CHARACTERIZATIONS OF TIME-BOUNDED COMPUTATIONS BY LIMITED PRIMITIVE RECURSION

Burkhard Monien

Universität Hamburg, Institut für Informatik

Complexity measures usually are based on a machine model (1-tape Turing machine, multi-tape Turing machine, bounded activity machine, random access machine). The class of all functions which are computable in polynomial time is the same for a wide variety of machines (for characterizations of this class see A. Cobham [2], D.B. Thompson [8], and S.A. Cook [3]). On the other hand the classes L_k of all functions which are computable in time $c \cdot n^k$ for a fixed $k \in \mathbb{N}$ depend very much on the machine model taken as a basis for the complexity measure.

Whereas the machine models can be described easily, not much is known about their complexity classes. In this paper we consider a simple random access machine (RAM) which works in many aspects like a real computer. It has an unbounded sequence of registers, a bounded number of "index registers" and has access to each register by means of indirect addressing. Furthermore the contents of any index register can be increased by 1 or compared with 0 in one step. We show that the classes L_k (and more generally the complexity classes $C_R(T(n))$ determined by primitive recursive time functions $T: \mathbb{N} \to \mathbb{N}$) which belong to this machine model can be described by using primitive recursive methods.

We define a class P_Σ (where Σ is any finite alphabet) of primitive recursive functions mapping $\mathbb{N}_0 \times \Sigma^*$ into $\mathbb{N}_0$ as the smallest class which contains some initial functions and which is closed under substitution and a special form of limited primitive recursion. We show that

$$C_R(T(n)) = F(T(n)) := \bigcup_\Sigma \left\{ f: \Sigma^* \to \mathbb{N}_0 \;\middle|\; \begin{array}{l} \exists\, \tilde{f} \in P_\Sigma \text{ such that} \\ f(w) = \tilde{f}(T(l(w)), w) \quad \forall w \in \Sigma^* \end{array} \right\}$$

holds for each RAM-honest function T.

Therefore the family of all complexity classes of the RAM is described by the class $\bigcup_{\Sigma} F_{\Sigma}$. If we take as additional basic operation a function or a predicate belonging to F_{Σ} , this does not increase the speed of a random access machine by more than a constant factor.

Furthermore there is a relationship between time bounded RAMs and pushdown automata with counters. It is well known ([4], [5]) that any language which is acceptable by a deterministic pushdown automaton which has in addition to its pushdown tape a counter of length L(n) (CPDA with counter length L(n)) can be recognized by a RAM within time c·L(n). The language $\{ucv \mid v \text{ substring of } u\}$, for example, which belongs to the "string matching problem" is accepted by a deterministic two-way pushdown automaton (CPDA with counter length n) and this implies that it is recognized by a RAM in linear time. On the other hand it was shown by the author ([6]) that any language which is recognized by our RAM with time bound T(n) is acceptable also by a CPDA with counter length $T(n)\cdot(\log T(n))^2$.

In this paper we show that the classes P(L(n)), consisting of all functions which are computable by CPDAs with counter length L(n), can also be described by using recursive methods, and by the application of similar methods in describing the complexity classes of both, the RAM and the CPDA, the close relationship between these two families of complexity classes becomes quite evident.

For each partial recursive function $f: \mathbb{N}_0 \longrightarrow \mathbb{N}_0$ we define classes G(f(n)), H(f(n)) of partial recursive functions as the smallest classes which contain some initial functions and which are closed under substitution and a special form of limited recursion. We show that for each CPDA-honest function f(n) the following holds:

$$P(f(n)) = G(f(n)) \subset H(f(n)) \subset F(f(n)) = C_R(f(n))$$

$$H(f(n)) \subset G(f(n)\cdot\log f(n))$$

$$P(f(n)) \subset H(f(n)\cdot\log f(n))$$

As a corollary we get:

$$P(f(n)) \subset C_R(f(n))$$

$$C_R(f(n)) \subset P(f(n)\cdot(\log f(n))^2)$$

It seems possible that some further insight into the structure of complexity classes can be gained by applying these recursive methods.

The results presented in this paper can be found with more details in [7].

1. Definition of the RAM

A random access machine (RAM) consists of a finite program which operates on an infinite sequence of registers. Each register can store any natural number (including 0). The contents of the registers are denoted by the sequence $x_0, x_1, x_2, \ldots$. The program consists of instructions of the following types:

$$x_i \leftarrow x_{x_j}, \quad x_{x_i} \leftarrow x_j, \quad x_i \leftarrow x_j + 1$$

$$\text{Read } x_i, \quad \text{TRA m if } x_i > 0$$

(the operation $x_i \leftarrow x_j + x_k$ is not allowed in one step and this is the main difference to Cook's RAM [4])

The effect of most of these instructions is evident. Normally the control of the program is transferred from one line to the next. The instruction TRA m if $x_i > 0$ causes the control to be transferred to line m of the program if $x_i > 0$. The RAM halts if the control of the program is transferred to a line with no instruction.

At the beginning of the computation of a RAM M the control of the program points to the first line, the contents of all registers is 0 and a sequence $i_1, i_2, \ldots, i_n, 0$ with $i_k \in \mathbb{N}$, $k=1,2,\ldots,n$, serve as inputs. The instruction Read x_i causes M to transfer the first number which has not been read in up to this time into register i.

Let M be a RAM which starts with the input sequence $i_1, \ldots, i_n, 0$. Then $M(i_1,\ldots,i_n)$ denotes the contents of the register 0 after the end of the computation.

Definition: M computes a function $f: \Sigma^* \to \mathbb{N}_0$, where $\Sigma = \{a_1,\ldots,a_r\}$ is a finite nonempty set, iff $M(i_1,\ldots, i_n) = f(a_{i_1}\ldots a_{i_n})$ for all $n \in \mathbb{N}_0$ and all $i_k \in \{1,\ldots, r\}$, $k=1,\ldots,n$.

Definition: M computes a function $f: \Sigma^* \to \mathbb{N}_0$ with time bound $T(n)$, iff

a. M computes f

b. For every input sequence $i_1,\ldots, i_n$, 0, where $i_k \in \{1,\ldots, r\}$ $\forall k=1,\ldots,n$, M halts after at most $T(n)$ steps.

$C_R(T(n))$ denotes the class of all functions which are computable by a RAM with time bound $d\cdot T(n)$ for some constant $d \in \mathbb{N}$.

It is not difficult to show that a multi-tape RAM (that is a random access machine which operates on a finite number of infinite sequences of registers and the program of which consists of instructions of the form $x_i^s r \leftarrow x_j^t$, ...) can be simulated on our normal RAM without enlarging the time bound by more than a constant factor.

2. Description by schemes with arrays

A RAM can be described quite naturally using notions which are defined in the theory of program schemes with arrays and in many programming languages.

Let x,y,z,... be variables and let A,B,C,... be arrays. We consider finite programs which consist of instructions of the form:

y = x + 1, y = A(x), A(y) = x,

read (y), If x > 0 then goto m,

with the usual semantic interpretation. This interpretation includes a certain storage structure since the value of x or of A(x) is available without loosing time.

It is evident that there is a bijective correspondence between such programs and RAMs (1 array $\leftrightarrow$ 1 infinite sequence of registers) and that the time bound is not changed by this correspondence (the time bound of a program is defined

as the number of instructions which are performed during the computation).

If we want to describe the behaviour of a multi-tape Turing machine by a similar program we have to consider the fact that a Turing machine has access only to the cells which are scanned by its heads and that its heads can reach in one step only a neighbouring cell. In order to describe this behaviour we have to introduce two types of variables.

Let x,y,z,... be assignment variables, let c,d,... be counter variables and let A,B,C,... be arrays. Then there is a bijective correspondence between multi-tape Turing machines and finite programs which consist of instructions of the form:

$$y = a \quad (a \in \mathbb{N}_0),\ c = c + 1,\ c = c \dot{-} 1,$$

$$x = A(c),\ A(c) = x,\ \text{read } (x),\ \text{If } x = a \text{ then goto } m$$

The time bound is changed by this correspondence (1 array $\leftrightarrow$ 1 tape, 1 counter variable $\leftrightarrow$ 1 head) at most by a constant factor.

These simple considerations already show that the complexity classes of a RAM can be described more easily than the complexity classes of a Turing machine if we take methods which are used in mathematics and programming theory.

3. Characterization of the complexity classes by limited primitive recursion

In this section we will characterize time-bounded computations of a RAM by classes of functions which we define by limited primitive recursion. In order to do this we have to define an arithmetization of the RAM and we cannot apply the usual method (where the whole inscription of the tape is coded by a number) because in this case the growth of the functions is determined by the tape bound and not by the time bound.

Let M be a RAM. Let $\Sigma = \{a_1, \ldots, a_r\}$ be a finite set. In order to describe the computation of M for a given $w \in \Sigma^*$ we introduce the following functions:

P(t,w) - position of the control of the program after t steps of M

K(t,w) - position of the first number in the input sequence which has not been

read in after t steps of M

$L(t,w)$ - number of the register the contents of which is altered in the t-th step of M

$I_j(t,w)$- contents of the j-th register after t steps of M (for all j such that x_j appears in the program of M)

$I(t,w)$ - number which is transferred in the t-th step of M

It should be noted that all these functions are lineary bounded in t. These functions can be defined simultaneously by primitive recursion if we use the maximum-operator which associates with two functions $f,g: \mathbb{N}_0 \times \Sigma^* \to \mathbb{N}_0$ a function $[f^{-1}g]: \mathbb{N}_0 \times \Sigma^* \to \mathbb{N}_0$ by:

$$[f^{-1}g](x,w) = \max\left\{y \in \mathbb{N}_0 \mid y = 0 \vee y \leq x \wedge f(y) = g(x)\right\}$$

For the sake of brevity we don't give the whole defining scheme here. For example L, I are defined by:

$$L(t+1,w) = \begin{cases} 0, & \text{if } P(t,w) \triangleq \text{no instruction} \vee \text{TRA m if } x_i > 0 \\ i, & \text{if } P(t,w) \triangleq x_i \leftarrow x_{x_j} \vee x_i \leftarrow x_j + 1 \vee \text{Read } x_i \\ I_i(t,w), & \text{if } P(t,w) \triangleq x_{x_i} \leftarrow x_j \end{cases}$$

$$I(t+1,w) = \begin{cases} I([L^{-1}I_j](t,w),w), & \text{if } P(t,w) \triangleq x_i \leftarrow x_{x_j} \\ I_j(t,w), & \text{if } P(t,w) \triangleq x_{x_i} \leftarrow x_j \\ I_j(t,w) + 1, & \text{if } P(t,w) \triangleq x_i \leftarrow x_j + 1 \\ i(K(t,w),w), & \text{if } P(t,w) \triangleq \text{Read } x_i \\ 0, & \text{if } P(t,w) \triangleq \text{no instruction} \\ & \quad \vee \text{ TRA m if } x_i > 0 \end{cases}$$

We write $P(t,w) \triangleq 4$ if 4 is the instruction in the P(t,w)-th line of the program. Of course this fact can described by using a simple function. In the definition of I the maximum-operator is used to determine that point in the computation of M in which the contents of the register j have been changed for the last time.

We define a class of functions in the following way:

Let $\Sigma = \{a_1,..,a_r\}$ be a finite set. Let $\xi: \Sigma \to \{1,...,r\}$ be the mapping which is defined by $\xi(a_i) = i$ for all $i = 1,..,r$. Then the class P is defined by:

i) initial functions

$p(x,w) = x$, $s(x,w) = x + 1$, $c_0(x,w) = 0$,

$$i(x,w) = \begin{cases} 0, & \text{if } x = 0 \vee x > l(w) \\ \xi(b_x), & \text{if } 1 \leq x \leq l(w),\ w = b_1 \ldots b_{l(w)} \end{cases}$$

are elements of F_Σ

ii) substitution

If $f,g \in F_\Sigma$ then the function $\lambda x,w\,[f(g(x,w),w)]$ is an element of F_Σ

iii) some kind of primitive recursion

Let $s,q \in \mathbb{N}$ and let f_{ij} $(1 \leq i \leq s,\ 1 \leq j \leq q)$ be elements of F_Σ. Let the functions $\varphi_1,\ldots,\varphi_s$ be defined by the scheme:

$$\varphi_i(0,w) = d_i \quad (d_i \in \mathbb{N}_0) \qquad \forall i = 1,\ldots,s$$

$$\varphi_i(x+1,w) = \begin{cases} f_{i1}(\chi_{i1}(x,w),w), & \text{if } p_{i1}(x,w) \\ f_{i2}(\chi_{i2}(x,w),w), & \text{if } p_{i2}(x,w) \\ \cdot & \\ \cdot & \\ f_{iq}(\chi_{iq}(x,w),w), & \text{if } p_{iq}(x,w) \end{cases} \qquad \forall i = 1,\ldots,s$$

where:

1. $\chi_{ij}(x,w) = \alpha_{ij1}([\alpha^{-1}_{ij2}\alpha_{ij3}]\,(x,w),w)$
   with some $\alpha_{ij1},\alpha_{ij2},\alpha_{ij3} \in F_\Sigma \cup \{\varphi_1,\ldots,\varphi_s\}$
2. $p_{ij}(x,w)$ can be composed of expressions of the form $g(\alpha_1([\alpha_2^{-1}\alpha_3]\,(x,w),w) = 0$ by means of $\neg, \wedge, \vee$ (where $g \in F_\Sigma$ and $\alpha_1, \alpha_2, \alpha_3 \in F_\Sigma \cup \{\varphi_1,\ldots,\varphi_s\}$).
3. For any $x \in \mathbb{N}_0$, $w \in \Sigma^*$, $i \in \{1,\ldots,s\}$ there is exactly one $j \in \{1,\ldots,q\}$ such that $p_{ij}(x,w)$ holds.

$\varphi_1,\ldots,\varphi_s$ are elements of F_Σ if there is a $c \in \mathbb{N}$ such that $\varphi_i(x,w) \leq c \cdot x$ for all $i = 1,\ldots,s$, $x \in \mathbb{N}$, $w \in \Sigma^*$.

iv) If G is a class of functions such that i), ii), iii) hold then $F_\Sigma \subset G$.

So our class F_Σ consists only of functions in two variables. Substitution and

primitive recursion are allowed only for the first variable. Classes of primitive recursive functions usually are closed under the application of simultaneous recursions including case distinctions. For our class we have to mention this explicitely in the definition because there seems to be no other way to define a class of functions in one variable with this property.

Our arithmetization of the RAM now leads to the following definition:

$$F_\Sigma(T(n)) = \{ f: \Sigma^* \to \mathbb{N}_0 \mid \exists \tilde{f} \in F_\Sigma \text{ such that } f(w) = \tilde{f}(T(1(w)),w) \ \forall w \in \Sigma^* \}$$

$$F(T(n)) = \bigcup_\Sigma F_\Sigma(T(n))$$

Theorem 1: $F(T(n)) = C_R(T(n))$ for each RAM-honest function $T: \mathbb{N}_0 \to \mathbb{N}_0$

We already indicated how $C_R(T(n)) \subset F(T(n))$ is proved. To prove the other direction it is shown that for each $f \in F_\Sigma$, $y \in \mathbb{N}$, $w \in \Sigma^*$ $f(0,w)$, $f(1,w),\ldots,f(y,w)$ can be computed by a RAM which uses no more than $c \cdot y$ steps (where c depends on f but not on y or w).

4. Pushdown automata with counters and the recursive description of their complexity classes

A deterministic counter pushdown automaton (CPDA) consists of a finite memory, an input tape (alphabet Σ) the head of which may move in both directions and may move off the input string to the right, and a pushdown tape. Such an automaton M defines a partial mapping f_M, $f_M: \mathbb{N}_0 \times \Sigma^* \to \mathbb{N}_0$. M starts with $\vdash w \dashv$ ($w \in \Sigma^*$) on its input tape, with its input head on the n-th cell ($n \in \mathbb{N}_0$) and with a distinguished symbol γ_0 on its pushdown tape. M stops when its pushdown tape is empty. In this case the position of the input head is the result $f_M(n,w)$ of the computation. If M does not stop then $f_M(n,w)$ is undefined.

It is known that such an automaton can compute each partial-recursive function. We define complexity classes by bounding the maximum counter length: $P^+_\Sigma(L(n))$ consists of all functions which are computable by a CPDA which has the following property:

There exists a $c \in \mathbb{N}$ such that for each input string $w \in \Sigma^*$ and for each initial position $x \in \mathbb{N}_0$ of the input head, this input head does not leave the first $c \cdot \max\{x, L(1(w))\}$ cells during its computation.

(It should be noted that for a "smooth" function L it is equivalent whether the automaton has one counter with maximum length L(n) or k counters with maximum length $(L(n))^{1/k}$ each.)

We will characterize the classes $P_\Sigma^+(L(n))$ by using recursive schemes of the form:

$$f(x,w) = \begin{cases} \alpha(x,w), & \text{if } x \equiv 0 \bmod k \\ f(\beta_\nu(f(\gamma_\nu(x,w),w),w),w), & \text{if } x \equiv \nu \bmod k \quad (1 \leq \nu \leq k-1) \end{cases}$$

with some $k \in \mathbb{N}$ and total functions $\alpha, \beta_1, \ldots, \beta_{k-1}, \gamma_1, \ldots, \gamma_{k-1}$.

A scheme of this form defines a partial function $f: \mathbb{N}_0 \times \Sigma^* \to \mathbb{N}_0$ in the usual way. Furthermore it defines an algorithm for the computation of f(x,w) in an obvious way. This algorithm stops iff f(x,w) is defined. During this algorithm for a given $x \in \mathbb{N}_0$, $w \in \Sigma^*$ the function f has to be evaluated at several points (y,w). Let $IP_{x,w}$ (set of intermediate points) be the set of all such y. Formally $IP_{x,w}$ is defined by:

i) If $x \equiv 0 \bmod k$ then $IP_{x,w} = \{\alpha(x,w)\}$

ii) If $x \equiv \nu \bmod k$ $(1 \leq \nu \leq k-1)$ then $IP_{x,w} = IP_{y,w} \cup IP_{z,w}$, where $y = \gamma_\nu(x,w)$ and $z = \beta_\nu(f(\gamma_\nu(x,w),w),w)$

$IP_{x,w}$ is a well defined finite set if f(x,w) is defined.

In the following we will consider only schemes to which there exist a function $L: \mathbb{N}_0 \to \mathbb{N}_0$ and a $c \in \mathbb{N}$ such that $y \leq c \cdot \max\{x, L(1(w))\}$ holds for all $y \in IP_{x,w}$ and for all $x \in \mathbb{N}_0$, $w \in \Sigma^*$. (This condition corresponds to the counter length of a CPDA)

Let $\Sigma = \{a_1, \ldots, a_r\}$ be a finite set and let $\xi: \Sigma \to \{1, \ldots, r\}$ be defined by $\xi(a_i) = i \ \forall_{i=1,\ldots,r}$. Then for any function $L: \mathbb{N}_0 \to \mathbb{N}_0$ a class $G_\Sigma^+(L(n))$ is defined by:

i) p, s, c_o, i are in $G_\Sigma^+(L(n))$.

(p, s, c_o, i are defined as in the definition of F_Σ)

ii) If $f,g \in G_\Sigma^+(L(n))$ then $\lambda x,w\,[f(g(x,w),w)]$ is in $G_\Sigma^+(L(n))$.

iii) Let $k \in \mathbb{N}$ and let $\alpha, \beta_1, \ldots, \beta_{k-1}, \gamma_1, \ldots, \gamma_{k-1}$ be total functions belonging to $G_\Sigma^+(L(n))$. Let f be defined by:

$$f(x,w) = \begin{cases} \alpha(x,w), & \text{if } x \equiv 0 \bmod k \\ f(\beta_j(f(\gamma_j(x,w),w),w),w), & \text{if } x \equiv j \bmod k,\ 1 \leq j \leq k-1 \end{cases}$$

Let $IP_{x,w}$ be the sets of intermediate points which belong to this scheme. Then f is an element of $G_\Sigma^+(L(n))$ if there exist $c,d \in \mathbb{N}$ such that for all $x \in \mathbb{N}_o$, $w \in \Sigma^*$ the following holds:

$y \leq c \cdot \max\{x, d \cdot L(l(w))\}$ for all $y \in IP_{x,w}$.

iv) If G is a class of functions such that i), ii), iii) hold, then $G_\Sigma^+(L(n)) \subset G$.

Theorem 2: If $L: \mathbb{N}_o \longrightarrow \mathbb{N}_o$ is a function such that $L(n) \geq n$ holds for all $n \in \mathbb{N}$, then $G_\Sigma^+(L(n)) = P_\Sigma^+(L(n))$.

The relation $G_\Sigma^+(L(n)) \subset P_\Sigma^+(L(n))$ is proved by applying the usual methods by means of which pushdown automata compute functions which are defined by recursive schemes.

To prove $P_\Sigma^+(L(n)) \subset G_\Sigma^+(L(n))$ we associate (this was done first in [1]) with each CPDA M (S - state set, Γ - pushdown alphabet) a function E_M which maps $\Gamma \times S \times \mathbb{N}_o \times \Sigma^*$ into $S \times \mathbb{N}_o$ in the following way:
Let $s \in S$, $\gamma \in \Gamma$, $x \in \mathbb{N}_o$, $w \in \Sigma^*$ be arbitrarily chosen. Let M start with the head position x, with the internal state s, with γ on its pushdown tape and with w on its input tape. If M stops with an empty pushdown tape and with internal state s' and head position x', then $E_M(\gamma, s, x, w) = (s', x')$. Otherwise $E_M(\gamma, s, x, w)$ is undefined.

E_M can be defined by a recursive scheme (this was already used in [4] and [5]) and this leads to the proof of Theorem 2.

5. Relationships between the complexity classes of the RAM and the classes which are defined by CPDAs with bounded counter length

The classes $G_\Sigma^+(L(n))$ are defined by using only a very restricted form of conditional expressions. If we drop this restriction we get another hierarchie of partial recursive functions.

Let $\Sigma = \{a_1,\ldots,a_r\}$ be a finite set and let $\xi : \Sigma \longrightarrow \{1,\ldots,r\}$ be defined by $\xi(a_i) = i \quad \forall_{i=1,\ldots,r}$. Then for any function $L: \mathbb{N}_0 \longrightarrow \mathbb{N}_0$ a class $H_\Sigma^+(L(n))$ is defined in the following way:

i) p, s, c_0, i are in $H_\Sigma^+(L(n))$.

ii) If $f,g \in H_\Sigma^+(L(n))$ then $\lambda\, x,w\,[f(g(x,w),w)]$ is in $H_\Sigma^+(L(n))$.

iii) Let $k \in \mathbb{N}$ and let f be defined by the following conditional expression (Mc Carthy's notation):

$f(x,w) = (p_1(x,w) \rightarrow f_1(x,w), p_2(x,w) \rightarrow f_2(x,w), \ldots, p_k(x,w) \rightarrow f_k(x,w))$

where:

1. $f_j(x,w) = \varphi_j(\alpha_j(\chi_j(\beta_j(x,w),w),w),w)$
   α_j, β_j are total functions out of $H_\Sigma^+(L(n))$ and φ_j, χ_j are total functions out of $H_\Sigma^+(L(n))$ or $\varphi_j = f$ or $\chi_j = f$.

2. $p_j(x,w)$ can be composed out of expressions of the form $\bar{\varphi}_j(\bar{\alpha}_j(\bar{\chi}_j(\bar{\beta}_j(x,w),w),w),w) = 0$ by means of $\neg$, $\wedge$, $\vee$.
   (where $\bar{\varphi}_j, \bar{\alpha}_j, \bar{\chi}_j, \bar{\beta}_j$ have the same properties as the functions $\varphi_j, \alpha_j, \chi_j, \beta_j$ of 1.)

Let $IP_{x,w}$ be the sets of intermediate points which belong to this scheme. Then f is an element of $H_\Sigma^+(L(n))$ if there exist $c,d \in \mathbb{N}$ such that for all $x \in \mathbb{N}_0$, $w \in \Sigma^*$ the following holds:
$y \leq c \cdot \max\{x, d \cdot L(l(w))\}$ for all $y \in IP_{x,w}$.

iv) If G is a class of functions such that i), ii), iii) hold, then $H_\Sigma^+(L(n)) \subset G$.

The classes $H_\Sigma^+(L(n))$ are defined by means of more general conditional expressions than the classes $G_\Sigma^+(L(n))$ and therefore it is obvious that for each func-

tion L the relation $G_\Sigma^+(L(n)) \subset H_\Sigma^+(L(n))$ holds.

Let M be a CPDA and let us assume that M always starts with its head on the leftmost cell of the input tape. Then M defines in a natural way a function $\bar{f}_M: \Sigma^* \longrightarrow \mathbb{N}_0$. So $\bar{f}_M(w) = f_M(0,w)$ where f_M is the function associated to M in section 4. This leads to the following definition:

$$P_\Sigma(L(n)) = \left\{ f: \Sigma^* \longrightarrow \mathbb{N}_0 \mid \exists\, \hat{f} \in P_\Sigma^+(L(n)) \text{ such that } f(w) = \hat{f}(0,w)\ \forall w \in \Sigma^* \right\}$$

$$P(L(n)) = \bigcup_\Sigma P_\Sigma(L(n))$$

Similary we define G(L(n)) and H(L(n)).

Theorem 3: $P(f(n)) = G(f(n)) \subset H(f(n)) \subset F(f(n)) = C_R(f(n))$ holds for any RAM-honest function $f: \mathbb{N}_0 \longrightarrow \mathbb{N}_0$

Theorem 4: $H(f(n)) \subset G(f(n)\cdot\log f(n))$ and $F(f(n)) \subset H(f(n)\cdot\log f(n))$ hold for any CPDA-honest function $f: \mathbb{N}_0 \longrightarrow \mathbb{N}_0$

In theorem 3 only the relation $H(L(n)) \subset F(L(n))$ has to be proved and this proof goes along the lines of the proof of the relation $P(f(n)) \subset C_R(f(n))$ which was given for slightly different machine models in [4] and [5].

To prove theorem 4 we have to describe the maximum operator by means of a recursive scheme. Let $f,g: \mathbb{N}_0 \times \Sigma^* \longrightarrow \mathbb{N}_0$ be arbitrary functions. We define a function $V_{f,g}: \mathbb{N}_0 \times \mathbb{N}_0 \times \Sigma^* \longrightarrow \mathbb{N}_0$ by:

$$V_{f,g}(x,i,w) = \max\left\{ x' \mid x' = 0 \vee x' \leq x \wedge \begin{array}{l}\text{the first i bits of } f(x',w) \\ \text{and } g(x,w) \text{ are equal}\end{array} \right\}$$

Then we have $[f^{-1}g](x,w) = V_{f,g}(x,\log_2 x,w)$. The function $V_{f,g}$ can be defined by a recursive scheme with conditional expressions of the general type and this leads to the relation $F(f(n)) \subset H(f(n)\cdot\log f(n))$.

On the other hand each scheme out of $H_\Sigma^+(f(n))$ can be simulated by a CPDA with counter length $f(n)\cdot\log f(n)$ (the factor $\log f(n)$ is needed to write the head position in binary notation on the pushdown tape without destroying the head position) and this completes the proof.

As a corollary we get: $P(f(n)) \subset C_R(f(n))$ and $C_R(f(n)) \subset P(f(n)\cdot(\log f(n))^2)$

6. More general random access machines

So far we considered only a very simple model of a random access machine (x+1 is the only arithmetical operation). We can show that the efficiency of a RAM is not altered if we allow additional arithmetical operations or additional predicates which belong to the class F_Σ (for example the additional instructions $x_i \leftarrow x_j * 1$, TRA m if $x_i = x_j$ do not increase the speed of a RAM by more than a constant factor).

Our methods can be extended to include random access machines with more general arithmetical operations of the form $x_i \leftarrow \gamma(x_j)$ provided that the maximum word length is bounded by $c \cdot T(n)$.

Furthermore (in cooperation with R. Weicker and G. Fleischhauer) we compared the efficiency of the RAM with other machine models:

a. Let $C_R^k(T(n))$ be the complexity classes which are defined by a machine which has random access to a k-dimensional storage. Then we have:

$$C_R^k(T(n)) = C_R^2(T(n)) \qquad \forall k \geq 2$$
$$C_R^2(T(n)) \subset C_R(T(n) \cdot \log T(n))$$

b. Let $C_A(T(n))$ be the complexity classes which are defined by a Turing machine with associative access to the memory (see R. Weicker, [9]). Then we have:

$$C_A(T(n)) \subset C_R(T(n) \cdot \log T(n))$$
$$C_R(T(n)) \subset C_A(T(n)^{1+\varepsilon}) \text{ for each } \varepsilon > 0$$

Acknowledgement:

I want to thank Mr. R. Weicker and Mr. G. Fleischhauer for several helpful discussions.

References:

1. Aho, A.V., Ullman, J.D., Hopcroft, J.E., On the computational power of pushdown automata, J.Comp.Syst.Sci. 4 (1970), 129-136

2. Cobham, A., The intrinsic computational complexity of functions, Proc.1964 Int.Congr.Logic, Method. and Ph. of Sci., North Holland 1965, 24-30

3. Cook, S.A., Characterizations of pushdown machines in terms of time-bounded computers, J.Assoc.Comput.Mach. 18 (1971), 4-18

4. Cook, S.A., Linear time simulation of deterministic two-way pushdown automata, Information Processing 71, vol.1, Amsterdam - London 1972, 75-80

5. Monien, B., Relationships between pushdown automata with counters and complexity classes, to appear in Math.Syst.Th.

6. Monien, B., On the simulation of time-bounded machines, 1. Fachtagung "Automatentheorie und Formale Sprachen", Proceedings 1973, Springer Verlag

7. Monien, B., Komplexitätsklassen von Automatenmodellen und beschränkte Rekursion, Bericht Nr. 8, Institut für Informatik, Universität Hamburg, Feb. 1974

8. Thompson, D.B., Subrecursiveness: Machine-independent notions of computability in restricted time and storage, Math.Syst.Th. 6 (1972), 3-15

9. Weicker, R., Turing machines with associative memory access, these proceedings