of F-erasing homomorphic duplication.

So as to get a basic representation for the languages of NTIME(F, A) in section 2, a modified version of the 'Representation Lemma' by R. V. Book and C. Wrathall in [5] serves as a starting-point for the characterisation of NTIME(F, A). At the end of each section, several well known classes of formal languages are described as an application of the general results: in section 1 the class RE of r.e. languages, the class REC of recursive languages, the class PRIMREC of primitive recursive languages, for $k \geqslant 3$ the class E_k of languages whose characteristic function is in the Grzegorczyk class $\mathscr{E}_k$ and the class NP of languages accepted in polynomial time by nondeterministic Turing machines, and in section 2 their relativized counterparts respectively.

LIST OF SYMBOLS

$x \in A$	x in A'
$A \subseteq B$	inclusion
$A \subset B$	proper inclusion: $A \subseteq B$ and not $A = B$
$A \cup B =$	$\{ x \mid x \in A$ or $x \in B \}$
$A \cap B =$	$\{ x \mid x \in A$ and $x \in B \}$
$A - B =$	$\{ x \mid x \in A$ and not $x \in B \}$
$A \times B =$	$\{ (x, y) \mid x \in A$ and $y \in B \}$
$A^n = A \times$	$\ldots \times A$ (n times)
$\mathbb{N}$	natural numbers 0, 1, 2, 3, ...
$\mathbb{N}^+ = \mathbb{N} - \{ 0 \}$	
$\varnothing$	empty set
$\mathbb{P}(A)$	power set of A

For functions f, g where $f : A \to B$ and $g : B \to D$,

DOMAIN$(f) = A$	
IMAGE(f)	$= fA = f(A) = \{ f(a) \mid a \in A \}$
$f' = f \mid E$	restriction of f to $E \subseteq A$
$h = g \circ f$	composition of g and f, with $h(a) = g(f(a))$ for all $a \in A$.

PRELIMINARIES

It is assumed that the reader is familar with the basic notions from theories of automata and formal languages. Only the concepts and notations that are most important for the understanding of this paper will be established in the following. For outstanding definitions, standard literature can be consulted [6, 7, 8].

If w is a word, then $|w|$ denotes the *length* of w. $|w|$ gives the number of symbols in w; for the empty word e, $|e| = 0$.

For any word w, let w^R be the *reversal* of w. The reversal of w is obtained from w by writing the symbols of w in reverse order. For the empty word e, $e^R = e$.

If A is a language and Σ the smallest (finite) alphabet such that $A \subseteq \Sigma^*$, then $\tilde{A} = \Sigma^* - A$ and $A \oplus \tilde{A} = \{c\} A \cup \{d\} \tilde{A}$ with two symbols c, d not in Σ.

A *homomorphism* is a function $h : \Sigma^* \to \Delta^*$ such that $h(uv) = h(u)h(v)$ for all $u, v \in \Sigma^*$.

A homomorphism $h : \Sigma^* \to \Delta^*$ is called *length-preserving* if $|h(w)| = |w L$ for all $w \in \Sigma^*$; *nonerasing* if $|h(w)| \geqslant |w|$ for all $w \in \Sigma^*$; *linear-erasing* on language $L \subseteq \Sigma^*$ if there exists a constant $k > 0$ such that for all $w \in L$ $k |h(w)| \geqslant |w|$ whenever $|w| \geqslant k$; *polynomial-erasing* on language $L \subseteq \Sigma^*$ if there exists a constant $k > 0$ such that for all $w \in L$ $|h(w)|^k \geqslant |w|$ whenever $|w| \geqslant k$.

The erasing properties of homomorphisms can be defined in a more general way, using classes of functions. If F is a class of functions $f : \mathbb{N} \to \mathbb{N}$, then a homomorphism h is called *F-erasing* on a language $L \subseteq \Sigma^*$ if there is a constant $k > 0$ and a function $f \in F$ such that for all $w \in L k f(|h(w)|) \geqslant |w|$ whenever $|w| \geqslant k$.

Let n be a positive integer, p a function $p : \{1, \ldots, n\} \to \{1, R\}$, L a language and $h_1, \ldots, h_n$ be n homomorphisms. Then the language

$$L' = \langle p; h_1, \ldots, h_n \rangle (L) = \{ (h_1(w))^{p(1)} \ldots (h_n(w))^{p(n)} \mid w \in L \}$$

is called a *homomorphic replication* on L.

If the function p has value 1 everywhere, then the homomorphic replication on L is called a *homomorphic duplication* on L and we write $\langle h_1, \ldots, h_n \rangle (L)$ instead of $\langle p; h_1, \ldots, h_n \rangle (L)$.

The concept of 'homomorphic replication' defined for languages can be extended to words if a word w is identified with its singleton $\{w\}$. Furthermore 'homomorphic duplication (replication)' also denotes the mapping that transforms the language L into L' or the word w into w'.

A class $\mathscr{L}$ of languages is closed under (length-preserving, non-, linear-, polynomial-, F-erasing) homomorphic replication if for every $n \in \mathbb{N}^+$, every function $p : \{1, \ldots, n\} \to \{1, R\}$, every language $L \in \mathscr{L}$ and every n (length-preserving, non-, linear-, polynomial-, F-erasing) homomorphisms $h_1, \ldots, h_n$ the language $L' = \langle p; h_1, \ldots, h_n \rangle (L)$ is in $\mathscr{L}$.

A *nondeterministic Turing machine* (TM) M is a quadruple $(Z, \Sigma, \delta, Z_{AC})$, where $Z = \{z_0, \ldots, z_s\}$ is a finite set of states, $\Sigma = \{a_0, \ldots, a_r\}$ the finite tape alphabet, z_0 the start state, $Z_{AC} \subset Z$ the set of accepting states and

$\delta : Z\times\Sigma \rightarrow \mathbb{P}(Z\times\Sigma\times\{-1, 0, +1\})$ the transition function. Conventionally $a_0\in\Sigma$ will be used for the empty tape square and will also sometimes be denoted by '$\langle\ \rangle$'.

M is called *deterministic* if for every state z and every tape symbol a the set $\delta(z, a)$ has at most one element.

A *configuration* C is a word in $\Sigma^*Z\Sigma^*$ and gives an instantaneous description of a step in M's computation. Let $C=wzav$ be a configuration. Then wav is the corresponding *tape inscription*; z denotes the current *state* of the finite control while M's read-write-head (RW-head) is scanning the symbol a. C is called a *halting configuration* whenever $\delta(z, a)=\emptyset$; if additionally $z\in Z_{AC}$, then C is called an *accepting configuration*; C is called a *start configuration* on input $x=a_1 \ldots a_n$, if $z=z_0$, $w=e$, $a=a_1$ and $v=a_2 \ldots a_n$.

Let $C=wzav$ and $C'=w'z'a'v'$ be configurations. Then C' is called a *successor configuration* of C and C a *predecessor configuration* of C', denoted by $C\vdash C'$, whenever one of the following statements holds:

(*a*) $(z', b, -1)\in\delta(z, a)$ and $w=w'a'$, $v'=bv$,
(*b*) $(z', a', 0)\in\delta(z, a)$ and $w=w'$, $v=v'$,
(*c*) $(z', b, +1)\in\delta(z, a)$ and $w'=wb$, $v=a'v'$ or $v=v'=e$, $a'=\langle\ \rangle$.

A *computation* of *length* k on input $x\in\Sigma^*$ is a sequence $C_0, \ldots, C_k$ of configurations, where C_0 is a start configuration on input x, and $C_j\vdash C_{j+1}$ for all j, $0\leqslant j<k$. A computation is called *halting* (*accepting*), if C_k is a halting (accepting) configuration.

The description given above allows the reader to imagine the TM as a 1-tape machine with the tape unbounded to the right. The machine is started by writing the input leftbound onto the tape. Then the RW-head is positioned over the first symbol of the input string and the finite state control is set into start state.

Since in every step of a Turing computation, only a finite part of the tape if filled with 'proper' symbols, i. e. tape symbols distinct from '$\langle\ \rangle$', a configuration C can be represented by a finite sequence of symbols of $Z\cup\Sigma$.

The definitions given above for 1-tape Turing machines can easily be done for multitape Turing machines, having k tapes, $k\geqslant 1$. Therefore, in the following, a Turing machine (TM) is to be understood as a nondeterministic multitape machine with k tapes, $k\geqslant 1$.

Let M be a TM with k tapes, $k\geqslant 1$, f, g and t functions from $\mathbb{N}$ to $\mathbb{N}$, and F a class of such functions.

Then $L(M)=\{x\in\Sigma^* \mid$ there is an accepting computation of M on input $x\}$ is the *language accepted* by M.

The function $T_M:\mathbb{N}\to\mathbb{N}$ defined by $T_M(n)=\max_{x\in L(M),|x|=n}\min\{k \mid k$ is the length of an accepting computation of M on input $x\}$ if such an x exists in $L(M)$ and $T_M(n)=0$, else, is called the *time-complexity* T_M of M. M is called $t(n)$-*time-bounded* and t a *time bound* for M if $T_M(n)\leqslant t(n)$ for $n\in\mathbb{N}$.

Let $O(f)=\{g:\mathbb{N}\to\mathbb{N} \mid$ there is a $c\in\mathbb{N}$ such that $g(n)\leqslant cf(n)$ for all $n\in\mathbb{N}\}$; then

$\text{NTIME}(f)=\{L(M) \mid M$ is a nondeterministic $O(f)$-time-bounded TM $\}$,
$\text{NTIME}(F)=\cup_{f\in F}\text{NTIME}(f)$.

We say 'the language L is accepted by a nondeterministic multitape Turing machine in F-time' if and only if L is in NTIME(F).

The function f *majorizes* the function g if $f(n)\geqslant g(n)$ for all $n\in\mathbb{N}$.

An *oracle machine* is a multitape Turing machine M with a distinguished work tape, the *query tape*, and the three distinguished states 'QUERY', 'YES', 'NO'. At some step of a computation on an input string w, M *may* transfer into state 'QUERY'. In state 'QUERY' M transfers into the state 'YES' if the string currently appearing on the query tape is in the *oracle set* A; otherwise, M transfers into state 'NO'; in either case, the query tape is instantly erased at the same step of the computation. Oracle machines can be deterministic or nondeterministic.

The *language accepted by M relative to the oracle set A* is $L(M, A)=\{x\in\Sigma^* \mid$ there is an accepting computation of M on input x when the oracle set is $A\}$

Time complexity for oracle machines can be defined in the same way as for Turing machines. The class of languages accepted in F-time by nondeterministic oracle machines with oracle set A will then be denoted by NTIME(F, A).

SECTION 1

First, lemma 1.1 gives a basic representation for recursively enumerable (r.e.) sets. Then, Remark 1.1 prepares Theorem 1, introducing F-erasing homomorphism. In Theorem 1, the representation for languages given by Lemma 1.1, leads to an algebraic characterisation of the class NTIME(F), with the intermediate result of a uniform representation for all languages of NTIME(F). Finally Corollary 1 characterizes several special classes of languages by specifying the class F of functions in Theorem 1.

LEMMA 1.1. *Every recursively enumerable language L can be represented as*

$$L=h\quad(h'\langle g_1, g_2\rangle(T)\cap h'\langle g_3, g_4\rangle(T))$$

with a regular language T, length-preserving homomorphisms g_1, g_2, g_3, g_4 and homomorphisms h and h'.

Proof. Let M be a 1-tape TM over the alphabet Σ_M such that $L\subseteq\Sigma_L{}^*\subseteq\Sigma_M{}^*$ is accepted by M. Let Σ be the alphabet $\Sigma_M\cup\{\$\}$ with a symbol '\$' not yet in Σ_M, Z be the set of states with the subset Z_{AC} of accepting states and the start state z_0. W.l.o.g. every accepting computation of M has even length. Every computation of M can be encoded in a word that consists of the successive configurations separated by a special sign. In the following construction, based on an idea of B. S. Baker and R. V. Book in [1], all accepting computations are represented in this way as an intersection of two sets. The language L is then obtained by deleting everything from these words except the input words contained in the start configurations.

An encoding scheme

A pair of configurations (C, C') such that $C\vdash C'$ can be coded by a word w in such a way that C and C', enriched by some '\$' symbols used to bring C and C' to equal length, can be regained from w by length-preserving homomorphisms, as follows:

including the tape squares under M's RW-head in two configurations C and C' such that $C\vdash C'$, no more than three '\$' symbols are needed to bring C and C' to equal length. Whereas C and C' are (possibly) of different length, D and D' are the corresponding *modified configurations*, representing finite sequences of equal length, which may differ only at three consecutive positions. Then the notation $D\vdash D'$ will be used for modified configurations if and only if $C\vdash C'$ was true.

Let $B:(\Sigma\cup Z)\times(\Sigma\cup Z)\to\Pi$ be a 1:1 function onto an alphabet Π with new symbols such that B codes the modified configurations $D=uabcv$ and $D'=ua'b'c'v$ with $D\vdash D'$ to the word $uB(a, a')B(b, b')B(c, c')v$. An empty tape square shall only be encoded by B as '\$' when it has no symbol of $\Sigma-\{\langle\,\rangle\}$ to its right; else the empty tape square shall be encoded as '$\langle\,\rangle$'.

Examples: For $\{a, b, c, \langle\,\rangle\}\subseteq\Sigma$, $\{u, v, w\}\subseteq\Sigma^*$, $\{z, z'\}\subseteq Z$, and configurations C and C' with $C\vdash C'$, let

(1) $C=uaz$ and $C'=uabz'$; then the modified configurations are $D=uaz\$\$$ and $D'=uabz'\$$, which are encoded by B in the word $uaB(z, b)B(\$, z')B(\$, \$)$.

(2) $C=uaz\langle\,\rangle v$ and $C'=uz'acv$; the modified configurations D, D' are

$D=C$ and $D'=C'$ which are encoded by B in the word $uB(a, z')B(z, a)B(\langle\,\rangle, c)v$.

The set of all such encoded pairs of modified configurations, each word concatenated with the new symbol '#' at its end is

$$S=\{\, uB(a, a')B(b, b')B(c, c')v\# \mid uabcv \vdash ua'b'c'v, \text{ with } u, v, a, a', b, b', c, c' \text{ in } (\Sigma \cup \Pi)^* \}.$$

S is a regular language:

Σ and Z are finite sets and each pair $(p, q) \in (\Sigma \cup Z)^2$ is mapped by B into the finite set Π. Encoding the Turing table, we get a proper subset of Π^3. So S becomes a proper subset of $\Sigma^*\Pi^3\Sigma^*\{\#\}$. Since every finite set is a regular language and the regular languages are closed under concatenation and Kleene*, S is a regular language. S remains regular if the choice of the symbols p, q is restricted and only a part of the Turing table is encoded by B.

The modified configurations D and D' such that $D \vdash D'$ can be obtained from S by the two length- preserving homomorphisms h_1 and h_2 that decode the symbols of Π and preserve all other symbols; thus

$$\{h_1(u)h_2(u) \mid u \in S\} = \{D\#D'\# \mid D \vdash D' \quad \text{and} \quad |D|=|D'|\}.$$

The words $D\#D'\#$ contain at most three '\$' symbols (added before coding) which can be deleted by a final homomorphism. Then one has the original word $C\#C'\#$ with the configurations C and C' corresponding D and D'.

The set of accepting computations of length $2n$ can now be represented as the intersection of the two sets

$$L_1=\{C_1\#C_3\#\ldots C_{2n+1}\#C_2\#C_4\#\ldots C_{2n}\# \mid C_{2j-1} \vdash C_{2j} \text{ for } 1 \leqslant j \leqslant n,\ n \in \mathbb{N}^+ \text{ with } C_1 \text{ a start- and } C_{2n+1} \text{ an accepting configuration}\}$$

and

$$L_2=\{C_1\#C_3\#\ldots C_{2n+1}\#C_2\#C_4\#\ldots C_{2n}\# \mid C_{2j} \vdash C_{2j+1} \text{ for } 1 \leqslant j \leqslant n,\ n \in \mathbb{N}^+ \text{ with } C_1 \text{ a start- and } C_{2n+1} \text{ an accepting configuration}\}.$$

Several codifications according to the encoding scheme are performed; so let (1) S' denote the set coding all start configurations together with a successor configuration; (2) A' denote the set coding all accepting configurations together with a precedessor configuration; (3) R' denote the set coding all pairs of consecutive configurations where neither of the two is a start nor an accepting configuration.

By translation with three length- preserving 1:1 onto homomorphisms, the common alphabet $\Sigma \cup \Pi$ of S', A' and R' is replaced by three new, pairwise disjoint alphabets Γ_S, Γ_A and Γ_R. The resulting new sets are S, A and R. Since S', A' and R' are regular and the regular sets are closed under homomorphisms, S, A and R are regular too. For each of these three sets, two length-preserving homomorphisms defined in the way described in the encoding scheme, provide the corresponding set of modified configurations. Thus, putting $\Gamma=\Sigma\cup Z\cup\{\#\}$, one has the following three pairs of length-preserving homomorphisms:

$$h_{1,S}:\Gamma_S{}^* \to \Gamma^* \quad\text{and}\quad h_{2,S}:\Gamma_S{}^* \to \Gamma^*,$$
$$h_{1,R}:\Gamma_R{}^* \to \Gamma^* \quad\text{and}\quad h_{2,R}:\Gamma_R{}^* \to \Gamma^*,$$
$$h_{1,A}:\Gamma_A{}^* \to \Gamma^* \quad\text{and}\quad h_{2,A}:\Gamma_A{}^* \to \Gamma^*,$$

and the three sets

$\{h_{1,S}(u)h_{2,S}(u)\mid u\in S\}=\{D_1\#D_2\#\mid D_1\vdash D_2$ where D_1 is a modified start configuration of $M\}$,

$\{h_{1,R}(v)h_{2,R}(v)\mid v\in R\}=\{D\#D'\#\mid D\vdash D'$ where neither D is a modified start configuration, nor D' a modified accepting configuration of $M\}$,

$\{h_{1,A}(w)h_{2,A}(w)\mid w\in A\}=\{D\#D_{AC}\#\mid D\vdash D_{AC}$ where D_{AC} is a modified accepting configuration of $M\}$.

In order to be able to distinguish the input symbols from all other symbols, $h_{1,S}$ is composed with a further length-preserving 1:1 onto homomorphism which translates IMAGE$(h_{1,S})$ into an alphabet Γ' consisting of new symbols except for '\$' in such a way that '\$' is mapped to '\$'. The modified length-preserving homomorphism is $h_{1,S}:\Gamma_S{}^*\to(\Gamma')^*$.

Construction of L_1

So as to delete all precedessor configurations of accepting configurations in L_1, the homomorphism $h_{1,A}$ is modified in the following way: after $h_{1,A}$ another length-preserving homomorphism is executed that translates IMAGE$(h_{1,A})$ into a new alphabet Δ. The modified length-preserving homomorphism is $h_{1,A}:\Gamma_A{}^*\to\Delta^*$.

Now L_1 can be represented as

$$L_1=h'\{h_{1,S}(u)h_{1,R}(v_1)\dots h_{1,R}(v_{n-1})h_{2,A}(w)h_{2,S}(u)h_{2,R}(v_1)\dots \dots h_{2,R}(v_{n-1})h_{1,A}(w)\mid \ u\in S, w\in A \ \text{and}\ v_j\in R \ \text{for}\ 1\leqslant j\leqslant n-1 \ \text{and}\ n\in\mathbb{N}\};$$

h' is the homomorphism that deletes all symbols of $\Delta\cup\{\$\}$, particularly the word $h_{1,A}(w)$, and preserves all other symbols; i. e. h is the homomorphism $h':(\Gamma'\cup\Gamma\cup\Delta)\to(\Gamma'\cup\Gamma-\{\$\})$ defined by $h'(a)=e$, for $a\in(\Delta\cup\{\$\})$, and $h'(a)=a$, else. First, the three length-preserving homomorphisms $h_{1,S}$, $h_{1,R}$ and $h_{2,A}$, defined on disjoint domains Γ_S, Γ_R and Γ_A can be combined into one length-preserving homomorphism g_1 defined on the union of these domains. Then the same is done for $h_{2,S}$, $h_{2,R}$ and $h_{1,A}$.

The resulting length-preserving homomorphisms are

$g_1 : (\Gamma_S \cup \Gamma_R \cup \Gamma_A)^* \to (\Gamma \cup \Gamma')^*$, defined by
$g_1 | \Gamma_S = h_{1,S}, \quad g_1 | \Gamma_R = h_{1,R}, \quad g_1 | \Gamma_A = h_{2,A},$

and

$g_2 : (\Gamma_S \cup \Gamma_R \cup \Gamma_A)^* \to (\Gamma \cup \Delta)^*$, defined by
$g_2 | \Gamma_S = h_{2,S}, \quad g_2 | \Gamma_R = h_{2,R}, \quad g_2 | \Gamma_A = h_{1,A}.$

Thus

$$
\begin{aligned}
L_1 &= h' \{ g_1(u) g_1(v_1 \ldots v_{n-1}) g_1(w) g_2(u) g_2(v_1 \ldots v_{n-1}) g_2(w) \mid u \in S, \ w \in A \\
&\qquad \text{and} \ v_j \in R \ \text{for} \ 1 \leqslant j \leqslant n-1 \ \text{and} \ n \in \mathbb{N}, n \geqslant 2 \} \\
&= h' \{ g_1(uv_1 \ldots v_{n-1} w) g_2(uv_1 \ldots v_{n-1} w) \mid \ u \in S, \ w \in A \ \text{and} \ v_j \in R \\
&\qquad \text{for} \ 1 \leqslant j \leqslant n-1 \ \text{and} \ n \in \mathbb{N}, n \geqslant 2 \} \\
&= h' \{ g_1(t) g_2(t) \mid t \in S(R+)A \} \\
&= h' \langle g_1, g_2 \rangle (T) \quad \text{for the regular language } T = S(R+)A.
\end{aligned}
$$

Construction of L_2

In order to delete all successor configurations of start configurations in L_2, the homomorphism $h_{2,S}$ is also modified: After $h_{2,S}$ another length-preserving homomorphism is executed that translates IMAGE($h_{2,S}$) into the same alphabet Δ as for L_1. The modified length-preserving homomorphism is $h_{2,S}$: $\Gamma_S^* \to \Delta^*$. Thus

$$
\begin{aligned}
L_2 = h' \{ & h_{1,S}(u) h_{2,R}(v_1) \ldots h_{2,R}(v_{n-1}) h_{2,A}(w) h_{2,S}(u) h_{1,R}(v_1) \ldots \\
& \ldots h_{1,R}(v_{n-1}) h_{1,A}(w) \mid \ u \in S, \ w \in A \quad \text{and} \ v_j \in R \\
& \text{for} \ 1 \leqslant j \leqslant n-1 \ \text{and} \ n \in \mathbb{N} \}
\end{aligned}
$$

where h' is the homomorphism defined above; in particular h' deletes the word $h_{2,S}(u)$. Combining the three homomorphisms defined on disjoint domains in the same way as for L_1, we obtain the following two length-preserving homomorphisms:

$g_3 : (\Gamma_S \cup \Gamma_R \cup \Gamma_A)^* \to (\Gamma \cup \Gamma')^*$, defined by
$g_3 | \Gamma_S = h_{1,S}, \quad g_3 | \Gamma_R = h_{2,R}, \quad g_3 | \Gamma_A = h_{2,A},$

and

$g_4 : (\Gamma_S \cup \Gamma_R \cup \Gamma_A)^* \to (\Gamma \cup \Delta)^*$, defined by
$g_4 | \Gamma_S = h_{2,S}, \quad g_4 | \Gamma_R = h_{1,R}, \quad g_4 | \Gamma_A = h_{1,A}.$

Thus

$$L_2 = h'\{g_3(u)g_3(v_1 \ldots v_{n-1})g_3(w)g_4(u)g_4(v_1 \ldots v_{n-1})g_4(w) \mid u \in S,\ w \in A \text{ and } v_j \in R \text{ for } 1 \leqslant j \leqslant n-1 \text{ and } n \in \mathbb{N}, n \geqslant 2\}$$
$$= h'\{g_3(uv_1 \ldots v_{n-1}w)\ g_4(uv_1 \ldots v_{n-1}w) \mid u \in S,\ w \in A \text{ and } v_j \in R \text{ for } 1 \leqslant j \leqslant n-1 \text{ and } n \in \mathbb{N}, n \geqslant 2\}$$
$$= h'\{g_3(t)g_4(t) \mid t \in S(R+)A\}$$
$$= h'\langle g_3, g_4\rangle(T) \text{ for the regular language } T = S(R+)A.$$

Now the situation is the following:

A word $x = C_1\# \ldots C_{2n+1}\#C_2\# \ldots C_{2n}\#$ represents an accepting computation of length $2n$ on input y contained in C_1 if and only if x is an element of $L_1 \cap L_2$. In order to obtain the input y of the accepting computation represented by x, a homomorphism h is needed that deletes all symbols not in Γ', translates the remaining symbols into Γ and finally deletes the symbols z_0 and $\#$.

Thus $L = h(L_1 \cap L_2)$, where $L_1 = h'\langle g_1, g_2\rangle(T)$, $L_2 = h'\langle g_3, g_4\rangle(T)$, with a regular language T, length-preserving homomorphisms g_1, g_2, g_3, g_4 and the homomorphisms h and h'. □

CLAIM: *The homomorphism h' is linear-erasing on $\langle g_1, g_2\rangle(T)$ and $\langle g_3, g_4\rangle(T)$.*

Proof. Let $x \in \langle g_1, g_2\rangle(T)$. Then x represents $2m$ configurations, $m \geqslant 2$. h' erases one configuration either in $\langle g_1, g_2\rangle(T)$ and $\langle g_3, g_4\rangle(T)$.

Thus $2m-1 \leqslant |h'(x)|$. x consists of m pairs of modified configurations, i. e. x contains at most $3m$ '\$' symbols. For the configuration D deleted by h', $|D| \leqslant |h'(x)|$ holds. Thus $|x| \leqslant |h'(x)| + 3m + |h'(x)|$, and hence $|x| \leqslant 6|h'(x)|$. By symmetry, the claim also holds for $x \in \langle g_3, g_4\rangle(T)$. □

REMARK 1.1. *Let F be a class of functions from $\mathbb{N}$ to $\mathbb{N}$ such that*

— *F is closed under composition,* (P1)

— *$f(n) \geqslant n$ for all $n \in \mathbb{N}$ and $f \in F$,* (P2)

— *there is a function $g' \in F$ such that $g'(n) \geqslant n^2$ for $n \in \mathbb{N}$.* (P3)

Then the following statement holds:

If $L \in$ NTIME(F), then in Lemma 1.1 the homomorphism h is F-erasing on language $L_1 \cap L_2$, and the homomorphism h' is F-erasing on the languages $\langle g_1, g_2\rangle(T)$ and $\langle g_3, g_4\rangle(T)$.

Proof. h deletes $2n$ configurations, each with at most $|C_1| + 2n$ symbols, $(2n+1)$ '#' symbols and z_0 from C_1, where $|C_1| = |y| + 1$. If $h(x) = y$, then

$$|x| \leqslant |C_1| + (|C_1| + 2n)2n + (2n+1) \leqslant (2n)^2 + 4n + |y|(2n+1) + 2.$$

If $L \in \mathrm{NTIME}(F)$, then $2n \leqslant cf(|y|)$ for a function $f \in F$ and a constant $c \in \mathbb{N}^+$. So

$$|x| \leqslant c^2 f(|y|)^2 + 2cf(|y|) + |y|(cf(|y|)+1)+2.$$

Using P1, P2 and P3, this yields $|x| \leqslant 7c^2 g(|y|)$ for $|y| \geqslant 1$. (*)

Since $|x| \geqslant 1$, (*) holds for $|x| \geqslant 7c^2$; so h is F-erasing on language $L_1 \cap L_2$. By P2, the claim implies that h' is F-erasing on the languages $\langle g_1, g_2 \rangle (T)$ and $\langle g_3, g_4 \rangle (T)$. □

LEMMA 1.2. *Let F be a class of functions from* $\mathbb{N}$ *to* $\mathbb{N}$ *such that*

— *F is closed under composition,* (P1)

— $f(n) \geqslant n$ *for all* $n \in \mathbb{N}$ *and* $f \in F$, (P2)

— $f(n+1) \geqslant f(n)$ *for all* $n \in \mathbb{N}$ *and* $f \in F$, (P4)

— *there is a function* $g' \in F$ *such that* $g'(n) \geqslant n^2$ *for* $n \in \mathbb{N}$. (P3)

Then the following statement holds:

The classes NTIME(F) *and* NTIME(F, A) *are closed under F-erasing homomorphic replication.*

Proof. Let $L_1 \in \mathrm{NTIME}(F, A)$, $m \geqslant 1$, p be a function from $\{1, \ldots, m\}$ to $\{1, R\}$, $h_1, \ldots, h_m$ be m homomorphisms from Σ^* to Δ^* which are F-erasing on L_1.

Let $L_2 = \langle p; h_1, \ldots, h_m \rangle (L_1)$. We have to show that $L_2 \in \mathrm{NTIME}(F, A)$. For $L_1 \in \mathrm{NTIME}(F, A)$ there exists an r-tape oracle machine M_1 with oracle set A which accepts L_1 in F-time. We construct a nondeterministic oracle machine M_2 with oracle set A which accepts L_2 in F-time.

Definition of M_2

Let M_2 have a finite state control, a finite tape alphabet and $3+r$ tapes performing the following tasks:

tape 1: M_2's input tape; takes $y \in \Delta^*$ as input;

tape 2: constructs the homomorphic replication of $w \in \Sigma^*$;

tape 3: stores $w \in \Sigma^*$, nondeterministically generated by M_2;

tape 4, ..., tape $3+r$: are simulating the *oracle machine* M_1; the input tape of M_1 is simulated by tape 4.

In the following, an accepting computation of M_2 on input y is defined by four consecutive phases.

Phase 1: M_2 guesses nondeterministically a word $w \in \Sigma^*$ by specifying at each step a symbol of Σ and writing it simultaneously onto tape 3 and 4. Finally the RW-heads of tapes 3 and 4 return to their start position over the first symbol of w. $2|w|$ steps are needed.

Phase 2: M_2 simulates the computation of M_1 on input w, i. e. M_2 checks if $w \in L_1$ using tapes $4, \ldots, 3+r$. If the computation of M_1 (simulated by M_2) is accepting, phase 3 is started. Since L_1 is in NTIME(F, A), '$w \in L_1$' can be verified in F-time by M_2 too. For an accepting computation of M_2 no more than $cf(|w|)$ steps are needed with a constant $c \geqslant 1$ and a function $f \in F$.

Phase 3: M_2 evaluates the m homomorphisms $h_1, \ldots, h_m$ for w and writes the values, after p is executed, one behind the other, onto tape 2. To do this, M_2 moves the RW-head of tape 3 m-times over w and back to the start position. At the j-th such pass, M_2 writes $h_j(w)$ onto tape 2 if $p(j)=1$; if $p(j)=R$, M_2 writes $h_j'(w^R)$ on the same tape, where h_j' is the homomorphism from Σ^* to Δ^*, defined by $h_j'(a)=h_j(a)^R$ for $a \in \Sigma$. So $h_j'(w^R)$ becomes $h_j(w)^R$. At the end of this phase $\langle p; h_1, \ldots, h_m \rangle (w)$ is written onto tape 2. Any homomorphism h can be evaluated by a suitably programmed oracle machine in linear time. So for m homomorphisms $h_1, \ldots, h_m$ *one* constant $k>0$ can be found, such that each of the m images of w is computed and written onto tape in no more than $k|w|$ steps. The steps needed for this phase do not exceed

$$\max(2m|w|, \quad mk|w| + |\langle p; h_1, \ldots, h_m \rangle (w)|)$$

and, since w.l.o.g. $k \geqslant 2$, $mk|w| + \langle p; h_1, \ldots, h_m \rangle (w)|$.

Phase 4: Beginning with the first tape square, M_2 reads tapes 1 and 2 simultaneously and checks whether the input y and the word $\langle p; h_1, \ldots, h_m \rangle (w)$ are equal. If equality holds for all symbols, M_2 transfers into an accepting state. During this phase, $|y|$ steps are needed. In order to compute the length of an accepting computation, we have to find a bound for $|w|$ in terms of $|y|$. Each h_j, $1 \leqslant j \leqslant m$, is F-erasing on L_1, i. e. there are $f_j \in F$ and $k_j \in \mathbb{N}$ such that for each j, $1 \leqslant j \leqslant m$, $|w| \leqslant k_j f_j(|h_j(w)|)$. Putting $k' = \max_{1 \leqslant j \leqslant m}\{k_j\}$, we have $|w| \leqslant k' f_j(|h_j(w)|)$ for every j, $1 \leqslant j \leqslant m$. For j, $1 \leqslant j \leqslant m$, each $f_j(|h_j(w)|])$ can be replaced by $f_j(|y|)$ using P4. Starting with the m functions and using P1, P2 and P4, a function of F can be defined by $f' = f_1 \circ \ldots \circ f_m$ which majorizes each f_j for j, $1 \leqslant j \leqslant m$. Using P3 and P1, another function $g \in F'$ exists such that $|w| \leqslant g(|y|)$. (*)

Adding the steps of the four phases, we obtain

$$2|w| + cf(|w|) + mk|w| + |y| + |y|;$$

using (*) and F's properties, the total amount of steps does not exceed $f''(|y|)$ for a function $f'' \in F$. That is, M_2 accepts L, in F-time.

The closure property of NTIME(F) can also be shown directly if at the beginning of the proof the tapes $4,\ldots,3+r$ simulate a r-tape Turing machine. □

THEOREM 1. *Let F be a class of functions from $\mathbb{N}$ to $\mathbb{N}$ such that*

— *F is closed under composition,*

— $f(n)\geq n$ *for all* $n\in\mathbb{N}$ *and* $f\in F$,

— $f(n+1)\geq f(n)$ *for all* $n\in\mathbb{N}$ *and* $f\in F$,

— *there is a function* $g'\in F$ *such that* $g'(n)\geq n^2$ *for* $n\in\mathbb{N}$. (*)

Then the following statements hold:

(i) NTIME(F) *is the smallest class of languages containing the regular languages that is closed under*

— *intersection,*

— *length-preserving homomorphic replication [duplication],*

— *F-erasing homomorphism.*

(ii) NTIME(F) *is the smallest class of languages containing the regular languages that is closed under*

— *intersection,*

— *F-erasing homomorphic replication [duplication].*

(iii) NTIME(F) *is closed under union, inverse homomorphism, concatenation and Kleene*.*

Proof. With the assumptions made for F, closure under F-erasing homomorphism and F-erasing homomorphic duplication follows immediately from Lemma 1.2. It is easy to see that NTIME(F) has the closure properties stated in (iii). Now let $L\in$NTIME(F) and $\mathscr{L}(F)$ be a class of languages having the properties claimed for NTIME(F) in (i). We show that $L\in\mathscr{L}(F)$. L is in NTIME(F) if and only if there exists a nondeterministic multitape TM which accepts L in F-time. Since F is closed under composition and has property (*), L can be accepted by a nondeterministic 1-tape TM in F-time too. By property (*) NTIME(F) contains the regular languages. Since L is a r. e. language, Lemma 1.1 yields a representation of the form

$$L=h\quad(h'\langle g_1,g_2\rangle(T)\cap h'\langle g_3,g_4\rangle(T)),$$

with a regular language T, length- preserving homomorphisms g_i and homomorphisms h and h'. By choice of F, h and h' become F-erasing on corresponding languages applying Remark 1.1. With the properties assumed for $\mathscr{L}(F)$, $L\in\mathscr{L}(F)$ and part one of the theorem is shown. Minimality for part two follows as in part one, since closure under F-erasing homomorphic replication implies closure under length-preserving homomorphic duplication and F-erasing homomorphism. □

The proof of Theorem 1 also gives a uniform representation for the languages of NTIME(F) whatever class of functions F (satisfying the conditions) is used. So every language $L \in$ NTIME(F) can be represented as the image under an F-erasing homomorphism h of the intersection of two languages L_1, L_2 where each L_i can be constructed out of the *same* regular set T by means of *two* length-preserving homomorphic duplications and *one* linear-erasing homomorphism h'.

Composing h' with each g_i, $1 \leqslant i \leqslant 4$, a cruder representation for L is obtained; thus L can be represented as the F-erasing homomorphic image of the intersection of *two* linear-erasing homomorphic duplications on the *same* regular set.

Specifying the class F of functions, several well known classes of formal languages now can be characterised in an analogous way. We now list the most important examples.

Let RE denote the class of recursively enumerable languages, REC denote the class of recursive languages, PRIMREC denote the class of primitive recursive languages, E_k denote the class of languages whose characteristic function is in the Grzegorczyk class $\mathscr{E}_k$, $k \geqslant 0$, and NP denote the class of languages accepted by nondeterministic Turing machines in polynomial time.

Then the following characterisations can be established:

COROLLARY 1. (i) *RE (REC, PRIMREC, NP, E_k, where $k \geqslant 3$) is the smallest class of languages containing the regular languages that is closed under*

— *intersection,*

— *length-preserving homomorphic replication [duplication],*

— *(recursive-, primitive-recursive-, polynomial-, $\mathscr{E}_k$-erasing) homomorphic replication [duplication].*

(ii) *RE (REC, PRIMREC, NP, E_k, where $k \geqslant 3$) is the smallest class of languages containing the regular languages that is closed under*

— *intersection,*

— *(recursive-, primitive-recursive-, polynomial-, $\mathscr{E}_k$-erasing) homomorphic replication [duplication].*

(iii) *RE (REC, PRIMREC, NP, E_k,* where $k \geqslant 3$) *is closed under union, inverse homomorphism, concatenation and Kleene*.*

Proof. (NP): Let $F = \{ \lambda n . n^k \mid k \geqslant 2 \}$, then NTIME($F$) = NP.

Now let $F_0 = \{ f: \mathbb{N} \to \mathbb{N} \mid f(n+1) \geqslant f(n)$ and $f(n) \geqslant n$ for $n \in \mathbb{N} \}$, and note that F_0 contains the function $\lambda n . n^2$.

(RE): Let $F = \{ f \mid f: \mathbb{N} \to \mathbb{N} \}$; F contains $\lambda n . n^2$. Every function $f \in F$

can be majorized by a function $g \in F \cap F_0$ and it is clear that NTIME(F)=RE, so NTIME$(F) \subseteq$ RE $\subseteq$ NTIME$(F \cap F_0) \subseteq$ NTIME(F), which implies that NTIME$(F \cap F_0)$=NTIME(F)=RE.

(REC): Let $F = \{ f : \mathbb{N} \to \mathbb{N} \mid f$ recursive $\}$; F contains $\lambda n \,.\, n^2$. Again, $f \in F$ can be majorized by a function $g \in F \cap F_0$ and NTIME(F)=REC, so NTIME$(F) \subseteq$ REC $\subseteq$ NTIME$(F \cap F_0) \subseteq$ NTIME(F). This implies NTIME$(F \cap F_0)$ = NTIME(F) = REC.

(E_k): Let $\mathscr{F}_k = \mathscr{E}_k \cap F_0$ for $k \geqslant 3$; so each $\mathscr{F}_k$, $k \geqslant 3$, contains the function $\lambda n \,.\, n^2$. Using properties of the Grzegorczyk classes $\mathscr{E}_k$ which are proven in [9], for each function $f \in \mathscr{E}_k$, $k \geqslant 3$, a function $g \in \mathscr{E}_k$ can be defined that majorizes f and satisfies the condition of F_0. So, for $k \geqslant 3$, NTIME$(\mathscr{E}_k) \subseteq$ NTIME$(\mathscr{F}_k)$.

In [9] a function $f \in \mathscr{E}_k$ is related to a 'step counting function' s_f which counts the number of steps needed to compute f on a register machine (*see* [8] or [9] for explicit definitions). From results in [9], the following statement follows immediately:

(*a*) For $k \geqslant 3$, a function f is in $\mathscr{E}_k$ if and only if f is computed by a register machine with time bound t in $\mathscr{E}_k$.

Using simulations of register machines by Turing machines and vice versa, carried out in [8], and properties of the step counting function from [9], the two following statements can be derived:

(*b*) Every register machine with time bound $\lambda n \,.\, t(n)$ can be simulated by an $O(t^3(n))$-time-bounded 3-tape Turing machine.

(*c*) Every $t(n)$-*time-bounded* multitape Turing machine can be simulated by a register machine with time bound $\lambda n \,.\, ct(n)$, $c \in \mathbb{N}$.

Combining these results with elementary properties of the Grzegorczyk classes given in [9], NTIME$(\mathscr{E}_k) = E_k$. So for $k \geqslant 3$,

$$E_k = \text{NTIME}(\mathscr{E}_k) \subseteq \text{NTIME}(\mathscr{F}_k) \subseteq \text{NTIME}(\mathscr{E}_k) = E_k,$$

which implies NTIME$(\mathscr{F}_k)$=NTIME$(\mathscr{E}_k) = E_k$ for $k \geqslant 3$.

(PRIMREC): The preceeding part implies this one as follows: $\cup_{k \geqslant 0} E_k$ = PRIMREC, $\cup_{k \geqslant 0} \mathscr{E}_k = \cup_{m \geqslant 1} \{ f : \mathbb{N}^m \to \mathbb{N} \mid f$ primitive recursive, $m \in \mathbb{N}^+ \}$.

Let $F = \{ f : \mathbb{N} \to \mathbb{N} \mid f$ primitive recursive $\}$; then F contains the function $\lambda n \,.\, n^2$ So $f \in F$ if and only if there is a $k \geqslant 0$ such that f is a unary function in $\mathscr{E}_k$. □

SECTION 2

Starting with the 'Representation Lemma' presented by R. V. Book and C. Wrathall in [5], a (uniform) representation for the languages of NTIME(F, A) is established in Lemma 2.1. Theorem 2 characterizes the class NTIME(F, A) algebraically. Finally, the same special classes of languages as in section 1 but relativized to the oracle set A, are characterized in Corollary 2. We recall the Representation Lemma in a modified notation:

Representation Lemma. Let M be an oracle machine that runs in time $t(n)$ and has tape alphabet Δ. There exist homomorphisms h and g and a language L_M such that

(i) for any oracle set $A \subseteq \Delta^*$, $L(M, A) = h(L_M \cap g^{-1}((A \oplus \tilde{A})^*))$,
(ii) L_M is accepted in linear time by a deterministic multitape Turing machine,
(iii) for all $w \in L_M$, $|w| \leqslant t(|h(w)|)$.

The proof of the Representation Lemma follows that of Theorem 2.3.1 [10]. We include a sketch:

Roughly speaking, L_M contains (encodings of) all triples (x, y, z) such that y is an accepting computation of M on the input string x with precisely information z about the oracle set. It is possible to construct a deterministic multitape TM which, on input (x, y, z), checks in linear time whether (x, y, z) has the above properties. The homomorphism g satisfies $g((x, y, z)) = z$ so that strings in $L_M \cap g^{-1}((A \oplus \tilde{A})^*)$ describe accepting computations of M with oracle set A; the homomorphism h satisfies $h((x, y, z)) = x$ so that the input string accepted by M is returned. Since M operates in time $t(n)$, the length of the encoding (x, y, z) can be made proportional to $t(|x|)$ so that for all $w \in L_M$, $|w| \leqslant t(|h(w)|)$.

Modified for our purposes, the Representation Lemma reads:

LEMMA 2.1. *Let F be a class of functions from $\mathbb{N}$ to $\mathbb{N}$; then the following statement holds:*

If $L \in$ NTIME(F, A), then L can be represented as

$$L = h(L_M \cap g^{-1}((A \oplus \tilde{A})^*)),$$

with homomorphisms h, g, and a language L_M accepted by a deterministic multitape Turing machine in linear time, and h F-erasing on language $L_M \cap g^{-1}((A \oplus \tilde{A})^*)$.

THEOREM 2. *Let F be a class of functions from* $\mathbb{N}$ *to* $\mathbb{N}$ *such that*

— *F is closed under composition,*

— $f(n) \geqslant n$ *for all* $n \in \mathbb{N}$ *and* $f \in F$,

— $f(n+1) \geqslant f(n)$ *for all* $n \in \mathbb{N}$ *and* $f \in F$,

— *there is a function* $g' \in F$ *such that* $g'(n) \geqslant n^2$ *for* $n \in \mathbb{N}$. (*)

Then the following statements hold:

(i) NTIME(F, A) *is the smallest class of languages containing the regular languages and the language* $(A \oplus \tilde{A})^*$ *that is closed under*

— *intersection,*

— *inverse homomorphism*

— *F-erasing homomorphic replication [duplication].*

(ii) NTIME(F, A) *is closed under union, concatenation and Kleene*.*

Proof. By Lemma 1.2, NTIME(F, A) is closed under F-erasing homomorphic replication and it is easy to see that it is also closed under the remaining operations. By (*), NTIME(F, A) contains the regular languages. Using F's properties and the special ability of an oracle machine, an oracle machine with oracle set A can be constructed which accepts the language $(A \oplus \tilde{A})^*$ in F-time.

Now let $\mathscr{L}(F, A)$ be a class of languages with the properties claimed for NTIME(F, A). We want to show that NTIME$(F, A) \subseteq \mathscr{L}(F, A)$. So let $L \in$ NTIME(F, A). Using Lemma 2.1, L can be represented as

$$L = h(L_M \cap g^{-1}((A \oplus \tilde{A})^*)),$$

with homomorphisms h, g, and a language L_M accepted by a deterministic multitape Turing machine in linear time, and h F-erasing on $L_M \cap g^{-1}((A \oplus \tilde{A})^*)$.

Assuming the same properties for F, by Theorem 1 (ii) NTIME(F) is the smallest class of languages containing the regular languages that is closed under intersection and F-erasing homomorphic replication (duplication). Thus NTIME$(F) \subseteq \mathscr{L}(F, A)$. By (*), NTIME$(F)$ also contains the language L_M which implies $L_M \in \mathscr{L}(F, A)$. Using L's representation together with the remaining properties assumed for $\mathscr{L}(F, A)$ yields $L \in \mathscr{L}(F, A)$. □

As in section 1, a uniform characterisation for several classes of languages can be established.

Let RE(A) denote the class of recursively enumerable languages, REC(A) denote the class of recursive languages, PRIMREC(A) denote the class of primitive recursive languages, $E_k(A)$ denote the class of languages whose characteristic function is in the Grzegorczyk class $\mathscr{E}_k$, $k \geqslant 0$, and NP(A) denote the class of languages accepted by nondeterministic Turing machines in polynomial time, each of these classes relativized to oracle set A. Then the following characterisations can be derived:

COROLLARY 2. (i) RE(A) (REC(A), PRIMREC(A), NP(A), $E_k(A)$, *where* $k \geq 3$) *is the smallest class of languages containing the regular languages and the language* $(A \oplus \tilde{A})^*$ *that is closed under*

— *intersection,*
— *inverse homomorphism,*
— (*recursive-, primitive-recursive-, polynomial-,* $\mathscr{E}_k$*-erasing*) *homomorphic replication* [*duplication*].

(ii) RE(A), (REC(A), PRIMREC(A), NP(A), $E_k(A)$, *where* $k \geq 3$) *is closed under union, concatenation and Kleene*.*

Proof. Analogous to the proof of Corollary 1 using oracle machines with oracle set A instead of Turing machines. □

DISCUSSION AND CONCLUDING REMARKS

The class of functions

In this paper, F is a class of time bounds for Turing or oracle machines. So, in both theorems, the first two conditions on F, 'F is closed under composition' and '$f(n) \geq n$ for $n \in \mathbb{N}$ and $f \in F$' are quite natural. The third condition, 'all functions are weakly monotonically increasing', was chosen to accomplish the proof of Lemma 1.2. We had to find *one* function g in F that majorizes *each* of m other functions in F. Together with the first two conditions, g is defined by composition of the m functions to be majorized. The last condition, 'F contains a function that majorizes the function $\lambda n . n^2$', was chosen for the calculation of running time and also compensates the loss of time while simulating a multitape TM by a 1-tape TM.

In another paper that gives a survey of computational complexity [4], R. V. Book states without proof the same characterisations of NTIME(F) and NTIME(F, A) as in the two theorems. Instead of requiring that each $f \in F$ is weakly monotonically increasing, he demands '$f(m)+f(n) \leq f(m+n)$ for every $f \in F$ and $n \in \mathbb{N}$'. It is easy to see that this implies weak monotony for each $f \in F$.

The results

Whereas in [1], B. S. Baker and R. V. Book prove a representation for r. e. languages based on linear context-free languages using intersection and homomorphism, Lemma 1.1 gives a representation for these languages based on more simple (regular) languages but needs, besides intersection and homomorphism, a more complicated operation (length-preserving homomorphic duplication). In the characterisation of the class RE of r. e. languages in [2], the proofs need homomorphic replication, whereas in the present paper homomorphic duplication is sufficient.

The characterisations of the classes NP, RE and NP(A) in [2, 5] use the class of linear context-free languages and $\mathscr{L}_{BNP}$ as auxiliary classes. In this paper they are obtained as special cases of general and uniform characterisations of NTIME(F) resp. NTIME(F, A) by specification of the class F of time bounds, with the regular languages as a basis and without use of further classes. Likewise, other characterisations stated in [5] without proof follow by specification of F.

In Theorems 1 and 2, the characterisations of NTIME(F) and NTIME(F, A) are obtained through representations of their languages. So, extending the results of [4], the theorems also provide uniform representations for the languages of the two (families of) classes, and reveal the common structure of many apparently different classes.

Comparing the characterisations of NTIME(F) and NTIME(F, A) and the representations for their languages, note that the (more complicated) relativized counterpart is received from the unrelativized one by adding a language which contains information about the oracle set and requiring closure under inverse homomorphism as a further operation. Furthermore, the representations for the languages show a strong connection between complexity and the erasing-properties of the homomorphisms needed for their construction.

ACKNOWLEDGEMENTS

I am indebted to my advisor, Dr Diana Schmidt (University of Karlsruhe) for the suggestions which led to this work, for her helpful criticism and the continued good communication and to Professor Werner Böge for his support which has been essential for the realisation of the diploma thesis. Finally, I wish to thank Professor Ronald V. Book for his 'three words' of encouragement.

REFERENCES

1. B. S. Baker and R. V. Book, *Reversal-bounded Multipushdown Machines*, J. of Computer and Systems Science, Vol. 8, 1974, pp. 315-332.
2. R. V. Book, *Simple Representations of certain classes of Languages*, J.A.C.M., Vol. 25, 1978, No. 1, pp. 23-31.

3. R. V. BOOK, *On languages Accepted by Space-bounded Oracle-Machines*, Acta Informatica, Vol. 12, 1979, pp. 177-185.
4. R. V. BOOK, *Complexity Classes of Formal Languages*, Springer Lect. Notes in Comp. Sci., Vol. 74, pp. 43-56, Springer, Berlin, New York, 1979.
5. R. V. BOOK and C. WRATHALL, *On languages specified by relative acceptance*, Theoretical Computer Science, Vol. 7, 1978, pp. 185-195.
6. S. GINSBURG, *Algebraic and automata-theoretic properties of formal languages*, North-Holland, Amsterdam, 1975.
7. J. HOPCROFT and J. ULLMAN, *Introduction to Automata Theory, Languages and Computation*, Addison-Wesley, Amsterdam, 1979.
8. W. J. PAUL, *Komplexitätstheorie*, Teubner, Stuttgart, 1978.
9. H. SCHWICHTENBERG, *Rekursionszahlen und die Grzegorczyk Hierarchie*, Arch. math. Logik, Vol. 12, 1969, pp. 85-97.
10. C. WRATHALL, *Subrecursive Predicates and Automata*, Ph. D. dissertation, Harvard University, 1975.