Computing 15, 217—234 (1975)

On Primitive Recursive Wordfunctions

F. W. v. Henke, K. Indermark, G. Rose*, and **K. Weihrauch**, Bonn

Received October 21, 1974

Abstract — Zusammenfassung

On Primitive Recursive Wordfunctions. In order to compare primitive recursive functions and transductions defined by automata in a natural way independent of encodings, we generalize the Grzegorczyk hierarchy, the recursion number hierarchy and the loop hierarchy from arithmetical to wordfunctions. We observe several differences between the arithmetical and the non-arithmetical theory. By means of Turingmachines and generalized sequential machines all inclusion problems for the function classes of these hierarchies are solved. Transductions and languages defined by automata are classified within these hierarchies. Moreover, we introduce and study primitive recursive transformations between different monoids.

Über primitiv-rekursive Wortfunktionen. Um einen natürlichen, von Kodierungen unabhängigen Vergleich zwischen primitiv-rekursiven Funktionen und Automatentransduktionen zu ermöglichen, werden die Grzegorczyk-Hierarchie, die Rekursionszahl-Hierarchie und die Loop-Hierarchie von arithmetischen auf Wortfunktionen verallgemeinert. Dabei ergeben sich einige Unterschiede zum arithmetischen Fall. Unter Benutzung von Turingmaschinen und verallgemeinerten endlichen Automaten werden alle Inklusionsprobleme der Funktionenklassen dieser Hierarchien gelöst. Von Automaten definierte Funktions- und Sprachklassen werden innerhalb dieser Hierarchien klassifiziert. Außerdem werden primitiv-rekursive Transformationen zwischen verschiedenen Monoiden behandelt.

Introduction

There exist essentially two methods of defining computable functions: by machines operating on symbol strings or by closing a set of functions under certain operations. We focus our attention here on Turingmachines and μ-recursiveness as representatives. In spite of their equivalence with respect to the whole class of computable functions, these methods allow different ways of studying the subrecursive structure of these classes of functions. Machines can be restricted in their use of storage during the computation (pushdown automata, sequential machines), and in the second case one can delimit the application of the closure operations. These restrictions leed to hierarchies which give some information

* This research was conducted at the Institut für Theorie der Automaten und Schaltnetzwerke (GMD) while the third author stayed there for one year. His present address: Case Western Reserve University, Cleveland, Ohio. Support was given by the Gesellschaft für Mathematik und Datenverarbeitung mbH, Bonn.

on the complexity of computable functions. The question whether this sort of complexity depends on the definition or not leads us to a comparison of the different hierarchies.

But as μ-recursiveness is based on positive integers and Turingmachines on symbol strings we have to compare classes of arithmetical functions with classes of wordfunctions. In the past this difficulty has been dealt with by translating wordfunctions into arithmetical functions [11], [14], [15]. So, the results depend on this translation, often called encoding.

One aim of this paper is to remove that dependency. Instead of restricting the study to arithmetical functions we extend the theory of μ-recursive functions along the lines developed by Asser [1], Eilenberg and Elgot [4]. The essential idea is to regard finitely generated, free monoids as monadic algebras where we have several successor functions instead of only one as in the arithmetical case. Although most theorems carry over directly to the general case there are some points where we have a difference between the arithmetical and the non-arithmetical theory. The reason lies in the fact that the use ot two symbols allows alternatives within the recursion scheme. So, in the non-arithmetical case, some results were strengthened, some problems could be solved.

In the first part of this paper we generalize primitive recursive hierarchies: the Grzegorczyk hierarchy [16], the recursion number hierarchy [2], [6] and the loop hierarchy [12], [18]. For the corresponding classes of functions all inclusion problems are solved. As a main tool we use a description of Turing machines by primitive recursive wordfunctions that shows a close connection between complexity classes of Turing machines and those hierarchies. Of particular interest is a simple characterization of the first Grzegorczyk class in the non arithmetical case. Furthermore we need a special construction of GSM's (generalized sequential machines) in order to characterize the second loop class and the second recursion number class.

In the second part we introduce primitive recursive transformations from one monoid to another with the intention of classifying encodings. Two different definitions turn out to be equivalent. Moreover we define Grzegorczyk classes of transformations and study the relation between transformation and function classes, in particular closure properties with respect to composition and the influence of encodings.

In the last part we give some applications. Transductions and languages defined by automata are classified within primitive recursive hierarchies.

This is the final version of a series of reports on this subject. Detailed proofs can be found in [7], [8], [9], [10], [17], [20], [21], [22].

Notation

$\mathbb{N}$ denotes the set of non-negative integers and Σ^* the free monoid generated by an alphabet $\Sigma=\{a_1, \ldots, a_r\}$ with e as the empty word. For $w \in \Sigma^*$ $|w|$ denotes the length of w. In most cases we write $\tilde{x}$ for $(x_1, \ldots, x_k)$ and $|\tilde{x}|$ for

$(|x_1|, \ldots, x_k|)$. $F^{(k)}(\Sigma)$ is the set of all functions $f: (\Sigma^*)^k \to \Sigma^*$ and

$$F(\Sigma) := \bigcup_{k=0}^{\infty} F^{(k)}(\Sigma).$$

For each $K \subseteq F(\Sigma)$ we define $K^{(n)} := K \cap F^{(n)}(\Sigma)$.

The index Σ is dropped for $r=1$. For two sets A and B we write $A \subset B$ iff $A \subseteq B$ and $A \neq B$. A partial function f from A to B is denoted by $f: A \to B$.

Part 1: Hierarchies of Primitive Recursive Wordfunctions

1. *Basic Definitions*

Primitive recursive wordfunctions were introduced by Asser [1] and studied by Eilenberg and Elgot [4]. The concept of primitive recursiveness can be generalized from $\mathbb{N}$ to Σ^* using r successor functions instead of only one ($r := |\Sigma|$; $\mathbb{N} \cong \{a_1\}^*$). This means in particular that the primitive recursion scheme must contain a recursion equation for each successor function. Now, we have the choice of starting from right- or left-successors. Without affecting the results we choose left-successors for the base.

Definition 1: *Base Functions*
The set $B(\Sigma)$ of base functions is the subset of $F(\Sigma)$ that consists exactly of the *left-successors* λ_i $(i=1, \ldots, r)$, the *zerofunctions* $w^{(0)}$ and $w^{(1)}$ and the *projections* $\pi_l^{(k)}$ $(k \geq 1,\ 1 \leq l \leq k)$. They are defined by $\lambda_i(x) = a_i x$, $w^{(0)} = e$, $w^{(1)}(x) = e$ and $\pi_l^{(k)}(x_1, \ldots, x_k) = x_l$.

Definition 2: *Operations*

a) *Composition*: For $f \in F^{(k)}(\Sigma)$ and $g_1, \ldots, g_k \in F^{(l)}(\Sigma)$ the composition $h \in F^{(l)}(\Sigma)$ is defined by $h(x) = f(g_1(x), \ldots, g_k(x))$. Notation: $h = f \circ (g_1, \ldots, g_k)$.

b) *Primitive recursion*: From $f \in F^{(k)}(\Sigma)$ and $g_1, \ldots, g_r \in F^{(k+2)}(\Sigma)$ the function $h \in F^{(k+1)}(\Sigma)$ is formed by primitive recursion iff

$$h(\tilde{x}, e) = f(\tilde{x})$$
$$h(\tilde{x}, a_i y) = g_i(\tilde{x}, y, h(\tilde{x}, y)) \qquad (i=1, \ldots, r).$$

Notation: $h = \mathfrak{P}(f, (g_i))$.

c) *Limited recursion*: If in addition to $h = \mathfrak{P}(f, (g_i))$ as in b) there is $d \in F^{(k+1)}(\Sigma)$ limiting h by $|h(\tilde{z})| \leq |d(\tilde{z})|$ for all $\tilde{z} \in (\Sigma^*)^{k+1}$, we say: h is formed by limited recursion from f and $g_1, \ldots, g_r$ resp. d. Notation $h = \mathfrak{L}(f, (g_i); d)$.

d) *Simultaneous recursion*: From $f_i \in F^{(k)}(\Sigma)$ and $g_{i1}, \ldots, g_{ir} \in F^{(k+l+1)}(\Sigma)$ the functions $h_i \in F^{(k+1)}(\Sigma)$ $(i=1, \ldots, l)$ are formed by simultaneous recursion iff

$$h_i(\tilde{x}, e) = f_i(\tilde{x}) \qquad (1 \leq i \leq l)$$
$$h_i(\tilde{x}, a_j y) = g_{ij}(\tilde{x}, y, h_1(\tilde{x}, y), \ldots, h_l(\tilde{x}, y)) \qquad (1 \leq j \leq r)$$

Notation: $h_m = \mathfrak{S}_m((f_i), (g_{ij}))$ for $1 \leq m \leq l$.

e) *Limited simultaneous recursion*: If in d) the functions h_i are limited by $d_i \in F^{(k+1)}(\Sigma)$, i.e. $|h_i(\tilde{z})| \leq |d_i(\tilde{z})|$ for all $\tilde{z} \in (\Sigma^*)^{k+1}$ $(1 \leq i \leq l)$, we say: the

functions h_i are formed by limited simultaneous recursion from f_i and g_{ij} resp. d_i. Notation: $h_m = \mathfrak{LS}_m((f_i), (g_{ij}); (d_i)) \quad (1 \leq m \leq l)$.

By repeated application of these operations to the base functions we construct the class of primitive recursive functions over Σ together with three hierarchies.

Definition 3: *Primitive Recursive Wordfunctions*
The class $Pr(\Sigma)$ of primitive recursive wordfunctions (over Σ) is the smallest class $K \subseteq F(\Sigma)$ that contains $B(\Sigma)$ and that is closed under composition and primitive recursion.

Definition 4: *Recursion Number Hierarchy*
The recursion number classes $R_n(\Sigma)$ $(n \geq 0)$ are defined by induction:

(i) $R_0(\Sigma)$ is the smallest class $K \subseteq F(\Sigma)$ that contains $B(\Sigma)$ and that is closed under composition.

(ii) $R_{n+1}(\Sigma)$ is the smallest class $K \subseteq F(\Sigma)$ that contains $R_n(\Sigma)$ and that is closed under composition and under primitive recursion over $R_n(\Sigma)$, i.e. from f, $(g_i) \in R_n(\Sigma)$ and $h = \mathfrak{P}(f, (g_i))$ we have $h \in R_{n+1}(\Sigma)$.

Definition 5: *Loop Hierarchy*
If we replace "primitive recursion" by "simultaneous recursion" in Definition 4 we get the loop classes $L_n(\Sigma)$ $(n \geq 0)$.

In order to define the Grzegorczyk hierarchy we need a sequence of "growth functions". We generalize Ritchie's definition [16] which is slightly different from the original one in Grzegorczyk [5].

Definition 6: *Generalized Ackermann Functions*

In $Pr^{(2)}(\Sigma)$ we define the sequence $(A_n^\Sigma)_{n \in \mathbb{N}}$ by $A_0^\Sigma(x, y) := a_1\, y$, $A_1^\Sigma(x, e) := x$, $A_2^\Sigma(x, e) := e$, $A_n^\Sigma(x, e) := a_1$ $(n \geq 3)$, $A_{n+1}^\Sigma(x, a_i\, y) := A_n^\Sigma(x, A_{n+1}^\Sigma(x, y))$ $(n \geq 0, i = 1, \ldots, r)$.

Definition 7: *Grzegorczyk Hierarchy*

For each $n \in \mathbb{N}$ the Grzegorczyk class $E_n(\Sigma)$ is the smallest subclass of $Pr(\Sigma)$ that contains $B(\Sigma)$ together with A_n^Σ and that is closed under composition and limited recursion.

These hierarchies are generalizations from the corresponding notions in the arithmetical case: see Heinermann [6] and Axt [2] for the recursion number hierarchy, Meyer [12] for the loop hierarchy which turns out to be equivalent to Schwichtenberg's definition [18] based on simultaneous recursion, and Ritchie [16] for the Grzegorczyk hierarchy. Grzegorczyk classes are defined as growth classes: $E_0(\Sigma)$ contains only functions with a growth of $|x_i| + k$, $E_1(\Sigma)$ functions with linear growth, $E_2(\Sigma)$ functions with polynomial growth, $E_3(\Sigma)$ functions with exponential growth, and so on. In general, primitive recursion produces functions with much more growth; limited recursion restricts this growth. As we used the length of words in Σ^* in the definition of limited recursion we have strong analogies between arithmetical and non-arithmetical Grzegorczyk classes.

2. *Elementary Inclusion Properties*

In the next three sections we solve the following inclusion problem: is $A \subseteq B$ for $A, B \in \{R_n(\Sigma), L_n(\Sigma), E_n(\Sigma) \mid n \in \mathbb{N}\}$?

The solution shows in particular that these classes form three hierarchies which are nearly identical.

From the definitions we conclude directly the following properties:

$$R_n(\Sigma) \subseteq R_{n+1}(\Sigma),\ L_n(\Sigma) \subseteq L_{n+1}(\Sigma),\ R_n(\Sigma) \subseteq L_n(\Sigma),\ R_0(\Sigma) = L_0(\Sigma),$$
$$\bigcup_{n=0}^{\infty} R_n(\Sigma) = Pr(\Sigma) \text{ and } \bigcup_{n=0}^{\infty} E_n(\Sigma) \subseteq Pr(\Sigma).\ (n \in \mathbb{N}) \tag{1}$$

There is a close relationship between arithmetical and non-arithmetical functions that is demonstrated by Lemma 1. It is very useful in generalizing results from the arithmetical case to the non-arithmetical one.

Lemma 1. Each $K \in \{R_n, L_n, E_n \mid n \geq 0\}$ has the following properties:

(i) For each $f \in K$ there is $f^{\Sigma} \in K(\Sigma)$ such that $f(|\tilde{x}|) = |f^{\Sigma}(\tilde{x})|$.

(ii) For each $f^{\Sigma} \in K(\Sigma)$ there is $f \in K$ such that $|f^{\Sigma}(\tilde{x})| \leq f(|\tilde{x}|)$.

The proof is straightforward by induction on the structure of K resp. $K(\Sigma)$. As an immediate consequence we have Theorem 1 which enables us to show the hierarchical structure of the Grzegorczyk classes.

Theorem 1. *Let f^{Σ} be defined by primitive recursion over $E_n(\Sigma)$. Then $f^{\Sigma} \in E_n^{(k)}(\Sigma)$ iff there is $d \in E_n^{(k)}$ with $|f^{\Sigma}(\tilde{x})| \leq d(|\tilde{x}|)$ for all $\tilde{x} \in (\Sigma^*)^k$.*

For the arithmetical Ackermann functions we know that

$$A_i(x, y) \leq A_i(x+2, y+2) \leq A_{i+1}(x+2, y+2)$$

and furthermore that A_{n+1} grows more than each function of $E_n^{(2)}$. By Lemma 1 and Theorem 1, this can be generalized to wordfunctions:

Lemma 2. *Let $n \in \mathbb{N}$. For all $k \leq n$ we have:*

$$A_k^{\Sigma} \in E_n(\Sigma) \quad \text{and} \quad A_{n+1}^{\Sigma} \notin E_n(\Sigma).$$

This result proves that the Grzegorczyk classes form a hierarchy:

$$E_n(\Sigma) \subset E_{n+1}(\Sigma) \quad (n \in \mathbb{N}). \tag{2}$$

Although $E_0(\Sigma)$ contains only functions which grow very slowly we can show that every recursively enumerable subset of Σ^* can be enumerated with a function of $E_0^{(1)}(\Sigma)$. The "conditional definitions" are other examples of functions in $E_0(\Sigma)$: for $f \in F^{(k)}(\Sigma)$ which is constant almost everywhere we have $f \in E_0(\Sigma) \cap R_1(\Sigma)$.

Our next aim is to compare loop classes with Grzegorczyk classes. Using encoding and decoding of word tupels we reduce simultaneous recursion to simple recursion. Essentially, we require the following theorem on pairing functions which demon-

strates in particular the difference between arithmetical and non-arithmetical function classes.

Theorem 2. *There exist functions* $\eta : (\Sigma^*)^2 \to \Sigma^*$ *and* $\eta_1, \eta_2 : \Sigma^* \to \Sigma^*$ *which have the following properties:*

(*i*) $\eta_i \circ \eta = \pi_i^{(2)} \quad (i = 1, 2)$

(*ii*) *If for* $i = 1, 2 \mid x_i \mid \leq \mid y_i \mid$, *then* $\mid \eta(x_1, x_2) \mid \leq r \cdot \mid \eta(y_1, y_2) \mid$.

(*iii*) *For* $r = 1$ *there is* $\eta \in R_2 \cap E_2$ *and* $\eta_1, \eta_2 \in R_3 \cap E_0$,
for $r > 1$ *there is* $\eta \in R_1(\Sigma) \cap E_1(\Sigma)$ *and* $\eta_1, \eta_2 \in R_2(\Sigma) \cap E_0(\Sigma)$.

Moreover, it can be shown that for $r = 1$ there are no functions η, η_1, η_2 such that (i) and $\eta \in R_1 \cup E_1$! The following functions satisfy Theorem 2:

$$\eta(x, y) = (x + y)^2 + y \quad (r = 1)$$

and

$$\eta(a_{i_1} \dots a_{i_k}, a_{j_1} \dots a_{j_l}) = a_1 a_2^{i_1} a_1 \dots a_1 a_2^{i_k} a_1 a_1 a_2^{j_1} a_1 \dots a_1 a_2^{j_l} a_1 \quad (r > 1).$$

But there does not exist any linear bounded injection $\eta : \mathbb{N}^2 \to \mathbb{N}$. Now, we can prove the following theorem:

Theorem 3. E_n *is closed under limited simultaneous recursion for* $n \geq 2$. *But* $E_n(\Sigma)$ *with* $\mid \Sigma \mid \geq 2$ *has this property even for* $n \geq 1$.

So, we can show some relations between loop classes and Grzegorczyk classes:

$$L_0(\Sigma) \subseteq E_0(\Sigma), \; L_1(\Sigma) \subseteq E_1(\Sigma), \; L_n(\Sigma) \subseteq E_{n+1}(\Sigma) \quad (n \geq 2) \tag{3}$$

As a corollary we have:

$$\bigcup_{n=0}^{\infty} R_n(\Sigma) = \bigcup_{n=0}^{\infty} L_n(\Sigma) = \bigcup_{n=0}^{\infty} E_n(\Sigma) = Pr(\Sigma) \tag{4}$$

In the next section we develop some theorems on Turing machines that give more insight into these relationships.

3. *Inclusion Properties Proved by Turing Machines*

The arithmetical Grzegorczyk classes form complexity classes. This fact follows immediately from the Union Theorem of McCreight and Meyer for Shepherdson-Sturgis machines [18] and for Turing machines [3]. In the latter case we see again some dependence on encodings. This difficulty is solved here by comparing functions over the same alphabet. It can be shown that the transition function of a Turing machine is describable by very simple primitive recursive wordfunctions.

The class $\mathfrak{M}(\Sigma)$ of turing machines we have in mind is specified in the following way: $M \in \mathfrak{M}(\Sigma)$ is deterministic and has n input tapes, m storage tapes and one output tape. M starts in its initial state with empty output and storage tapes reading with each input head the blank b on the left to each input word ($b \notin \Sigma$). At each step M prints from left to right a symbol, possibly b, on the output tape. Occasionally, M stops in its final state with that word as output word which is obtained from the output tape by omitting the blanks. Without loss of generality

we assume furthermore that all tapes, except the output tape, are always of the form ... $bb\,w\,bb$ ... ($w \in \Sigma^*$), w called "inscription", and that the heads do not exceed the first blank to the left and to the right of w. Finally, it is convenient to change this model such that M does not really stop in its final state but continues working without affecting the output. This can be achieved by a loop not leaving the final state and printing blanks.

We shall now describe the computation process of M by some wordfunctions. For simplicity we assume $n=m=1$. Furthermore, we take the set S of states, the set $D=\{\text{left, right, not}\}$ of moves and $\Sigma_0=\Sigma \cup \{b\}$ as disjoint subsets of Σ^* such that b becomes e and $a \in \Sigma$ remains unchanged.

a) The transition function of M $g_M : (\Sigma^*)^8 \to (\Sigma^*)^8$ determines for each instantaneous description of M the successor description that is achieved in one computation step. An instantaneous description $\tilde{x}=(x_1, \ldots, x_8)$ has the following components: x_1 is the reverse of the word on the left to the input head, x_2 the symbol below the input head, x_3 the word on the right to the input head, x_4, x_5 and x_6 give the corresponding parts of the storage tape, x_7 is the word on the output tape and x_8 is the state.

b) The function $0_M : (\Sigma^*)^2 \to \Sigma^*$ is defined such that $0_M(x, y)$ is the inscription of the output tape after $|y|$ steps of M beginning with input x.

c) The step counting function $T_M : \Sigma^* \to \Sigma^*$ has the following property: if M reaches its final state on input x in k steps then $T_M(x)=a_1^k$; $T_M(x)$ is undefined iff M does not reach a final state on x.

d) The function computed by M $f_M : \Sigma^* \to \Sigma^*$ gives for each input x the output word that is on the output tape when M enters the final state, else undefined. Thus: $f_M(x)=0_M(x, T_M(x))$.

Theorem 4. *Let* $M \in \mathfrak{M}(\Sigma)$ *and* $t:=3n+3m+2$. *The transition function* $g_M : (\Sigma^*)^t \to (\Sigma^*)^t$ *can be extended to* $\tilde{g} : (\Sigma^*)^t \to (\Sigma^*)^t$ *such that* $\tilde{g}_i := \pi_i^{(t)} \circ \tilde{g}$ *is in* $R_1(\Sigma) \cap E_1(\Sigma)$ $(i=1, \ldots, t)$.

Proof: Again we assume $n=m=1$. M can then be given by a function

$$\delta : S \times \Sigma_0 \times \Sigma_0 \to S \times \Sigma_0 \times \Sigma_0 \times D \times D \cdot \delta(s, c_1, c_2)=(s', c_2', c_3, d_1, d_2)$$

means: reading in state s c_1 on the input tape and c_2 on the storage tape, M changes into state s', printing c_2' on the storage and c_3 on the output tape and moving input resp. storage head by d_1 resp. d_2. With the embedding from above we have $\delta : (\Sigma^*)^3 \dashrightarrow (\Sigma^*)^5$ which we extend to $\Delta : (\Sigma^*)^3 \to (\Sigma^*)^5$ by defining $\Delta(\tilde{x}) := (e, e, e, e, e)$ if $\tilde{x} \notin \text{domain}(\delta)$. It can be shown easily that $\Delta_i := \pi_i^{(5)} \circ \Delta$ is in $R_1(\Sigma) \cap E_0(\Sigma)$ $(i=1, \ldots, 5)$, see the remark following property (2). As auxiliary functions we need the concatenation of words $cn : (\Sigma^*)^2 \to \Sigma^*$, the function $df : \Sigma^* \to \Sigma^*$ that deletes the first symbol of a word and the function $fs : \Sigma^* \to \Sigma^*$ that gives the first symbol of a word, additionally $df(e)=fs(e)=e$. They are all in $R_1(\Sigma) \cap E_1(\Sigma)$.

Now, we can describe the desired function $\tilde{g}=(\tilde{g}_1, \ldots, \tilde{g}_8)$ over $\Delta_1, \ldots, \Delta_5$, cn and df by definition of cases. Let $\tilde{x}=(x_1, \ldots, x_8)$ and $\tilde{y}=(x_8, x_2, x_5)$.

$$\tilde{g}_1(\tilde{x}) = \begin{cases} x_1 & \text{if } \Delta_4(\tilde{y}) = \text{not} \\ df(x_1) & \text{if } \Delta_4(\tilde{y}) = \text{left} \\ cn(x_2, x_1) & \text{if } \Delta_4(\tilde{y}) = \text{right} \\ x_1 & \text{otherwise} \end{cases}$$

$$\tilde{g}_2(\tilde{x}) = \begin{cases} x_2 & \text{if } \Delta_4(\tilde{y}) = \text{not} \\ fs(x_1) & \text{if } \Delta_4(\tilde{y}) = \text{left} \\ fs(x_3) & \text{if } \Delta_4(\tilde{y}) = \text{right} \\ x_2 & \text{otherwise} \end{cases}$$

$\tilde{g}_3$ is symmetrical to $\tilde{g}_1$; $\tilde{g}_4, \tilde{g}_5, \tilde{g}_6$ are defined in a similar way with Δ_5 instead of Δ_4 and by using Δ_2 in addition.

$$\tilde{g}_7(\tilde{x}) = cn(x_7, \Delta_3(\tilde{y}))$$
$$\tilde{g}_8(\tilde{x}) = \Delta_1(\tilde{y})$$

All functions used in this definition are in $R_1(\Sigma) \cap E_1(\Sigma)$. As this class is closed under definition by cases we have proved the theorem. Q.E.D.

Theorem 5. *For $M \in \mathfrak{M}(\Sigma)$ with n input tapes the function $0_M : (\Sigma^*)^{n+1} \to \Sigma^*$ is in $L_2 \cap E_2$ if $|\Sigma| = 1$ and in $L_2(\Sigma) \cap E_1(\Sigma)$ if $|\Sigma| > 1$.*

Proof: We define $G : (\Sigma^*)^{n+1} \to (\Sigma^*)^{n'}$ $(n' := 3(n+m)+2)$ by $G(\tilde{x}, e) = \alpha(\tilde{x})$, where $\alpha(\tilde{x})$ is the initial instantaneous description of M with input $\tilde{x}$, and $G(\tilde{x}, a_i y) = \tilde{g}(G(\tilde{x}, y))$. As we have $|\pi_i^{(n')} \circ \tilde{g}(z_1, \ldots, z_{n'})| \leq |z_i| + 1$ for all $i = 1, \ldots, n'$, G is defined by limited simultaneous recursion over $R_1(\Sigma) \cap E_1(\Sigma)$. The assertion follows now from Theorem 3 and $0_M = \pi_{n'-1}^{(n')} \circ G$.

Theorem 6. *Let $M \in \mathfrak{M}(\Sigma)$ such that $|T_M(\tilde{x})| \leq |t(\tilde{x})|$ for some $t : (\Sigma^*)^n \to \Sigma^*$. Then the following holds:*

(i) $t \in E_n(\Sigma) \Rightarrow f_M \in E_n(\Sigma)$ *(if $(r = 1$ and $n > 1)$ or $(r > 1$ and $n \geq 1)$)*
(ii) $t \in L_n(\Sigma) \Rightarrow f_M \in L_n(\Sigma)$ $(n \geq 2)$
(iii) $t \in R_n(\Sigma) \Rightarrow f_M \in R_n(\Sigma)$ $(n \geq 3)$

Proof: $f_M(\tilde{x}) = 0(\tilde{x}, t(\tilde{x}))$. See Schwichtenberg [18] for a proof of (iii) in the case $r = 1$.

Theorem 7. *For $f \in E_n(\Sigma)$ $(n \geq 2)$ there is a Turing machine $M \in \mathfrak{M}(\Sigma)$ and a $t \in E_n(\Sigma)$ such that $f_M = f$ and $|T_M(\tilde{x})| \leq |t(\tilde{x})|$ for all $\tilde{x} \in$ domain (f).*

This theorem can be proved by induction over the construction of f in $E_n(\Sigma)$. For the base functions there are simple machines, composition and primitive recursion can be simulated by combining the corresponding machines appropriately.

Theorem 6 (i) and Theorem 7 characterize Grzegorczyk classes $E_n(\Sigma)$ for $n \geq 2$ as complexity classes. If $|\Sigma| \geq 2$, we can characterize even $E_1(\Sigma)$ by Turing machines.

Theorem 8. *Let $|\Sigma| \geq 2$ and $f \in F(\Sigma)$. Then we have $f \in E_1(\Sigma)$ iff there is $M \in \mathfrak{M}(\Sigma)$ such that $f = f_M$ and $|T_M(\tilde{x})| \leq |t(\tilde{x})|$ for some $t \in E_2(\Sigma)$ and M is linear bounded*

on all tapes by the input length (i.e. M computes f in polynomial time and on linear bounded tapes).

As we do not need this theorem to prove any inclusion properties we omit the proof.

These results enable us to show that the three hierarchies are nearly identical:

$$L_2(\Sigma)=E_3(\Sigma),\ R_n(\Sigma)=L_n(\Sigma)=E_{n+1}(\Sigma)\quad (n\geq 3). \tag{5}$$

Proof: See Schwichtenberg [18] for $|\Sigma|=1$. Let $|\Sigma|\geq 2$ and $f\in E_3(\Sigma)$. By Theorem 7 there is $M\in\mathfrak{M}(\Sigma)$ with $f_M=f$ and $|T_M(\tilde{x})|\leq|t(\tilde{x})|$ for some $t\in E_3(\Sigma)$. By Lemma 1 we have $t'\in E_3$ with $|t(\tilde{x})|\leq t'(|\tilde{x}|)$.

Now, we can apply a lemma of Schwichtenberg by which each $g\in E_{n+1}$ can be bounded by some $g'\in R_n$ $(n\geq 2)$. So, together with Lemma 1, we conclude: $|T_M(\tilde{x})|\leq|t''(\tilde{x})|$ for some $t''\in R_2(\Sigma)$. As $R_2(\Sigma)\subseteq L_2(\Sigma)$ we see from Theorem 6 that $f\in L_2(\Sigma)$. In just the same way we prove that $E_{n+1}(\Sigma)\subseteq R_n(\Sigma)$ for $n\geq 3$. We conclude this section with the following result:

$$\begin{gathered}\text{For } |\Sigma|\geq 2 \text{ we have } R_2(\Sigma)=L_2(\Sigma).\\ \text{(Note added in proof: } R_2=L_2\text{, shown by H. Müller, Münster.)}\end{gathered} \tag{6}$$

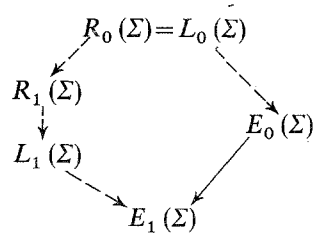
Proof (Sketch): We have only to show that $E_3(\Sigma)\subseteq R_2(\Sigma)$. This problem can be reduced to the problem of proving that $E_3^{(1)}(\Sigma)\subseteq R_2^{(1)}(\Sigma)$.
For $f\in E_3^{(1)}(\Sigma)$ there is a Post machine P over Σ together with functions $c, d, t\in R_2(\Sigma)$ such that $f=d\circ f_P\circ c$ and $|T_P(c(x))|\leq|t(x)|$ for all $x\in\Sigma^*$. The assertion follows now from the fact that $0_P\in R_2(\Sigma)$.

4. *Inclusion Properties Proved by Generalized Sequential Machines*

In order to give a complete solution to the inclusion problem we still have to investigate the relations between the following classes:

$$R_0(\Sigma),\ R_1(\Sigma),\ L_0(\Sigma),\ L_1(\Sigma),\ E_0(\Sigma),\ E_1(\Sigma).$$

The diagram shows what we have proved already.



$A\to B$ stands for $A=B$, $A\dashrightarrow B$ for $A\subseteq B$.

Moreover, as the concatenation is in $R_1(\Sigma)$, but not in $R_0(\Sigma)$ and the Ackermann function A_1^x in $R_1(\Sigma)$, but not in $E_0(\Sigma)$, we have:

$$R_0(\Sigma)\subset R_1(\Sigma),\ R_1(\Sigma)\not\subseteq E_0(\Sigma),\ L_1(\Sigma)\not\subseteq E_0(\Sigma). \tag{7}$$

The remaining problems are solved by characterizations of $R_1(\Sigma)$ and $L_1(\Sigma)$. This is quite simple in the arithmetical case [19], for $|\Sigma| \geq 1$ we use Generalized Sequential Machines (*GSM* for short).

Definition 8. M is a Generalized Sequential Machine over Σ ($M \in GSM(\Sigma)$) iff $M = (K; q_0, \delta, \lambda, \omega)$ where K is a non-empty, finite set of states $q_0 \in K$ the initial state, $\delta : K \times \Sigma \to K$ the transition function, $\lambda : K \times \Sigma \to \Sigma^*$ the output function and $\omega : K \to \Sigma^*$ the last-output function. M computes a function $f_M : \Sigma^* \to \Sigma^*$ in the following way: let $\delta_M : \Sigma^* \to K$ be the function that gives the state of M after input w, i.e. $\delta_M(e) = q_0$, $\delta_M(wa_i) = \delta(\delta_M(w), a_i)$ $(i = 1, \ldots, r)$ and $f'_M : \Sigma^* \to \Sigma^*$ be the function that describes the output without recognizing the end of the input tape, i.e. $f'_M(e) = e$, $f'_M(wa_i) = f'_M(w)\,\lambda(\delta_M(w), a_i)$, then we can define f_M by $f_M(w) = f'_M(w)(\delta_M(w))$.

Thus, a *GSM* computes a one-place function. In order to compute many-place functions, too, we introduce *GSM*-products.

Definition 9. Let $n \geq 1$. $P^{(n)}$ is a *GSM*-product over Σ ($P^{(n)} \in GSMP(\Sigma)$) iff $P^{(n)} = (M_1, M_2, \ldots, M_q, is, ms)$ where $M_1, \ldots, M_q \in GSM(\Sigma)$ $(q \geq 1)$, $is : \{1, 2, \ldots, q\} \to \{1, 2, \ldots, n\}$ the input-selector function and $ms : K \to \{1, \ldots, q\}$ (K disjoint union of $K_1, \ldots, K_q$) the machine-selector function.

For each $d \geq 1$ $P^{(n)}$ computes a function $f_{P^{(n)},d} : (\Sigma^*)^n \to (\Sigma^*)$: let $\delta_{P^{(n)},d} : (\Sigma^*)^n \to K$ be the function that gives the state of $P^{(n)}$ after d *GSM*-computations, more exactly we define by induction

$$\delta_{P^{(n)},1}(w_1, \ldots, w_n) = \delta_{M_1}(w_{is(1)}),\ \delta_{P^{(n)},d+1}(w_1, \ldots, w_n) = \delta_{M_k}(w_{is(k)})$$

where

$$k = ms(\delta_{P^{(n)},d}(w_1, \ldots, w_n)),$$

then we get $f_{P^{(n)},d}$ by $f_{P^{(n)},1}(w_1, \ldots, w_n) = f_{M_1}(w_{is(1)})$,

$$f_{P^{(n)},d+1}(w_1, \ldots, w_n) = f_{P^{(n)},d}(w_1, \ldots, w_n)\ f_{M_k}(w_{is(k)}) \quad (k \text{ as above}).$$

Finally, we call $f : (\Sigma^*)^n \to \Sigma^*$ *GSMP*-computable iff there is $P^{(n)} \in GSMP(\Sigma)$ and $d \geq 1$ with $f = f_{P^{(n)},d}$.

Now we have a tool to characterize $L_1(\Sigma)$.

Theorem 9. *Let $f \in F(\Sigma)$. Then $f \in L_1(\Sigma)$ iff f is GSMP-computable or $f \in F^{(0)}(\Sigma)$.*

Theorem 10. *The range of an $f \in L_1(\Sigma)$ is context-sensitive.*

The last result can be shown as a consequence of Theorem 8 and the fact that each *GSMP*-computable $f : (\Sigma^*)^n \to \Sigma^*$ has the following property:

$$\bigvee_{k \in \mathbb{N}} \bigwedge_{z \in range(f)} \bigvee_{\tilde{x} \in (\Sigma^*)^n} z = f(\tilde{x}) \wedge |x_1 x_2 \ldots x_n| \leq k(|z| + 1).$$

As we know on the other hand that every recursively enumerable set is the range of an $f \in E_0(\Sigma)$, we have the corollary

$$E_0(\Sigma) \not\subseteq L_1(\Sigma),\ E_0(\Sigma) \not\subseteq R_1(\Sigma),\ R_0(\Sigma) \subset E_0(\Sigma) \text{ and } L_1(\Sigma) \subset E_1(\Sigma). \quad (8)$$

As an example we note that the reversal function is in $E_0(\Sigma)$, but not in $L_1(\Sigma)$.

Now, it remains to solve the problem whether $L_1(\Sigma) \subseteq R_1(\Sigma)$ or not. For the solution we characterize $R_1(\Sigma)$ by loopfree *GSM*'s.

Definition 10.

(i) Let $M=(K; q_0, \delta, \lambda, \omega) \in GSM(\Sigma)$.
M is called loopfree iff for all $p, q \in K$ and $x, y \in \Sigma^*$ we have

$$\delta_M(p, x)=q,\ \delta_M(q, y)=p \Rightarrow p=q.$$

(ii) Let $P^{(n)}=(M_1, \ldots, M_q, is, ms) \in GSMP(\Sigma)$.
$P^{(n)}$ is called loopfree iff M_i is loopfree for $i=1, \ldots, q$.

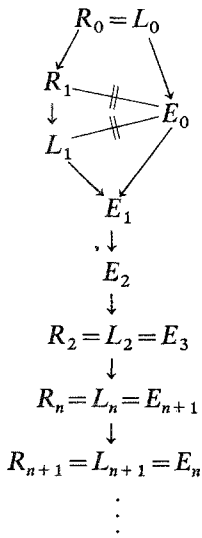
Theorem 11. *For $f \in F(\Sigma)$ we have:*
$f \in R_1(\Sigma)$ iff f is computable by a loopfree GSM-product or $f \in F^{(0)}(\Sigma)$.

$$L_1(\Sigma) \nsubseteq R_1(\Sigma) \tag{9}$$

For the proof we can show that $f: \Sigma^* \to \Sigma^*$ defined by

$$f(x)=\begin{cases} e & \text{if 2 divides } |x| \\ a_1 & \text{else} \end{cases} \qquad \text{is in } L_1(\Sigma), \text{ not in } R_1(\Sigma).$$

We summarize our results in the following diagram:



$(n \geq 3)$

$A \to B$ means $A \subset B$
$A \nrightarrow B$ means $A \nsubseteq B$ and $B \nsubseteq A$

Part 2: Primitive Recursive Transformations

The notion of a primitive recursive function between different monoids cannot be defined in the same way as for wordfunctions. We shall discuss two possibilities that prove to be equivalent.

Let $\Sigma=\{a_1, \ldots, a_r\}$ and $\Pi=\{b_1, \ldots, b_t\}$ be two, not necessarily disjoint alphabets. The generalization can be achieved by taking special functions for the transition from Σ^* to Π^*.

Definition 11.

(i) A function $c: \Sigma^* \to \Pi^*$ is called *encoding* $(c \in K(\Sigma, \Pi))$ iff for each $f \in Pr(\Sigma)$ there is $f' \in Pr(\Pi)$ such that $c \circ f = f' \circ c$.

(ii) The class $Pr_1(\Sigma, \Pi)$ of *primitive recursive transformations from* Σ^* *to* Π^* is defined by $Pr_1(\Sigma, \Pi) := K(\Sigma, \Pi) \circ Pr(\Sigma)$ $(= \{c \circ f \mid c \in K(\Sigma, \Pi) \wedge f \in Pr(\Sigma)\})$.

On the other hand we can start from $Pr(\Sigma \cup \Pi)$ and then restrict these functions appropriately.

Definition 12.

(i) Let $\iota: \Sigma^* \to (\Sigma \cup \Pi)^*$ be the embedding and $\nu: (\Sigma \cup \Pi)^* \to \Pi^*$ the projection defined as monoid homomorphisms by $\iota(a) = a$, $\nu(b) = b$, $\nu(b') = e$ for all $a \in \Sigma$, $b \in \Pi$ and $b' \in (\Sigma \cup \Pi) \backslash \Pi$.

(ii) The class $Pr_2(\Sigma, \Pi)$ of *primitive recursive transformations from* Σ^* *to* Π^* is defined by $Pr_2(\Sigma, \Pi) := \{\nu \circ \phi \circ \iota \mid \phi \in Pr(\Sigma \cup \Pi)\}$. (As ν can be understood as an element of $Pr(\Sigma \cup \Pi)$ we could have taken without loss of generality those ϕ with *range* $(\phi) \subseteq \Pi^*$.)

In order to show that these definitions are equivalent we give some preparatory lemmas and theorems.

Lemma 3. *For* $f \in Pr(\Sigma)$ *there exist* $f' \in Pr(\Sigma \cup \Pi)$ *with* $f = \nu \circ f' \circ \iota$.

Proof: The base functions over Σ^* are natural restrictions of corresponding functions over $(\Sigma \cup \Pi)^*$. If $f = g_1 \circ g_2$ and $g_i = g_i' \circ \iota$ with $g_i' \in Pr(\Sigma \cup \Pi)$, then $f = (g_1' \circ g_2') \circ \iota$ with $(g_1' \circ g_2') \in Pr(\Sigma \cup \Pi)$. If f is defined by primitive recursion over $g, h_1, \ldots, h_r$ and $g = g' \circ \iota$, and so on, then f' can be given by the following recursion schema:

$$f'(\tilde{x}, e) = g'(\tilde{x})$$
$$f'(\tilde{x}, a_i w) = h_i'(\tilde{x}, w, f'(\tilde{x}, w)) \quad (i = 1, \ldots, r)$$
$$f'(\tilde{x}, b_j w) = f'(\tilde{x}, w) \quad b_j \in (\Sigma \cup \Pi) \backslash \Sigma$$

The proof shows in particular that the embedding ι is an encoding.

Lemma 4. *For each* $c \in K(\Sigma, \Pi)$ *there is* $c' \in Pr(\Sigma \cup \Pi)$ *with* $c = \nu \circ c' \circ \iota$.

Proof: For an encoding $c: \Sigma^* \to \Pi^*$ we have $\phi_i \in Pr(\Pi)$ such that $c \circ \lambda_i = \phi_i \circ c$ $(i = 1, \ldots, r)$. Let ϕ_i' be the corresponding function in $Pr(\Sigma \cup \Pi)$ according to Lemma 3. Now, the assertion can be satisfied by the following $c' \in Pr(\Sigma \cup \Pi)$:

$$c'(e) = c(e)$$
$$c'(a_i w) = \phi_i'(c'(w)) \quad (i = 1, \ldots, r)$$
$$c'(b_j w) = c'(w) \quad \text{for } b_j \in (\Sigma \cup \Pi) \backslash \Sigma.$$

This means that encodings are primitive recursive functions in a certain sense, in particular: $K(\Sigma, \Sigma) \subseteq Pr(\Sigma)$. — The next lemma is obvious.

Lemma 5. *The composition of two encodings is an encoding.*

We shall construct now two encodings that are of special importance. A word $w \in \Sigma^*$ can be understood as the r-adic representation of a natural number.

Let $\beta: \mathbb{N} \to \Sigma^*$ be defined by $\beta(0)=e$ and $\beta(m+1)=h(\beta(m))$, where h is given by $h(e)=a_1$, $h(a_i w)=a_{i+1} w$ $(i=1, \ldots, r-1)$ and $h(a_r w)=a_1 h(w)$.

The function $\alpha: \Sigma^* \to \mathbb{N}$, where $\alpha(e)=0$ and $\alpha(a_i w)=i+r \cdot \alpha(w)$ is the corresponding "decoding", i.e. $\alpha \circ \beta = id_{\mathbb{N}}$.

Theorem 12. *α and β are encodings, i.e.*
(i) For each $f \in Pr(\Sigma)$ there is $f' \in Pr$ with $\alpha \circ f = f' \circ \alpha$.
(ii) For each $g \in Pr$ there is $g' \in Pr(\Sigma)$ with $\beta \circ g = g' \circ \beta$.

Proof: The proof is by induction on the structure of $Pr(\Sigma)$ resp. Pr.

(i) For $f=\lambda_i$ the assertion is proved by f' with $f'(m)=i+r \cdot m$. The other base functions and compositions are dealt with in an obvious way. If f is defined by primitive recursion we construct $f' \in Pr$ such that the recursion over words is simulated in $\mathbb{N}$. To this end we need the following auxiliary functions:

$$\mu_r(m) := [(m-1) \bmod r]+1 \quad \text{and} \quad \delta_r(m) := \left\lceil \frac{m-1}{r} \right\rceil$$

They allow a recursion formula for β: $\beta(m)=\lambda_{\mu_r(m)}(\beta(\delta_r(m)))$.

The function η_r with $\eta_r(m)=|\beta(m)|$ gives the number of recursions which have to be simulated. δ_r, η_r, μ_r are primitive recursive. The details together with the construction of f' are rather lengthy and must be omitted here.

(ii) The function h from above proves the assertion for the successor function.

Primitive recursion is simulated by means of a function ϕ where $\phi(w)=a_1^{|\alpha(w)|}$.

Corollary. *There exist a bijective $f: \Sigma^* \to \Pi^*$ such that f and f^{-1} are encodings.*

Now, we are able to prove the equivalence of our definitions.

Theorem 13. $Pr_1(\Sigma, \Pi)=Pr_2(\Sigma, \Pi)$

Proof: 1. For $f \in Pr_1(\Sigma, \Pi)$ we have $f=c \circ f'$ where $c \in K(\Sigma, \Pi)$ and $f' \in Pr(\Sigma)$. By Lemma 4, $c=v \circ c' \circ \iota$ with $c' \in Pr(\Sigma \cup \Pi)$ and by Lemma 3, $f'=v \circ f'' \circ \iota$ with $f'' \in Pr(\Sigma \cup \Pi)$. (Note that we use different v's!) So, we conclude $f=v \circ c' \circ \iota \circ v \circ f'' \circ \iota \in Pr_2(\Sigma, \Pi)$ because $\iota \circ v \in Pr(\Sigma \cup \Pi)$.

2. For $f \in Pr_2(\Sigma, \Pi)$ we have $f=v \circ f' \circ \iota$ where $f' \in Pr(\Sigma \cup \Pi)$. By the corollary there is a bijective $c_1: \Pi^* \to (\Sigma \cup \Pi)^*$ such that c_1 and c_1^{-1} are encodings. Using Lemma 4 we see that $c_1 \circ v \in Pr(\Sigma \cup \Pi)$ and so $f=c_1^{-1} \circ c_1 \circ v \circ f' \circ \iota$ with $c_1 \circ v \circ f' \in Pr(\Sigma \cup \Pi)$. Now we take a bijective encoding $c_2: \Sigma^* \to (\Sigma \cup \Pi)^*$ that allows to represent f by $f=c_1^{-1} \circ c_2 \circ c_2^{-1} \circ c_1 \circ v \circ f' \circ \iota$. As c_2^{-1} is an encoding there is $f'' \in Pr(\Sigma)$ such that $c_2^{-1} \circ c_1 \circ v \circ f' = f'' \circ c_2^{-1}$, and so $f \in Pr_1(\Sigma, \Pi)$.

This result justifies the notion of a primitive recursive transformation. From now on we simply write $Pr(\Sigma, \Pi)$. The second definition implies a natural generalization of Grzegorczyk classes.

Definition 13. For $n \in \mathbb{N}$ we call $E_n(\Sigma, \Pi) := \{v \circ f \circ \iota \mid f \in E_n(\Sigma \cup \Pi)\}$ the *n-th Grzegorczyk class of transformations.*

As an immediate consequence we see that these classes form a hierarchy for $Pr(\Sigma, \Pi)$.

Theorem 14.
(i) $E_n(\Sigma, \Pi) \subset E_{n+1}(\Sigma, \Pi)$ for all $n \in \mathbb{N}$
(ii) $Pr(\Sigma, \Pi) = \bigcup_{n \in \mathbb{N}} E_n(\Sigma, \Pi)$

In the remaining part of this section we investigate the relation between Grzegorczyk classes of transformations and those of wordfunctions.

Lemma 6. *For all $n \in \mathbb{N}$ the following holds:*

$$E_n(\Sigma, \Pi) \circ E_n(\Sigma) = E_n(\Sigma, \Pi) = E_n(\Pi) \circ E_n(\Sigma, \Pi)$$

The proof is similar to that of Theorem 13.

Let $\gamma: \Sigma^* \to \Pi^*$ be the bijective encoding of Theorem 12 and its corollary. Then we can give a stronger version of that theorem.

Theorem 15. *For all $n \geq 3$, if $|\Pi| = 1$, or for all $n \geq 1$, if $|\Pi| > 1$, we have:* $E_n(\Sigma) = \gamma^{-1} \circ E_n(\Pi) \circ \gamma$.

For the proof we proceed as in the proof of Theorem 12. It is possible to take the auxiliary functions from E_1 resp. $E_1(\Sigma)$ or $E_1(\Pi)$. The difference depending on $|\Pi|$ is due to the fact that for $|\Pi| > 1$ we have $\gamma \in E_1(\Sigma, \Pi)$ and $\gamma^{-1} \in E_1(\Pi, \Sigma)$, whereas in the case of $|\Pi| = 1$ the function γ (i.e. α of Theorem 12) has exponential growth that cannot be eliminated in the proof. However, we can show a weaker result for $|\Pi| = 1$ and $n = 0, 1, 2$:

$$E_n(\Sigma) = \alpha^{-1} \circ E_{n+1} \circ \alpha$$

Unfortunately, we do not know anything about the other inclusion direction.

This difference between arithmetical and non-arithmetical theory turns out to be fundamental. We abbreviate for the following

$$\#(\Pi) := \begin{cases} 1 & \text{if } |\Pi| > 1 \\ 3 & \text{if } |\Pi| = 1 \end{cases}$$

Lemma 7. *Let Π_1, Π_2 be non-empty subalphabets of Σ. Then we have for $n \geq \#(\Pi_1 \cup \Pi_2)$ the following property: for each $f \in E_n(\Sigma)$ there is $f' \in E_n(\Pi_1, \Pi_2)$ with $f' = v \circ f \circ \iota$ where $\iota: \Pi_1^* \to \Sigma^*$ and $v: \Sigma^* \to \Pi_2$ as above.*

Theorem 16. *Let $\Pi_1, \Pi_2, \Pi_3 \subseteq \Sigma$. Then we have for $n \geq \#(\Pi_1 \cup \Pi_3)$*

$$E_n(\Pi_2, \Pi_3) \circ E_n(\Pi_1, \Pi_2) \subseteq E_n(\Pi_1, \Pi_3)$$

Proof: Using Lemma 3 we show for $f \in E_n(\Pi_1, \Pi_2)$ and $g \in E_n(\Pi_2, \Pi_3)$ the existence of $h \in E_n(\Sigma)$ such that $g \circ f = v \circ h \circ \iota$. From Lemma 7 we have $g \circ f \in E_n(\Pi_1, \Pi_3)$.

Corollary. For $n \geq \max(\#(\Sigma), \#(\Pi))$ we have:
(i) $E_n(\Pi, \Sigma) \circ E_n(\Sigma, \Pi) = E_n(\Sigma)$
(ii) $Pr(\Pi, \Sigma) \circ Pr(\Sigma, \Pi) = Pr(\Sigma)$

This corollary shows in particular that the definitions of primitive recursive wordfunctions and of primitive recursive transformations are compatible.

As encodings are special primitive recursive transformations our results give some information about their influence: let $c \in E_n(\Sigma, \Pi)$ be an encoding and $d \in E_n(\Pi, \Sigma)$ with $d \circ c = id_{\Sigma^*}$, then we have for $n \geq \max(\#(\Sigma), \#(\Pi))$ by Lemma 6 and Theorem 15 that $c \circ E_n(\Sigma) \circ d \subseteq E_n(\Pi)$ or $c \circ E_n(\Sigma) \subseteq E_n(\Sigma) \circ c$. So, encodings have no influence on the complexity of these classes of functions provided that n is not too small. Yet, the example of the encoding α shows that small Grzegorczyk classes may be shifted.

Part 3: Applications to Automata Theory

The theory of primitive recursive wordfunctions as developed in Part 1 enables us to give a natural classification of transductions defined by automata within primitive recursive hierarchies. First we introduce primitive recursive relations by which we can define partial primitive recursive functions on the one hand, and classify languages on the other hand.

Definition 14. Let $f \in F(\Sigma)$ and $K \subseteq F(\Sigma)$.

(i) $Rel(f) := f^{-1}(\{e\})$ is called the relation recognized by f.

(ii) $Rel(K) := \{Rel(f) \mid f \in K\}$

(iii) $P(K) := \{f \mid$ there is $g \in K$ and $R \in Rel(K)$ such that $f = g \mid R\}$.

In particular, we call $f \in P(Pr(\Sigma))$ partial primitive recursive. This is justified by the following result.

Theorem 17. *Let* $|\Sigma| \geq 2$ *and* $n \geq 1$. *If for* $M \in \mathfrak{M}(\Sigma)$ *there is* $t \in E_n(\Sigma)$ *such that* $|T_M(\tilde{x})| < |t(\tilde{x})|$ *for all* $\tilde{x} \in domain(f_M)$ *then we have* $f_M \in P(E_n(\Sigma))$.

It is easy to prove that the primitive recursive relations form hierarchies analoguous to functions.

Theorem 18. *For all* $k, n \in \mathbb{N}$ *we have*

(i) $Rel(E_n^{(k)}(\Sigma)) \subseteq Rel(E_{n+1}^{(k)}(\Sigma))$

(ii) $Rel(R_n^{(k)}(\Sigma)) \subseteq Rel(R_{n+1}^{(k)}(\Sigma))$

(iii) $Rel(L_n^{(k)}(\Sigma)) \subseteq Rel(L_{n+1}^{(k)}(\Sigma))$

For $k \geq 1$ *and* $n \geq 2$ *we can even show that*

(iv) $Rel(E_n^{(k)}(\Sigma)) \subset Rel(E_{n+1}^{(k)}(\Sigma))$

It is an open problem whether $Rel(E_0(\Sigma)) \neq Rel(E_1(\Sigma)) \neq Rel(E_2(\Sigma))$.

Furthermore, we investigate some closure properties of these classes. The results are put together in the following theorem.

Theorem 19.

(i) $Rel(R_n^{(k)}(\Sigma))$, $Rel(L_n^{(k)}(\Sigma))$ *and* $Rel(E_m^{(k)}(\Sigma))$ *are closed under the operations of complement, union and intersection if* $k \geq 0$, $n \geq 1$ *and* $m \geq 0$.

(ii) $Rel(R_n(\Sigma))$, $Rel(L_n(\Sigma))$ *and* $Rel(E_m(\Sigma))$ *are closed under cartesian product if* $n \geq 1$ *and* $m \geq 0$.

(iii) *For all* $n \in \mathbb{N}$ $Rel(E_n^{(1)}(\Sigma))$ *is closed under inverse mappings from* $E_n^{(1)}(\Sigma)$. (*Analoguous for* R_n *and* L_n.)

(iv) *For* $n \geq 1$ $Rel(E_n^{(1)}(\Sigma))$ *is closed under concatenation.*

(v) *For* $n \geq 3$ $Rel(E_n^{(1)}(\Sigma))$ *is closed under crossproduct* $(M^+ := M \cup M^2 \cup \ldots)$.

Meyer and Ritchie [13] call $f \in F$ elementary honest iff *graph* $(f) \in Rel(E_3)$. There are such functions with arbitrary growth, We can prove a stronger result.

Theorem 20. *Let* $|\Sigma| \geq 2$. *For each recursive* $f \in F(\Sigma)$ *there is* $g \in F(\Sigma)$ *such that* $|f(\tilde{x})| \leq |g(\tilde{x})|$ *for all* $\tilde{x} \in domain(f)$ and *graph* $(g) \in Rel(E_1(\Sigma))$.

We have now provided the necessary tools for the classification of functions and sets defined by automata. It turns out that the primitive recursive hierarchies do not form an appropriate framework for this task. Restricted Turing machines define functions that are on a very low hierarchical level.

Theorem 21. *Let* $|\Sigma| \geq 2$. *The non-regular language* $\{a_2^n a_1^n \mid n \in \mathbb{N}\}$ *and the non-context-free language* $\{a_1^n a_2^n a_1^n \mid n \in \mathbb{N}\}$ *are in* $Rel(E_0(\Sigma))$.

Theorem 22. *The class* $Reg(\Sigma)$ *of regular languages over* Σ *is properly contained in* $Rel(E_0^{(1)}(\Sigma))$.

$FS(\Sigma)$ shall denote the set of finite state transductions over Σ, i.e. the set of those functions which are computable by $M \in \mathfrak{M}(\Sigma)$ without storage tapes. We use lower index 1 to denote one-way input and lower index t to denote total functions.

The following results hold for the case $|\Sigma| \geq 2$, most of them are also true for $|\Sigma| = 1$.

Theorem 23.

(i) $FS_{1t}^{(1)}(\Sigma) \# E_0^{(1)}(\Sigma)$

(ii) $R_0(\Sigma) \subset FS_{1t}(\Sigma) \subset L_1(\Sigma)$

(iii) $R_1^{(1)}(\Sigma) \not\subseteq FS_{1t}^{(1)}(\Sigma)$

(iv) $FS_t^{(1)}(\Sigma) \subset E_1^{(1)}(\Sigma)$

(v) $FS_t(\Sigma) \subset E_2(\Sigma)$

Let $PD(\Sigma)$ be the class of all pushdown transductions, i.e. those functions which are computable by $M \in \mathfrak{M}(\Sigma)$ with exactly one pushdown store as storage tape. Indices are used as above.

Theorem 24.

(i) $PD_1(\Sigma) \subset P(E_1(\Sigma))$

(ii) $PD_{1t}^{(1)}(\Sigma) \# E_0^{(1)}(\Sigma)$

(iii) $PD_{1t}^{(1)}(\Sigma) \# R_1^{(1)}(\Sigma)$

Let $Cf(\Sigma)$ resp. $Cfd(\Sigma)$ be the class of all context-free resp. deterministic languages over Σ.

Theorem 25.

(i) $Cfd(\Sigma) \subset Rel(E_1^{(1)}(\Sigma))$

(ii) $Rel(E_0^{(1)}(\Sigma)) \not\subseteq Cfd(\Sigma)$

(iii) $Cf(\Sigma) \subset Rel(E_2^{(1)}(\Sigma))$

(iv) $Rel(E_0^{(1)}(\Sigma)) \not\subseteq Cf(\Sigma)$

Let $LB(\Sigma)$ be the class of all functions that are computable by a linear tape bounded $M \in \mathfrak{M}(\Sigma)$.

Theorem 26. $E_1(\Sigma) \subseteq LB(\Sigma)$.

Finally we conclude by Theorem 6 that the functions and languages defined by linear bounded automata or stack automata are in $E_3(\Sigma)$ $(P(E_3(\Sigma)), Rel(E_3(\Sigma)))$: both types of automata are time bounded in E_3.

References

[1] Asser, G.: Rekursive Wortfunktionen. Zeitschr. math. Logik Grdl. Math. *6*, 258—278 (1960).

[2] Axt, P.: Iteration of primitive recursion. Zeitschr. math. Logik Grdl. Math. *11*, 253—255 (1965).

[3] Cobham, A.: The intrinsic computational difficulty of functions. In: Logic, Methodology, and Philosophy of Science (Bar-Hillel, Y., ed.), Amsterdam: North-Holland Publ. Comp. 1965.

[4] Eilenberg, S., Elgot, C. C.: Recursiveness. New York-London: Academic Press. 1970.

[5] Grzegorczyk, A.: Some classes of recursive functions. Rozprawy matem. *4*, 1—46 (1953).

[6] Heinermann, W.: Untersuchungen über die Rekursionszahlen rekursiver Funktionen. Dissertation, Münster, 1961.

[7] v. Henke, F. W.: Verallgemeinerte primitiv-rekursive Funktionen zwischen freien Monoiden. GMD Seminarbericht Nr. 46 (1972).

[8] v. Henke, F. W.: Primitiv-rekursive Transformationen. Diss. Bonn 1973, to appear in: Berichte der GMD.

[9] v. Henke, F. W., Indermark, K., Weihrauch, K.: Hierarchies of primitive recursive wordfunctions and transductions defined by automata, in: Automata, Languages, and Programming (Nivat, M., ed.), Amsterdam: North-Holland Publ. Comp. 1972.

[10] Indermark, K.: Pushdown transductions as primitive recursive wordfunctions. GMD Seminarbericht Nr. 56 (1972) and Proc. IRIA Sem. (1972).

[11] Kreider, D. L., Ritchie, R. W.: A universal two-way automaton. Archiv math. Logik Grdl. *9*, 43—58 (1966).

[12] Meyer, A. R., Ritchie, D. M.: Computational complexity and program structure. IBM Research Report RC 1817 (1967).

[13] Meyer, A. R., Ritchie, D. M.: A classification of the recursive functions. Zeitschr. math. Logik Grdl. Math. *18*, 71—82 (1972).

[14] Müller, H.: Über die mit Stackautomaten berechenbaren Funktionen. Archiv math. Logik Grdl. *13*, 60—73 (1970).

[15] Ritchie, R. W.: Classes of predictably computable functions. Trans. AMS *106*, 139—173 (1963).

[16] Ritchie, R. W.: Classes of recursive functions based on Ackermann's function. Pacific J. Math. *15*, 1027—1044 (1965).

[17] Rose, G., Weihrauch, K.: Eine Charakterisierung der Klassen L_1 und R_1 primitiv-rekursiver Wortfunktionen. GMD Seminarbericht Nr. 63 (1973).

[18] Schwichtenberg, H.: Rekursionszahlen und die Grzegorczyk-Hierarchie. Archiv math. Logik Grdl. *12*, 85—97 (1969).

[19] Tsichritzis, D.: A note on comparison of subrecursive hierarchies. Inf. Proc. Letters *1*, 42—44 (1971).

[20] Weihrauch, K.: Hierarchien primitiv-rekursiver Wortfunktionen I. GMD Seminarbericht Nr. 49 (1972).
[21] Weihrauch, K.: Hierarchien primitiv-rekursiver Wortfunktionen II. GMD Seminarbericht Nr. 57 (1972).
[22] Weihrauch, K.: Teilklassen primitiv-rekursiver Wortfunktionen. Dissertation, Bonn, 1973; to appear in: Berichte der GMD.

Dr. F. W. v. Henke
Institut für Rechner- und Programmstrukturen
Gesellschaft für Mathematik und Datenverarbeitung
Schloß Birlinghoven
D-5205 St. Augustin 1
Bundesrepublik Deutschland

Prof. Dr. G. Rose
Case Western Reserve University
Cleveland, Ohio
U.S.A.

Prof. Dr. K. Indermark
Dr. K. Weihrauch
Institut für Angewandte Mathematik und Informatik
Universität Bonn
Wegelerstraße 6
D-5300 Bonn
Bundesrepublik Deutschland