Theoretical Computer Science 3 (1977) 61–74.

A RECURSIVE AND A GRAMMATICAL CHARACTERIZATION OF THE EXPONENTIAL-TIME LANGUAGES

Burkhard MONIEN

Abteilung Informatik, Universität Dortmund, Dortmund, Federal Republic of Germany

Communicated by A. Salomaa
Received 15 July 1975
Revised 15 January 1976

We characterize the class of all languages which are acceptable in exponential time by means of recursive and grammatical methods. (i) The class of all languages which are acceptable in exponential time is uniquely characterized by the class of all (0–1)-functions which can be generated, starting with the initial functions of the Grzegorczyk-class $\mathscr{E}^2$, by means of substitution and limited recursion of the form $f(x, y+1) = h(x, y, f(x, y), f(x, l(x, y))), l(x, y) \leq y$. (ii) The class of all languages which are acceptable in exponential time is equal to the class of all languages generated by context-sensitive grammars with context-free control sets.

1. Introduction

In this paper we will characterize the class E of all languages which are acceptable by deterministic multi-tape Turing machines in exponential time. This class is very interesting in connection with the relationship between time and tape complexity. It is not known yet whether E is different from the class of all languages acceptable by deterministic linear bounded automata. E is formally defined in the following way:

$$E = \{L \mid \exists d \in \mathbf{N} \wedge \exists \text{ deterministic multi-tape Turing machine } M \text{ accepting } L \text{ within the time bound } d^n\}.$$

This definition is largely independent from the Turing machine being the underlying machine model.

A characterization of the class E by means of some machine model was given by Cook in [1]. He proved that a language belongs to E if and only if it is accepted by a writing pushdown acceptor [5]. A writing pushdown acceptor consists of a finite memory, a pushdown tape and an input tape with a head that can print and move in both directions on the tape segment given by the input string.

In this paper we will give two further characterizations of the class E, a recursive and a grammatical one. These characterizations are similar to those known for deterministic context-sensitive languages.

Ritchie proved in [9] that the deterministic context-sensitive languages are characterized by the (0–1)-functions of the class $\mathscr{E}^2$ of the Grzegorczyk hierarchy [2]. We get a similar description for E replacing the scheme for the limited primitive recursion

$$[g, h, j \in \mathscr{E}^2; f(x, y) = \lambda xy\,[y = 0 \to g(x), h(x, y, f(x, y-1))], \\ f(x, y) \leqslant j(x, y)] \Rightarrow f \in \mathscr{E}^2$$

in the definition of $\mathscr{E}^2$ by the scheme

$$[g, h, j, l \in \mathscr{F};\ f(x, y) = \lambda xy\,[y = 0 \to g(x), h(x, y, f(x, y-1), f(x, l(x, y))], \\ l(x, y) < y, f(x, y) \leqslant j(x, y)] \Rightarrow f \in \mathscr{F}.$$

Then E is characterized by the (0–1)-functions belonging to the class $\mathscr{F}$.

On the other hand E is equal to the class of all languages which are generated by context-sensitive grammars with context-free control sets (with or without "appearance checking"). When we denote these classes by $\mathscr{L}(1,2,0)$ and $\mathscr{L}(1,2,1)$, respectively (notation from [10]), then we get $\mathscr{L}(1,2,0) = \mathscr{L}(1,2,1) = E$. Therefore the question asked in [10] whether $\mathscr{L}(1,2,0)$ is equal to the class of the context-sensitive languages is a question about the relationships which hold between deterministic time bounds and nondeterministic tape bounds.

Proving the recursive characterization of E we have to define an arithmetization of the computation of a Turing machine. We do this without encoding whole tape inscriptions by natural numbers. Thus we avoid defining a great number of functions and predicates describing the moves of a Turing machine. This method leads to a proof of Kleene's normal form theorem which is considerably shorter than those given in the literature.

In Section 2 we will prove the recursive characterization of E and in Section 3 the grammatical one. Some of the proofs can be found in more detail in [7].

2. Recursive characterization of E

Up to now two different methods have been used to describe time bounded computations by means of recursive methods. On the one hand there has been used a certain type of limited general recursion [4, 6] — this is closely related to Cook's characterization since recursive calls can be controlled by means of pushdown tapes — on the other hand limited syntactic recursions have been applied [11, 12].

The recursion scheme used in this paper has the advantage that it is quite a natural generalization of the primitive-recursive scheme. Therefore our characterization gives some new information about the difference between time bounded and tape bounded computations.

Definition. $\mathscr{F}$ is the *smallest class of functions* such that the following conditions hold:

(i) $\lambda x[0], \lambda x[x+1], \lambda x_1 \ldots x_n[x_i]$ for $n \in \mathbf{N}$, $i \in \{1, \ldots, n\}$ and $\lambda xy[x \cdot y]$ belong to $\mathscr{F}$.

(ii) $\mathscr{F}$ is closed under substitution.

(iii) Let k be some natural number and let $g : \mathbf{N}^k \to \mathbf{N}$; $j, l : \mathbf{N}^{k+1} \to \mathbf{N}$; $h : \mathbf{N}^{k+2} \to \mathbf{N}$ be functions belonging to $\mathscr{F}$. Let $l(\bar{x}, y) \leqslant y$ hold for all $\bar{x} \in \mathbf{N}^k$, $y \in \mathbf{N}$ and let $f : \mathbf{N}^{k+1} \to \mathbf{N}$ be defined by:

$$f(\bar{x}, 0) = g(\bar{x})$$

$$f(\bar{x}, y+1) = h(\bar{x}, y, f(\bar{x}, y), f(\bar{x}, l(\bar{x}, y))).$$

If $f(\bar{x}, y) \leqslant j(\bar{x}, y)$ holds for all $\bar{x} \in \mathbf{N}^k$, $y \in \mathbf{N}$, then $f \in \mathscr{F}$.

Therefore the difference between $\mathscr{F}$ and the Grzegorczyk class $\mathscr{E}^2$ is the following: If you want to compute $f(\bar{x}, y+1)$ by means of its recursive scheme then you must know not only the value of $f(\bar{x}, y)$ but also an additional value $f(\bar{x}, y')$, $y' = l(\bar{x}, y) \leqslant y$. In this context it is interesting that we don't leave the class $\mathscr{E}^2$ if we use schemes of the form

$$f(\bar{x}, y+1) = h(\bar{x}, y, f(\bar{x}, l(\bar{x}, y))), l(\bar{x}, y) \leqslant y \qquad \text{(see [8])}.$$

But it is not known whether $\mathscr{F} = \mathscr{E}^2$ holds.

First we show, how much time we need to compute a function out of $\mathscr{F}$.

Lemma 1. *Let $\phi : \mathbf{N}^k \to \mathbf{N}$ be a function out of $\mathscr{F}$. Then there exist natural numbers c, p and a Turing machine M which needs — starting with the binary encoding of a tupel $(x_1, \ldots, x_k) \in \mathbf{N}^k$ on its input tape — not more than $c \cdot (\max_{1 \leqslant i \leqslant k}\{x_i\})^p$ steps in order to write $\phi(x_1, \ldots, x_k)$ in binary notation on its tape.*

Proof. The proof of the lemma is straightforward. It is clear that the initial functions have the above property. Furthermore the growth of all functions out of $\mathscr{F}$ is bounded by functions which can be generated by means of (i) and (ii) — that is without recursion. Therefore to each $f \in \mathscr{F}$ there exist $\alpha, \beta \in \mathbf{N}$ such that $f(\bar{x}) \leqslant \alpha \cdot (\max\{x_i\})^\beta$. The lemma is now proved in the usual way by induction on the number of substitutions and recursions necessary to define the function ϕ. Our recursive scheme does not lead to a greater demand for time than the primitive recursive one. For let f be some function which is defined by either of the recursion schemes. Then we have to compute in both cases successively $f(\bar{x}, 0), f(\bar{x}, 1), \ldots, f(\bar{x}, y)$ in order to get the value $f(\bar{x}, y+1)$. □

In the following we want to go the way in the other direction and show how time bounded computations can be described by means of functions out of $\mathscr{F}$. First we will show that every computation can be simulated by a one-tape Turing machine whose head can move only in a special way.

We say that a one-tape Turing machine has normal form if it can move its head

only in the following way: Starting on the first cell the head moves to the right (it is never allowed to remain on the same cell) until it reaches a cell which has not been scanned before. After that the head moves back to the first cell. This process is started again and again until the machine stops. The first cell is marked in the first step.

To give a formal definition, a Turing machine in normal form is a 6-tuple $M = (S, X, T, \delta, s_0, F)$, where

(1) X and T are finite sets (sets of input symbols and symbols written by M, respectively), $X \cap T = \emptyset$.

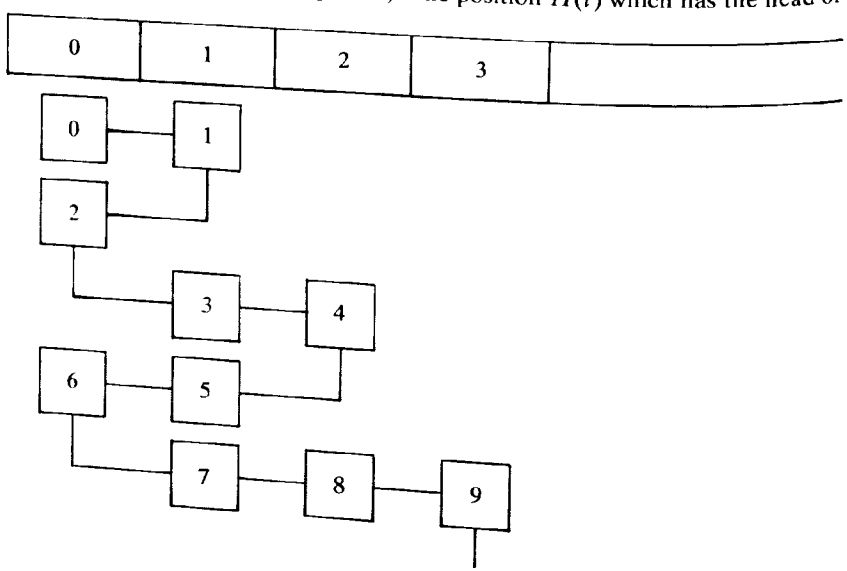
(2) $S = \tilde{S} \times \{l, r\}$ for some finite set $\tilde{S}$. $s_0 \in S$ is the initial state and $F \subset S$ the set of final states.

(3) $\delta : S \times (X \cup T \cup \{B\}) \to S \times T \times \{-1, +1\}$ is a mapping such that $\delta(s, \gamma) = (s', \gamma', \eta)$ for some $s, s' \in S$; $\gamma \in X \cup T \cup \{B\}$, $\gamma' \in T$, $\eta \in \{-1, +1\}$ implies

$$\eta = \begin{cases} +1 & \text{if } s \in \tilde{S} \times \{r\} \wedge \gamma \in T \vee s \in \tilde{S} \times \{l\} \wedge \gamma \in T_1 \\ -1 & \text{otherwise.} \end{cases}$$

Here B denotes the blank symbol and $T_1 \subset T$ is the set of those tape symbols which announce the left end of the tape. In the first step a symbol out of T_1 is written and afterwards it is always just the first cell which stores a symbol out of T_1.

If a Turing machine has normal form then the movement of its head is uniquely determined and does not depend on the input. If we number the cells beginning with 0, then the jth cell of the tape is reached by the Turing machine in its j^2th step for the first time (see Diagram 1). The position $H(t)$ which has the head of M



after t steps is given by $|t - (\sqrt{t})^2 - \sqrt{t}|$, where $\sqrt{t}$ is the greatest natural number such that $(\sqrt{t})^2 \leq t$.

Lemma 2. *Let L be some language which is accepted by a one-tape Turing machine M with the time bound $T(n)$. Then there exists a one-tape Turing machine in normal form which accepts L with the time bound $T(n)^2$.*

Proof. We can assume that M is a 6-tuple $M = (S, X, T, \delta, s_0, F)$, where X is the set of input symbols, T is the set of symbols written by M and $X \cap T = \emptyset$.

It is not difficult to construct a Turing machine $\tilde{M}$ in normal form which simulates M step by step in the following way: If M reaches some configuration then $\tilde{M}$ reaches a configuration where the inscription of the tape is the same as for M except that the first cell and the cell, scanned by the head of M, are marked. In order to simulate the next move of M, the machine $\tilde{M}$ moves its head to the right until it reaches a cell which has not been scanned before. During this process $\tilde{M}$ reads the symbol which is stored in the marked cell and changes the tape inscription as it is determined by the next move function of M as far as it is possible (only in the case where the head of M moves to the left in its next step this is not possible). After M has reached its turning-point it moves back to the first cell. During this process it eventually marks the new head position of M.

It is clear that $\tilde{M}$ needs not more than $T(n)^2$ steps in order to simulate the $T(n)$ steps of M. □

If we want to characterize languages $L \subset X^*$ by functions $f : \mathbf{N} \to \mathbf{N}$, we have to define a bijective mapping $g : X^* \to \mathbf{N}$. We demand the mapping to be bijective (and not only injective) because otherwise the function f is also defined in numbers not belonging to $g(X^*)$ and therefore it contains more information than L. In this case we should not call the correspondence between f and L a characterization.

We may assume that $X = \{1, \ldots, r\}$ for some $r \in \mathbf{N}$. Then a bijective mapping $g_r : X^* \to \mathbf{N}$ is defined by:

$$g_r(\varepsilon) = 0, \quad g_r(va) = a + r \cdot g_r(v) \quad \forall a \in X, v \in X^*.$$

Now let $M = (S, X, T, \delta, s_0, F)$ be a one-tape Turing machine. We may assume that $X = \{1, \ldots, r\}$ and that $S, T \subset \mathbf{N}$. The dynamic behaviour of M is described by the following functions:

$H(y, t)$ = head position which is reached by M after t steps if $g_r^{-1}(y)$ is the input string,

$P(y, t) = 1 + \max\{t' < t \mid t' = 0 \vee H(y, t) = H(y, t')\}$.

$P(y, t)$ determines that moment in the computation of M when the tape symbol, scanned after t steps, has been written.

$S(y, t)$ = state computed by M in its tth step if $g_r^{-1}(y)$ is the input string.

$T(y, t)$ = tape symbol written by M in its tth step if $g_r^{-1}(y)$ is the input string.

If M is a Turing machine in normal form then H and P are functions not depending on the machine M or on the input string. Therefore in this case we will only write $H(t)$ and $P(t)$. We already saw that

$$H(t) = |t - (\sqrt{t})^2 - \sqrt{t}|.$$

Suppose now that $M = (S, X, T, \delta, s_0, F)$ is a Turing machine in normal form and that $S = \{1, \ldots, \sigma\} \subset \mathbf{N}$. Then the next move function δ defines the following two functions

$$\alpha(s, \gamma) = \begin{cases} s' & \text{if } \delta(s, \gamma) = (s', \gamma', \eta) \text{ with some } \gamma' \in T, \eta \in \{-1, +1\}, \\ 0 & \text{if } s \in \{0\} \cup F, \\ \sigma + 1 & \text{otherwise;} \end{cases}$$

$$\beta(s, \gamma) = \begin{cases} \gamma' & \text{if } \delta(s, \gamma) = (s', \gamma', \eta) \text{ with some } s' \in S, \eta \in \{-1, +1\}, \\ 0 & \text{otherwise.} \end{cases}$$

α and β can be regarded as extensions of the component-functions of δ on $\mathbf{N} \times \mathbf{N}$. They are total functions and constant outside finite intervals. α has the value 0 if δ leads to the acceptance of the input string. In the definition of α we assumed that $\delta(s, \gamma)$ is not defined for any $s \in F$.

If we now define a function $I_r : \mathbf{N}^2 \to \{0, \ldots, r\}$, such that $I_r(z, i)$ is just the ith symbol of $g_r^{-1}(z)$ by

$$I_r(z, i) = \begin{cases} 0 & \text{if } i \geq l(g_r^{-1}(z)), \\ x & \text{if } \exists \phi, \psi \in X^* : l(\phi) = i \wedge g_r(\phi x \psi) = z \end{cases}$$

(note that $I_r(z, 0)$ is the leftmost symbol of $g_r^{-1}(z)$), then the functions S and T are described by the following recursion scheme:

$$S(y, 0) = s_0,$$

$$S(y, t+1) = \begin{cases} \alpha(S(y, t), T(y, P(t))) & \text{if } P(t) \neq 1, \\ \alpha(S(y, t), I_r(y, H(t))) & \text{if } P(t) = 1, \end{cases}$$

$$T(y, 0) = 0,$$

$$T(y, t+1) = \begin{cases} \beta(S(y, t), T(y, P(t))) & \text{if } P(t) \neq 1, \\ \beta(S(y, t), I_r(y, H(t))) & \text{if } P(t) = 1. \end{cases}$$

This simple recursion scheme totally describes the computation of a Turing machine. It should be compared with the complicated functions which have to be defined in the usual arithmetization of Turing machines. The behaviour of M in its $(t+1)$th step depends on the current state after t steps and the content of the cell which is scanned after t steps. The content of this cell is just that tape symbol which was written by M when its head scanned the same cell for the last time. Therefore this tape symbol is $T(y, P(t))$. If the cell $H(t)$ has never been scanned before, then $P(t) = 1$ and M reads the $H(t)$th symbol of the input string. This symbol is just $I_r(y, H(t))$ if we assume that the blank symbol B is identified with 0.

Now let us assume that M operates with the time bound d^n for some $d \in \mathbf{N}$. Let $v \in X^*$. Then M needs at most $d^{l(v)}$ steps in order to decide whether v is accepted by M. If we set $y = g_r(v)$ then there exists a $c \in \mathbf{N}$ such that $y^c \geq d^{l(v)}$. Therefore v is accepted by M if and only if $S(g_r(v), g_r(v)^c) = 0$. This leads together with Lemma 1 to our first main theorem.

Theorem 1. *Let $g_r : \{1, \ldots, r\} \to \mathbf{N}$ be the function which we defined above. Then the following holds:*

(1) *Let $f : \mathbf{N} \to \{0, 1\}$ be some function out of $\mathcal{F}$. Then $L_r = \{v \in \{1, \ldots, r\}^* \mid f(g_r(v)) = 1\}$ belongs to E for all $r \geq 2$.*

(2) *Let $L \subset \{1, \ldots, r\}^*$, $r \geq 2$, be some language out of E. Then the function*

$$f(x) = \begin{cases} 1 & \text{if } g_r^{-1}(x) \in L, \\ 0 & \text{otherwise,} \end{cases}$$

belongs to $\mathcal{F}$.

Proof. (1) We have to define a Turing machine M which accepts L_r. Let $v \in \{1, \ldots, r\}$ be the input string and set $x = g_r(v)$. The computation of M consists of two parts: (a) M computes the binary notation of x; (b) M computes the value $f(x)$ (starting from the binary notation of x).

Let $v = \alpha_1 \ldots \alpha_m$, $\alpha_i \in \{1, \ldots, r\}$. Then

$$x = g_r(v) = \alpha_m + \alpha_{m-1} \cdot r + \ldots + \alpha_1 \cdot r^{m-1} \leq r^{l(v)+1}.$$

Therefore M can compute first the unary notation of x and, starting from this unary notation, the binary notation of x in not more than $c \cdot r^{l(v)}$ steps where $c \in \mathbf{N}$ is some number. Afterwards M computes $f(x)$. Because of Lemma 1 there exists a $p \in \mathbf{N}$ such that M needs not more than $c \cdot x^p$ steps to do this. Therefore the total amount of time, needed by M, is bounded above by $c' \cdot r^{p \cdot l(v)}$.

(2) Now let $L \subset \{1, \ldots, r\}^*$ be a language out of E. Let M be a Turing machine which accepts L with the time bound d^n for some $d \in \mathbf{N}$. Furthermore we can assume that M is a one-tape Turing machine in normal form. Because of Lemma 2 and because of the well-known theorem about the simulation of multi-tape machines by one-tape machines [3, Theorem 10.4] there exists to each Turing machine working within time bound $T(n)$ an equivalent one-tape machine in normal form working within time bound $T(n)^4$.

We already described the dynamic behaviour of M by the two functions S and T which were defined by the recursion scheme

$S(y,0)=1$ (s_0 is identified with the number 1),

$$S(y,t+1)=\begin{cases}\alpha(S(y,t),T(y,P(t))) & \text{if } P(t)\neq 1,\\ \alpha(S(y,t),I_r(y,H(t))) & \text{if } P(t)=1,\end{cases}$$

$T(y,0)=0$,

$$T(y,t+1)=\begin{cases}\beta(S(y,t),T(y,P(t))) & \text{if } P(t)\neq 1,\\ \beta(S(y,t),I_r(y,H(t))) & \text{if } P(t)=1.\end{cases}$$

Here α and β are functions which are constant outside finite intervals and therefore they belong to the Grzegorczyk class $\mathscr{E}^0$ ($\mathscr{E}^0\subset\mathscr{E}^2\subset\mathscr{F}$). Furthermore it is shown in [2] that functions like our H belong to $\mathscr{E}^0$ and that $\mathscr{E}^0$ is closed under the operation of limited maximum. Therefore $P\in\mathscr{E}^0$. I_r can be defined by using the functions $D_r(z)=\lfloor z/r\rfloor$ (greatest natural number $\leq z/r$) and $R_r(z)=r-D_r(z)\cdot r$. It is easy to see that $I_r\in\mathscr{E}^0$.

In order to show that $S,T\in\mathscr{F}$, we set $A(y,t)=S(y,t)+(\sigma+2)\cdot T(y,t)$. Then $S(y,t)=R_{\sigma+2}(A(y,t))$ and $T(y,t)=D_{\sigma+2}(A(y,t))$ and in the case $P(t)\neq 1$ from the recursion scheme follows:

$$A(y,t+1)=\alpha(R_{\sigma+2}(A(y,t)),D_{\sigma+2}(A(y,P(t))))+(\sigma+2)\cdot\beta(R_{\sigma+2}(A(y,t)),D_{\sigma+2}(A(y),P(t)))).$$

We get an analogous formula for the case $P(t)=1$ and this leads to the recursive scheme:

$A(y,0)=1$,

$A(y,t+1)=\psi(y,t,A(y,t),A(y,P(t)))$,

where

$$\psi(y,t,x,z)=\begin{cases}\alpha(R_{\sigma+2}(x),D_{\sigma+2}(z))+(\sigma+2)\cdot\beta(R_{\sigma+2}(x),D_{\sigma+2}(z)) & \text{if } P(t)\neq 1,\\ \alpha(R_{\sigma+2}(x),I_r(y,H(t)))+(\sigma+2)\cdot\beta(R_{\sigma+2}(x),I_r(y,H(t))) & \text{if } P(t)=1.\end{cases}$$

Since $\psi\in\mathscr{E}^0\subset\mathscr{F}$ and A is bounded above by a constant we see that $A\in\mathscr{F}$ and therefore also $S\in\mathscr{F}$.

Furthermore we already showed that there exists a $c\in\mathbf{N}$ such that $v\in L$ is equivalent to $S(g_r(v),g_r(v)^c)=0$. Let us define a function $f:\mathbf{N}\to\{0,1\}$ by

$$f(x)=\begin{cases}1 & \text{if } S(x,x^c)=0,\\ 0 & \text{otherwise.}\end{cases}$$

Then $f\in\mathscr{F}$ and $f(x)=1$ holds if and only if $g_r^{-1}(x)\in L$. □

Theorem 1 states that (except of the necessary encoding g_r) the class E is equal to the class of all (0–1)-functions belonging to $\mathscr{F}$. With the methods of this section this result cannot be transferred to arbitrary functions computable by Turing machines in exponential time, because in this case the values of the function are given by whole inscriptions of the Turing tape. The simplicity of our method was just based on the fact that we avoided the encoding of whole tape inscriptions.

It is clear that we can use our arithmetization to prove a version of Kleene's normal form theorem. (It should be noted that $\mathscr{E}^2\subset\mathscr{F}\subset\mathscr{E}^3$.)

Theorem 2. *Let $f:\mathbf{N}\to\{0,1\}$ be some computable function. Then there exist $g,h\in\mathscr{F}$, such that $f(x)=g((\mu/t)[h(x,t)=0])$ holds.*

Proof. Let $M=(S,X,T,\delta,s_0,F)$ be a Turing machine computing f. We can assume that S is a disjunctive union $S=S_0\cup S_1\cup S_2$ such that δ is defined on $S_2\times(X\cup T\cup\{B\})$ and M stops in a state belonging to S_i, $i\in\{0,1\}$, if and only if i is the value of the function f.

Now we set

$$h(x,t)=\begin{cases}0 & \text{if } t=\text{even}\wedge S(x,\tfrac{1}{2}t)\in S_0\vee t=\text{odd}\wedge S(x,\tfrac{1}{2}(t+1))\in S_1,\\ 1 & \text{otherwise,}\end{cases}$$

$$g(t)=\begin{cases}1 & \text{if } t=\text{odd},\\ 0 & \text{otherwise.}\end{cases}$$

Then $g,h\in\mathscr{F}$ and $f(x)=g((\mu/t)[h(x,t)=0])$ holds. □

As mentioned above we cannot transfer this result with our methods to arbitrary-valued functions if we choose the Turing machine to be the underlying machine model. If we consider however a random access machine whose only arithmetical operation is the successor function and which can store in each of its registers a natural number, then we can interpret the content of the first register as the result of the computation. We can describe the dynamic behaviour of such a machine in the same way as above (see also [6]). This leads to a version of Kleene's normal form theorem which is valid for arbitrary-valued computable functions.

In any case we get a proof of Kleene's normal form theorem which does not use any of those complicated encodings whose application make the proof known from the literature so tedious.

3. Grammatical characterization of E

Definition: A grammar $G=(V_N,V_T,P,S)$, where V_N is the set of non-terminal symbols, V_T the set of terminal symbols and $S\in V_N$ the start symbol, is called *context-sensitive* if all rules belonging to P have the form $u_1Au_2\to u_1vu_2$, where $u_1,u_2\in(V_N\cup V_T)^*$, $v\in(V_N\cup V_T)^+$ and $A\in V_N$.

Now let G be a context-sensitive grammar. Let $\zeta:P\to\mathbf{N}$ be some bijective mapping, set $\mathrm{Lab}(P)=\zeta(P)$ and let $F_1\subset\mathrm{Lab}(P)$ be some set. Let D be a derivation induced by G and let $U\in(\mathrm{Lab}(P))^*$. As in [10] we define: U is called control word of D if one of the following conditions is fulfilled:

(i) D is the derivation $u \to v$ and $U = f$, where $u = u_1 u_2 u_3$, $v = u_1 w u_3$ for some $u_1, u_2, u_3, w \in (V_N \cup V_T)^*$ such that $u_2 \to w \in P$ and $\zeta(u_2 \to w) = f$.

(ii) D consists of one word $v \in (V_N \cup V_T)^*$ and $U = \varepsilon$ or $U \in F_1$ and $\zeta^{-1}(U) = u_1 \to u_2$ has the property that u_1 is no subword of v.

(iii) D is a derivation $v_1 \to v_2 \to v_3$ and $U = U_1 U_2$, where U_1 is the control word of $v_1 \to v_2$ and U_2 is the control word of $v_2 \to v_3$.

The control word of a derivation gives us the numbers of all rules which are used in the derivation in the succession of their application. If a rule with a number $f \in F_1$ is not applicable then this rule can be omitted and the derivation continues at the next rule of the control word.

We define:

$$L(G, C, F_1) = \{w \in V_T \mid \exists \text{ derivation } D : S \xrightarrow[G]{*} w \text{ and } U \in C \text{ such that } U \text{ is control word of } D\}.$$

In this paper we consider context-sensitive grammars G and context-free control sets C. We set:

$$\mathcal{L}(1) = \{L(G, C, F_1) \mid G \text{ is a context-sensitive grammar, } C \subset (\text{Lab}(P))^* \text{ is a context-free language and } F_1 \subset \text{Lab}(P)\},$$

$$\mathcal{L}(0) = \{L(G, C, \emptyset) \mid G \text{ is a context-sensitive grammar and } C \subset (\text{Lab}(P))^* \text{ is a context-free language}\}.$$

In [10] our families $\mathcal{L}(1)$ and $\mathcal{L}(0)$ are denoted by $\mathcal{L}(1,2,1)$ and $\mathcal{L}(1,2,0)$, respectively. We will show now that $\mathcal{L}(0) = \mathcal{L}(1) = E$. In this proof we use the characterization of E by means of writing pushdown acceptors. The reader should remember that deterministic and non-deterministic writing pushdown acceptors define the same families of languages. First we prove that $\mathcal{L}(1) \subset E$. This result is not trivial since the control word of a derivation can be far longer than the derivation itself.

Lemma 3. $\mathcal{L}(1) \subset E$.

Proof. Let $G = (V_N, V_T, P, S)$ be a context-sensitive grammar, let $C \subset (\text{Lab}(P))^*$ be a context-free language and let $P_1 \subset \text{Lab}(P)$. Set $L = L(G, C, P_1)$. We will now define a non-deterministic writing pushdown acceptor $M = (S, X, T, \delta, s_0, F)$ which accepts L. We can assume that M has in addition to its pushdown tape an input tape and a working tape (because of this arrangement we avoid writing on two tracks). A configuration of M is a 6-tuple (s, w, v, u, i, j) where $s \in S$; $w \in X^*$; $u, v \in T^*$ and $i, j \in \mathbf{N}$ denote the state, the inscription on the input tape, working tape, pushdown tape and the head positions on the input tape and the working tape. Furthermore let K be a non-deterministic pushdown automaton accepting C. We can assume that K moves its head one cell to the right in each step.

M simulates (non-deterministically) on its working tape an arbitrary derivation D of the grammar G and simultaneously it simulates on its pushdown tape the computation of K where K takes the control word of D as its input word. M accepts a word $w \in X^*$ if and only if it reaches a configuration where the inscription on the working tape is w and the pushdown tape is empty.

Now let $w \in X^*$ be an arbitrary word and let D be some derivation $S \xrightarrow[G]{*} v$ with control word U. In this case M will reach the configuration $(\tilde{s}, w, v, \tilde{u}, 1, 1)$ where $\tilde{s}$ is the state and $\tilde{u}$ is the inscription on the pushdown tape which can be reached by K starting from the input string U.

We have to show how a derivation step of the grammar G is simulated by M. Let $\alpha \to \beta$ be an arbitrary (non-deterministically chosen) rule out of P and let f be the number associated with this rule.

Then M first examines whether α is a subword of v. If this is true, then M chooses non-deterministically a place where α occurs (that means: $v = v_1 \alpha v_2$ with $v_1, v_2 \in T^*$) and replaces on its working tape v by $v_1 \beta v_2$, if $l(v_1 \beta v_2) \leq l(w)$. The state and the inscription on the pushdown tape are changed according to the next step which is performed by K when it reads the input symbol f. That means, if δ is the next-move function of K and if $\tilde{u} = \bar{u} \circ \gamma$ for some $\gamma \in T$, then M can proceed to every configuration $(s', w, v_1 \beta v_2, \bar{u} \circ \gamma', 1, 1)$ such that $v = v_1 \alpha v_2$ and $(s', \gamma') \in \delta(s, f, \gamma)$. M stops if $l(v_1 \beta v_2) > l(w)$.

Now let us consider the case that α is not a subword of v. If $f \notin P_1$, then M stops. If however $f \in P_1$, then M can proceed to any configuration $(s', w, v, \bar{u} \circ \gamma', 1, 1)$ such that $(s', \gamma') \in \delta(s, f, \gamma)$. Here again we assume that $u = \bar{u} \circ \gamma$ for some $\gamma \in T$.

G is a context-sensitive grammar. Let $v, w \in X^*$ be two words with $l(v) \leq l(w)$. Then obviously there exists a derivation $S \xrightarrow[G]{*} v$ with a control word $U \in C$ if and only if M can proceed from the configuration $(s_0, w, S, \gamma_0, 1, 1)$—where s_0 is the initial state and γ_0 the initial symbol on the pushdown tape of K—to some configuration $(\tilde{s}, w, v, \varepsilon, 1, 1)$.

In order to decide whether there exists a derivation $S \xrightarrow[G]{*} w$ with a control word $U \in C$, M performs the following operations: (1) M starts in the configuration $(s_0, w, s, \gamma_0, 1, 1)$. (2) M simulates on its working tape an arbitrary derivation of G (as far as the derivated string is not too long) and on its pushdown tape the computation of K taking the control word as the input string. (3) When no non-terminal symbol is left on the working tape, then M examines whether the inscription on the working tape is w. (4) M accepts w if and only if the question in (3) is positively answered.

Thus M accepts the language $L(G, C, P_1)$. Furthermore we constructed M so that the length of its inscription on the working tape is always bounded by the length of the input string. □

Lemma 4. $E \subset \mathscr{L}(0)$.

Proof. Let $M = (Q, X, T, \delta, q_0, F)$ be a non-deterministic writing pushdown acceptor. Let us assume that M can write on its input tape, that it has no further Turing tape and that during its whole computation it does not leave the tape segment determined by the input word. Furthermore M does not write more than two symbols on its pushdown tapes in one step.

First we consider a linear bounded automaton $M_1 = (Q, X, T, \delta_1, q_0, F)$ which is given by M simply omitting the pushdown tape and let M_1 perform all moves which can be performed by M for some inscription of its pushdown tape. That means, if $\delta : Q \times T \times T \to Q \times \{-1, 0, +1\} \times T \times T^*$ is the nextmove function of M, then $\delta_1 : Q \times T \to Q \times \{-1, 0, +1\} \times T$ is defined by:

$$(s', \eta, \gamma') \in \delta_1(s, \gamma) \iff \exists \gamma_1 \in T, \tilde{\gamma} \in \{\varepsilon\} \cup T \cup T^2 : (s', \eta, \gamma', \tilde{\gamma}) \in \delta(s, \gamma, \gamma_1).$$

The moves of M_1 can be described in the usual way [3, Theorem 7.4] by means of a context-sensitive grammar

$$G = ((Q \cup \{\varepsilon\}) \times (T \cup \{\varepsilon\}) \times X \cup \{S\}, X, P, S),$$
$$S \notin (Q \cup \{\varepsilon\}) \times (T \cup \{\varepsilon\}) \times X.$$

G is defined in such a way that whenever M_1 can proceed from a configuration $q_0 a_1 \dots a_n$; $a_1, \dots, a_n \in X$ to some configuration $\alpha_1 \dots \alpha_i q \alpha_{i+1} \dots \alpha_r a_{r+1} \dots a_n$ with $\alpha_1, \dots, \alpha_r \in T$; $a_{r+1}, \dots, a_n \in X$; $q \in Q$, then the derivation

$$S \xrightarrow{*} (\varepsilon, \alpha_1, a_1) \dots (\varepsilon, \alpha_i, a_i)(q, \alpha_{i+1}, a_{i+1}) \dots (\varepsilon, \alpha_r, a_r)(\varepsilon, \varepsilon, a_{r+1}) \dots (\varepsilon, \varepsilon, a_n)$$

is induced by the rules of G. Here we assume that M can write no symbol belonging to X and therefore M_1 has not scanned the string $a_{r+1} \dots a_n$ up to this time.

There are three types of rules in P.

(i) rules, which generate the "input word": $S \to S(\varepsilon, \varepsilon, a)$, $S \to (q_0, \varepsilon, a)$ belong to P for all $a \in X$;

(ii) rules, which simulate one step of M_1: Let $\nu_i = (q_i, \alpha_i, a_i) \in (Q \cup \{\varepsilon\}) \times (T \cup \{\varepsilon\}) \times X$ for $i = 1, 2, 3$, with $q_1 = q_3 = \varepsilon$ and $q_2 \in Q$. Set $\beta = \alpha_2$ if $\alpha_2 \neq \varepsilon$ and $\beta = a_2$ if $\alpha_2 = \varepsilon$. Then the following rules belong to P:

$$\nu_1 \nu_2 \nu_3 \to \nu_1(q', \alpha', a_2)\nu_3 \quad \text{if } (q', \alpha', 0) \in \delta_1(q_2, \beta),$$
$$\nu_1 \nu_2 \nu_3 \to \nu_1(\varepsilon, \alpha', a_2)(q', \alpha_3, a_3) \quad \text{if } (q', \alpha', 1) \in \delta_1(q_2, \beta),$$
$$\nu_1 \nu_2 \nu_3 \to (q', \alpha_1, a_1)(\varepsilon, \alpha', a_2)\nu_3 \quad \text{if } (q', \alpha', -1) \in \delta_1(q_2, \beta).$$

These rules are not context-sensitive but we know how equivalent context-sensitive rules can easily be constructed.

(iii) terminal rules: $(\varepsilon, \alpha, a) \to a$, $(q, \alpha, a) \to a$ belong to P for all $a \in X$, $\alpha \in T \cup \{\varepsilon\}$, $q \in F$.

In the usual way it follows that $S \xrightarrow[G]{*} w \in X^*$ holds if and only if w is accepted by M_1.

Now let M be our writing pushdown acceptor and let L be the language accepted by M. We have to show that there exists a context-sensitive grammar G and a context-free language C such that $L = L(G, C, \emptyset)$. We take G to be the grammar defined above. By means of the control language we will now guarantee that only those rules of G can be applied which belong to the actual inscription of the pushdown tape. In order to do this we split the set P up into disjunctive sets $P_0, P_1, P(\gamma, \tilde{\gamma})$ with $\gamma \in T, \tilde{\gamma} \in \{\varepsilon\} \cup T \cup T^2$.

Here P_0 is the set of all rules defined in (i) and P_1 is the set of all rules defined in (iii). Now set $\nu_i = (q_i, \alpha_i, a_i)$ for $i = 1, 2, 3$, with $q_1 = q_3 = \varepsilon$, $a_2 \in Q$ and set $\beta = \alpha_2$ if $\alpha_2 \neq \varepsilon$; $\beta = a_2$ if $\alpha_2 = \varepsilon$. Then the rules

$$\nu_1 \nu_2 \nu_3 \to \nu_1(q', \alpha', a_2)\nu_3, \quad \text{or}$$
$$\nu_1 \nu_2 \nu_3 \to \nu_1(\varepsilon, \alpha', a_2)(q', \alpha_3, a_3), \quad \text{or}$$
$$\nu_1 \nu_2 \nu_3 \to (q', \alpha_1, a_1)(\varepsilon, \alpha', a_2)\nu_3$$

belong to $P(\gamma, \tilde{\gamma})$ if and only if $(q', \alpha', \eta, \tilde{\gamma}) \in \delta(q_2, \beta, \gamma)$ with $\eta = 0$ or $\eta = 1$ or $\eta = -1$, respectively.

We set

$$P_2 = \bigcup_{\gamma \in T, \tilde{\gamma} \in \{\varepsilon\} \cup T \cup T^2} P(\gamma, \tilde{\gamma}).$$

Then $P = P_0 \cup P_1 \cup P_2$. In order that $P_0, P_1, P(\gamma, \tilde{\gamma})$ are disjunctive sets we have to permit that some rules out of P occur in P more than once and that these rules are associated with different markers.

We choose $C = P_0^* \circ C_1 \circ P_1^*$ with a certain $C_1 \subset P_2^*$. The rules out of P_0 generate an encoding of the input word, the rules out of P_1 generate the terminal word. The computation of M is simulated by the rules belonging to P_2 in such a way that the rule itself describes the alteration on the Turing tape and the marker of the rule describes the alteration on the pushdown tape. To make this clear we use a homomorphism $h : P_2^* \to T_2^*$, $T_2 = T \times (\{\varepsilon\} \cup T \cup T^2)$, which is defined by

$$h(f) = (\gamma, \tilde{\gamma}), \quad \forall (\gamma, \tilde{\gamma}) \in T_2, f \in P(\gamma, \tilde{\gamma}).$$

A word out of T_2^* totally describes all alterations which are performed on the pushdown tape. We define a context-free language $C_2 \subset T_2^*$ such that a word $(\gamma_1, \tilde{\gamma}_1)(\gamma_2, \tilde{\gamma}_2) \dots (\gamma_r, \tilde{\gamma}_r) \in C_2$ is the control word (or more exactly: homomorphic image of the control word) of a derivation in P_2 if and only if the automaton M reads in its ith step, $1 \leq i \leq r$, on its pushdown tape the symbol γ_i and replaces it by $\tilde{\gamma}_i$.

Therefore if $\tilde{\gamma}_i \in T$ or $\tilde{\gamma}_i = \beta_1 \beta_2 \in T^2$ for some $i \in \{1, \dots, r-1\}$ then γ_{i+1} must be equal to $\tilde{\gamma}_i$ or β_2, respectively. However if $\tilde{\gamma}_i = \varepsilon$ (this means that the uppermost

symbol on the pushdown tape, in our case γ_i, has to be erased), then γ_{i+1} is uniquely determined by $(\gamma_1, \bar{\gamma}_1) \ldots (\gamma_i, \bar{\gamma}_i)$ to be the rightmost symbol of some $\bar{\gamma}_j$ which has not been erased yet. This can be described formally in the following way:

We define a partial mapping $\psi : T_2^* \to 2^{T_2}$ for all $\gamma_1, \gamma_2, \gamma_3 \in T$, $u \in T_2^*$ by:

$$\psi((\gamma_1, \gamma_2)(\gamma_2, \varepsilon)) = \{(\gamma_1, \varepsilon)\}, \psi\{((\gamma_1, \gamma_2\gamma_3)(\gamma_3, \varepsilon)) = \{(\gamma_1, \gamma_2)\},$$

$$\psi(u) = \{u_1 u_2 u_3 \mid \exists v \in T_2^* : u = u_1 v u_3 \wedge \psi(v) = u_2\}.$$

Let ψ^* be the transitive closure of ψ. Then we set

$$C_2 = \{v \in T_2^* \mid (\gamma_0, \varepsilon) \in \psi^*(v)\}, \qquad C = P_0^* \circ h^{-1}(C_2) \circ P_\uparrow^*.$$

C_2 is a context-free language and therefore C is a context-free language, too. Furthermore we defined C_2 in such a way that a control word out of C allows only derivation in G describing alterations of the Turing tape which are induced by the actual inscription of the pushdown tape. Therefore $L(G, C, \emptyset) = L$ holds. □

From Lemmas 3 and 4 we get the theorem:

Theorem 3. $\mathscr{L}(0) = \mathscr{L}(1) = E$.

References

[1] S.A. Cook, Characterizations of pushdown machines in terms of time-bounded computers, *J. ACM* **18** (1971) 4–18.

[2] A. Grzegorczyk, Some classes of recursive functions, *Rozprawy Matematyczne* **4** (1953) 1–46.

[3] J.E. Hopcroft and J.D. Ullman, *Formal languages and their Relation to Automata* (Addison-Wesley, Reading, Mass., 1969).

[4] H. Kopp, Bemerkungen zur Übersetzbarkeit rekursiver in nicht rekursive Unterprogrammstrukturen, Fachbereich Angewandte Mathematik und Informatik der Universität des Saarlandes, A 74/12.

[5] G. Mager, Writing pushdown acceptors, *J. Comput. System Sci.* **3** (1969) 276–318.

[6] B. Monien, Charaterizations of time-bounded computations by limited primitive recursion, 2nd Colloquium, Automata, Languages and Programming (1974) 280–293.

[7] B. Monien, Rekursive und Grammatikalische Charakterisierung der Exponentialzeitsprachen und ein neuer Beweis des Kleene'schen Normalformtheorems, Institut für Informatik der Universität Hamburg, Bericht Nr. 14 (1975).

[8] H. Müller, personal communication.

[9] R.W. Ritchie, Classes of predictably computable functions, *Trans. Am. Math. Soc.* **106** (1963) 139–173.

[10] A. Salomaa, *Formal Languages* (Academic Press, New York, 1973).

[11] D.B. Thompson, Subrecursiveness: Machine-independent notions of computability in restricted time and storage, *Math. Systems Theory* **6** (1972) 3–15.

[12] K. Weihrauch, Teilklassen primitiv-rekursiver Wortfunktionen (Dissertation), Berichte der GMD, Nr. 91 (1974).