Lecture Notes in Computer Science

Edited by G. Goos, Karlsruhe and J. Hartmanis, Ithaca

5

International Symposium on Theoretical Programming

Edited by Andrei Ershov and Valery A. Nepomniaschy

Springer-Verlag
Berlin · Heidelberg · New York 1974

Proc.of the Internat.Symp.on Theoretical Programming
(E.Ershov, V.A.Nepomniaschy ed.), Novosibirsk, 1972.
Lecture Notes in Computer Science, Springer, Vol.5, 152-171.

Certain decompositions of Gödel Numbering and the Semantics of Programming Languages

B. Buchberger *)

N o t a t i o n :

N ... set of natural numbers including o.

$N^k := \underbrace{N \times \ldots \times N}_{k \text{ times}}$

The whole treatment will be over the natural numbers. Thus, by a function (predicate) we normally mean an arithmetical function (predicate). Of course, all what follows could be done over arbitrary (effectively given) denumerable domains using an appropriate version of recursive function theory (ASSER 6o, SHOENFIELD 71 etc.).

$\bar{A}$ ... complement of the set A.

$A \subset_e B$ means $A \subset B$ and $A \neq B$.

If f, g denote functions then we often write fg(x) for f(g(x)) and $f^{(t)}(x)$ for $\underbrace{f \ldots f}_{t \text{ times}}(x)$. In addition, $f^{(o)}(x) := x$.

(x), (Ex) ... universal and existential quantification over N.

Let f be a function, $A \subset N$. We define $f(A) := \{y \mid (Ex \in A)(f(x)=y)\}$.

An analogous notation will be used for n-ary functions.

|C| ... cardinality of the set C.

$|C| = \infty$ ... C is an infinite set.

P_n ... set of all n-ary partial recursive functions.

R_n ... set of all n-ary total recursive functions.

---

*) Inst. f. Num. Mathematik und Elektronische Informationsverarbeitung, Universität Innsbruck, A6020 Innsbruck, Austria/Europe.

1. A certain type of universal functions and the semantics of universal programming languages

Definition 1.1:

τ is a pairing function: $\Leftrightarrow \tau \in R_2$, τ is 1-1, and $\tau(N,N)$ is decidable.

Notation: Let τ be a pairing function. By τ_1, τ_2 we denote total recursive functions for which

$$\tau_1\tau(x,y) = x, \quad \tau_2\tau(x,y) = y,$$

$$z \in \tau(N,N) \rightarrow \quad \tau(\tau_1(z),\tau_2(z)) = z.$$

Further, $\langle x,y \rangle$ stands for $\tau(x,y)$, where τ is an arbitrary pairing function that remains fixed throughout the following.

Definition 1.2. (ROGERS 58, USPENSKIĬ 60):

Ψ describes a Gödel numbering of the unary partial recursive functions (in short: Ψ "is" a Gödel numbering) $\Leftrightarrow$

(GN1) $\Psi \in P_2$,

(GN2) for all $\Psi' \in P_2$ there exists a $\sigma \in R_1$ such that

$$\Psi'(p',x) = \Psi(\sigma(p'),x).$$

Remarks: The relevance of Gödel numberings for a semantical theory of universal programming languages has been pointed out in various papers (see, for instance, SCHWENKEL 66; the earliest investigation in this direction is due to USPENSKIĬ 56). Briefly summarized, this relevance relies on the following observations:

(L1) We could content ourselves with the knowledge we obtain on the semantics of a programming language L by knowing the result $^{L}\Psi(p,x)$ of the application of the program p to the data x for all p,x. Otherwise stated, we could conceive the semantics of a programming language to be given by the correspondence

$$p \longmapsto (\lambda x)(^{L}\Psi(p,x)),$$

where p ranges over all programs of the language L.

(L2) The result $^{L}\Psi(p,x)$ should be effectively computable form p and x.

(L3) $\{ (\lambda x)(^{L}\Psi(p,x)) \mid p \in N \} = P_1$ should hold for a universal language L.

(L4) Given a description p' of a function g in some standard mathematical notation, e. g. in some other programming language (that could be equally well characterized by some binary function Ψ'), a program p for g in L should be effectively obtainable from p'.

It is clear that requirement (L2) is guaranteed by (GN1) and requirements (L3) and (L4) are simultaneously guaranteed by (GN2). Hence, if we accept (L1) the theory of Gödel numberings would just coincide with the semantical theory of universal programming languages.

However, what makes the concept of (L1) unsatisfactory is, first, the total absence of a notion of "computation in the language L" (i. e. all intermediate stages between input of data and output of the result remain unspecified); and second, the impossibility of seperating the role which input/output coding plays for the determination of the Gödel numbering associated with a given language. Thus, Rogers isomorphism theorem (ROGERS 58) can be viewed

as telling that by suitable program coding every universal language can determine every possible Gödel numbering. The data input/output coding has a similarly severe influence. Our Definition 1.4 is intended to give a precise version of a semantical concept for universal programming languages that takes into account both the stepwise work of programs and the role of input/output. Definition 1.3 is preparatory.

Definition 1.3: Let $\bar{\psi}, \kappa \in R_1$. We say that ψ^* is obtained by "conditioned iteration" from $\bar{\psi}$ and κ, if $\psi^*(\xi)$ is defined to be the first $\bar{\psi}^{(t)}(\xi)$ in the sequence $\xi, \bar{\psi}(\xi), \bar{\psi}^{(2)}(\xi), \ldots$ for which $\kappa\bar{\psi}^{(t)}(\xi) = 0$. More formally,

(Ci) $$\psi^*(\xi) = \begin{cases} \xi, & \text{if } \kappa(\xi) = 0 \\ \psi^*\bar{\psi}(\xi), & \text{otherwise.} \end{cases}$$

Notation: We write $[\bar{\psi},\kappa]$ for the function which is obtained form $\bar{\psi},\kappa$ by conditioned iteration.

Remarks: $[\bar{\psi},\kappa](\xi)$ can be viewed, for instance, as defining the terminal state which an (infinite) automaton with transition function $\bar{\psi}$ and "termination criterion" κ, eventually, assumes when started in state ξ. Of course, $[\bar{\psi},\kappa](\xi)$ may be undefined for certain ξ. Thus, $[\bar{\psi},\kappa] \in P_1$ for $\bar{\psi},\kappa \in R_1$.

Definition 1.4: $\bar{\psi},\kappa$ define a universal automaton
(in short: $\bar{\psi},\kappa$ "are" universal) : $\Longleftrightarrow$

(U1) $\bar{\psi},\kappa \in R_1$

(U2) there exist $\rho \in R_1$ and $\gamma \in R_2$ such that $\rho[\bar{\psi},\kappa]\gamma$ is a Gödel numbering.

Remarks: In BUCHBERGER 72 we give a detailed exposition of the intuitive reasons why we think that this notion of "universal" $\bar{\Psi},\kappa$ is an adequate precise substitute for the notion of a universal programming language. We briefly summarize the discussion given there:

1. The semantics of a programming language can be given by telling, first, what is done during one "step" of a computation according to a program of the language (i. e. by giving the successor "state" $\bar{\Psi}(\xi)$ for every possible state ξ that may arise in a computation in the language) and, second, which states are "terminal" (this is the role of κ). Of course, $\bar{\Psi}$ and κ should be total recursive. One "component" of the state is the program. By the Definition 1.4. it is not excluded that the program is altered during the computation. Hence, this concept is wide enough to encompass, for instance, machine languages.

2. Of course, also functions Ψ^* that are not defined in terms of two functions $\bar{\Psi},\kappa$ by the scheme (C1) can be used to define the semantics of programming languages by giving the terminal "state" $\Psi^*(\xi)$ corresponding to every possible initial state met in the computations of the language (see, for instance, the function apply [fn;x;a] for LISP in McCARTHY 62, or the flow chart interpreter $\mathfrak{U}(d)(\sigma)$ in SCOTT 71).
   However, as long as the available hardware essentially functions in the way given by the scheme (C1), at some stage of the implementation of a language its semantics must be given by a function of the form $[\bar{\Psi},\kappa]$ (compare, however, Remark 4. in Section 3).

3. Infact, a review of the relevant literature schows that most "programming languages" (including the various computability formalisms of recursive

function theory) are (informally) given in the form $[\bar{\Psi},\kappa]$. Thus, for instance, the Vienna method (LUCAS/LAUER/STIGLEITNER 68) uses essentially this form. Also, our concept (even that given in Definition 1.5 below) is still wide enough to embrace the concepts given in SCOTT 67.

4. We would not like to dispense universal programming languages from being capable of defining a Gödel numbering for some suitable input/output conventions. This is guaranteed by (U2). The restriction laid onto admissable input/output functions ρ,γ (namely their recursiveness) seems to be very wide. However, it is difficult to require additional properties for these functions without excluding, perhaps, interesting cases prematurely. An important special case is singled out by the following

Definition 1.5: $\bar{\Psi},\kappa$ define a normal universal automaton (in short: $\bar{\Psi},\kappa$ "are" normal universal"): $\Longleftrightarrow$

(NU1) $\bar{\Psi},\kappa \in R_1$

(NU2) there exists a pairing function τ and a $\bar{\bar{\Psi}} \in R_2$ such that

$$\kappa\tau(p,n) \neq 0 \hookrightarrow \bar{\Psi}\tau(p,n) = \tau(p,\bar{\bar{\Psi}}(p,n))$$

(NU3) there exist $\rho,\gamma \in R_1$ such that

$(\lambda p,x)(\rho\tau_2[\bar{\Psi},\kappa]\tau(p,\gamma(x)))$ is a Gödel numbering.

Remarks: The above definition characterizes those "interpreters" (automata) $\bar{\Psi},\kappa$ that do not change the program during execution time. Whether or not an interpreter changes the programs by execution also depends on how we split the states ξ in a program component $p:=\tau_1(\xi)$ and a working store compo-

ponent $\eta := \tau_2(\xi)$. In order that some given $\bar{\Psi}, \kappa$ define a normal universal automaton (NU2) requires that $\bar{\Psi}$ does not alter the program for some possible splitting of the states. Given such a splitting it is natural to require that input/output should refer to the working store component only. This is just what is expressed in (NU3).

2. Propositions on the decompositions of Gödel numberings given in the Definitions 1.4 and 1.5.

In this section we state some theorems concerning the decompositions of Gödel numberings that appear in the Definitions 1.4 and 1.5. Partly, their proofs are quite elaborate and will be given in detail in BUCHBERGER/ROIDER 72, if not stated otherwise.

First ist is good to know that every Gödel numbering has a decomposition of the form given in Definition 1.4:

Theorem 2.1: Every Gödel numbering Ψ can be written in the form

$$\Psi(p,x) = \rho\tau_2[\bar{\Psi},\kappa]\tau(p,\gamma(x)),$$

where

$$\bar{\Psi}\tau(p,\xi) = \tau(p,\bar{\Psi}(p,\xi))$$

with suitable $\rho,\kappa,\gamma \in R_1, \bar{\Psi}, \tau \in R_2$, τ being a pairing function.

Remarks: Theorem 2.1 is a consequence of Corollary 2.6 in BUCHBERGER 71, where it is shown that suitable ρ, κ, γ, $\bar{\Psi}$, τ can be obtained by mere substitution from arbitrary functions α,σ that satisfy

(ST1) $(E\varepsilon)(a)(\alpha(\varepsilon,a) = \varepsilon)$ ("empty-storage assumption")

(ST2)
$$\alpha(\sigma(s,a,c),b) = \begin{cases} c, & \text{if } b = a \quad (\text{"component-wise change} \\ \alpha(s,b), & \text{otherwise} \quad \text{of storage"}). \end{cases}$$

(For $\bar{\Psi}$, in addition to α, σ, the successor function is needed). Of course, some versions of functions α, σ appear frequently in recursive function theory and studies on storage models. However, it is noteworthy that one can do without any further assumptions on α, σ which appear elsewhere (for instance "finite storage assumption" etc. in BEKIĆ/WALK 71, or assumptions (3.9), (3.1o) in McCARTHY/PAINTER 67). Theorem 2.1 may be viewed as a kind of normal form theorem which, for our purposes, is more suitable than Kleene's.

Second, the following equivalence theorem gives an intimate connection between Blum's complexity theory(BLUM 67) and the concept in Definition 1.4.

Theorem 2.2a: Let Ψ be a Gödel numbering and

$$\Psi(p,x) = \rho[\bar{\Psi},\kappa]\gamma(p,x)$$

for certain ρ, $\bar{\Psi}$, $\kappa \in R_1$, $\gamma \in R_2$. Define

$$\Phi(p,x) := (\mu t)(\kappa\bar{\Psi}^{(t)}\gamma(p,x) = o),$$

then Φ is a step counting function for Ψ in the sense of Blum, i.e.

(S1) $\Phi(p,x)$ defined $\longleftrightarrow$ $\Psi(p,x)$ defined

(S2) $\Phi(p,x) = m$ is decidable.

Theorem 2.2b: If Ψ is a Gödel numbering and Φ a step counting function for Ψ (i.e. (S1), (S2) hold) then one can find ρ, $\bar{\Psi}$, $\kappa \in R_1$ and $\gamma \in R_2$ such that

$$\Psi(p,x) = \rho[\bar{\Psi},\kappa]\gamma(p,x), \quad \text{and}$$

$$\Phi(p,x) = (\mu t)(\kappa\bar{\Psi}^{(t)}\gamma(p,x) = 0).$$

Remarks: It is easy to check Theorem 2.2a. For the proof of Theorem 2.2b choose

$$\gamma(p,x) := \langle p,x,0\rangle,$$

$$\bar{\Psi}(\langle p,x,t\rangle) := \langle p,x,t+1\rangle,$$

$$\kappa(\langle p,x,t\rangle) := \begin{cases} 0, & \text{if } \Phi(p,x) = t, \\ 1, & \text{otherwise,} \end{cases}$$

$$\rho(\langle p,x,t\rangle) := \begin{cases} \Psi(p,x), & \text{if } \Phi(p,x) = t, \\ 0, & \text{otherwise.} \end{cases}$$

Here $\langle\ \rangle$ is the notation for a pairing function which, in addition, is onto. Subsequently, these functions ρ, $\bar{\Psi}$, κ, γ will serve as an interesting example.

By Theorem 2.2a we can use all the information given in Blum's theory to investigate universal $\bar{\Psi}$, κ. By Theorem 2.2b we can conclude that the intuitive concept of a computation according to some program (and, hence, of the semantics of programming languages), which lies behind Blum's complexity theory, is just the same as that which has been made precise in Definition 1.4. This is yet another reason for maintaining the adequacy of this definition.

Next, we shall give some propositions on the "possible" ρ, κ, γ in Definition 1.4. First, a characterization of the possible γ:

Theorem 2.3: Let $\gamma \in R_2$.
There exist ρ, $\bar{\psi}$, $\kappa \in R_1$ such that $(\lambda p,x)(\rho[\bar{\psi},\kappa]\gamma(p,x))$ is a Gödel numbering $\Longleftrightarrow$ there exists an $f \in R_1$ such that $(\lambda p,x)(\gamma(f(p),x))$ is 1-1.

Remarks: The intuitive meaning of this theorem is that "possible" input functions γ, though not 1-1 everywhere, turn out to be 1-1 at least on an effectively constructible "cylinder". The proof of "$\Longrightarrow$" would be slightly easier if one knew that every $\psi^* \in P_1$ can be written in the form

$$\psi^* = \rho[\bar{\psi},\kappa]$$

for suitable ρ, $\bar{\psi}$, $\kappa \in R_1$. However one can show

Theorem 2.4: $\{\psi^* \mid \psi^* = \rho[\bar{\psi},\kappa]$ for some ρ, $\bar{\psi}$, $\kappa \in R_1\} \subset_+ P_1$.

Remark: This theorem also tells us that we should not expect that every ψ^* which might define a programming language in the manner described in Remark 2. (after Definition 1.4) can immediately be given in the form $[\bar{\psi},\kappa]$. Some input (and output, see Theorem 2.7) will be necessary, in general.

Theorem 2.3 still admitts a wide class of possible input functions. One might suspect that γ already could do "most of the computational work". However, Theorem 2.2a and the theorems of Blum's complexity theory (for instance the compression theorem) tell us that however complex (total re-

cursive!) input functions may be, the number of applications of $\bar{\Psi}$ (counted by the Φ of Theorem 2.2a) may be arbitrarily large for certain computations. In this respect the following theorem might be of some interest, too.

Theorem 2.5: Let $(\lambda p,x)(\rho[\bar{\Psi},\kappa]\gamma(p,x))$ be a Gödel numbering, where ρ, $\bar{\Psi}$, $\kappa \in R_1$, $\gamma \in R_2$. Then we can find a pairing function τ, such that

$$\rho[\bar{\Psi},\kappa]\gamma(p,x) = \rho[\bar{\Psi},\kappa]\tau(p,x).$$

Remark: Whenever $\bar{\Psi}$, κ are universal, then by this theorem they are so with respect to an input function τ that defines the initial states of the computations such that the program and data involved can uniquely be reconstructed from the initial state. Thus, prior to the "execution" the information contained in the program has not yet been used to alter the data and vice versa.

Theorem 2.6: Let $\rho \in R_1$.

There exist $\bar{\Psi}$, $\kappa \in R_1$, $\gamma \in R_2$ such that $(\lambda p,x)(\rho[\bar{\Psi},\kappa]\gamma(p,x))$ is a Gödel numbering $\Longleftrightarrow$ ρ is onto and $|\{\xi \mid (E\xi')(\rho(\xi') = \rho(\xi) \;\&\; \xi' < \xi)\}| = \infty$.

Remarks: This is a characterization theorem for the possible output functions ρ. In BUCHBERGER 72 we have shown that these ρ must not be 1-1, that every ρ of "large oscillation" is suitable, and furthermore that there exist suitable ρ which are not of large oscillation.+). The condi-

---

+) ρ is of large oscillation $:\Longleftrightarrow$ $(y,z)(Ex)(x > z \;\&\; \rho(x) = y)$. The total recursive functions of large oscillation are exactly the possible τ_1 (or τ_2) for paring functions τ. In MARKOV 47 it is shown that these functions are just the functions suitable as "output" functions in Kleene's normal form theorem.

tion in the theorem defines a class of functions which "lies between" the class of 1-1 functions and those of large oscillation.

Theorem 2.7: $\{ \psi^* \mid \psi^* = [\bar{\psi},\kappa]\gamma$ for some $\bar{\psi}, \kappa \in R_1, \gamma \in R_2 \} \subsetneq P_1$.

Remark: Compare Theorem 2.4.

Theorem 2.8: Let $\kappa \in R_1$.
There exist $\rho, \bar{\psi} \in R_1, \gamma \in R_2$ such that $(\lambda p,x)(\rho[\bar{\psi},\kappa]\gamma(p,x))$ is a Gödel numbering $\iff$ $|\{ \xi \mid \kappa(\xi) = 0 \}| = |\{ \xi \mid \kappa(\xi) \neq 0\}| = \infty$.

Remark: This gives a characterization of the possible κ.

Next we examine the functions γ which might occur in Definition 1.5.

Theorem 2.9: Let $\rho, \bar{\psi}, \kappa, \gamma \in R_1$, $\bar{\bar{\psi}} \in R_2$, τ be a pairing function,

$$\kappa\tau(p,\eta) \neq 0 \rightarrow \bar{\psi}\tau(p,\eta) = \tau(p,\bar{\bar{\psi}}(p,\eta)), \text{ and}$$

$(\lambda p,x)(\rho\tau_2[\bar{\psi},\kappa]\tau(p,\gamma(x)))$ be a Gödel numbering, then γ is 1-1 and $|\overline{\gamma(N)}| = \infty$.

Remarks: By Theorem 2.3 one might suspect that γ can be onto. However, this is excluded by the above theorem which even shows that infinitely many states of the working store must be"preserved for computation only". They cannot be met by the input function (in perfect correspondence to our experience with concrete languages).

Theorem 2.1o: Let $\gamma \in R_1$ be such that γ is 1-1, $|\overline{\gamma(N)}| \neq \infty$ and $\gamma(N)$ is decidable. Then γ may possibly occur in decompositions of the form given in (NU3).

Remarks: We have not been able to prove Theorem 2.1o without the assumption "$\gamma(N)$ decidable". Nor was it possible to derive "$\gamma(N)$ decidable" as a necessary condition in Theorem 2.9.

We still don't have a nice characterization of the possible ρ in Definition 1.5.

We next concentrate on the possible $\bar{\Psi}$. The study of the $\bar{\Psi}$ deserves our special interest since, intuitively, a programming language is most strikingly characterized by its $\bar{\Psi}$. However, the characterization we can present still suffers from a nonsymmetry (details of the proof and a discussion of the result appeared in BUCHBERGER 72):

Theorem 2.11: $\bar{\Psi}$, κ universal $\Longrightarrow$ there exists an infinite sequence $\xi_0, \xi_1, \ldots$ (all $\xi_i \in N$) such that

$$(\bar{\Psi}1) \qquad (i,j,t_1,t_2)(i \neq j \vee t_1 \neq t_2 \rightarrow \bar{\Psi}^{(t_1)}(\xi_i) \neq \bar{\Psi}^{(t_2)}(\xi_j)).$$

Remarks: Condition ($\bar{\Psi}$1) tells that if $\bar{\Psi}$, κ are universal then $\bar{\Psi}$ must contain infinitely many "tracks" that don't run into a "cycle" and don't "meet" each other.

Theorem 2.12: Let $\bar{\Psi}$ be such that for some $f \in R_1$

$$(\bar{\Psi}1') \qquad (i,j,t_1,t_2)(i \neq j \vee t_1 \neq t_2 \rightarrow \bar{\Psi}^{(t_1)}f(i) \neq \bar{\Psi}^{(t_2)}f(j)),$$

($\bar{\Psi}2$) $\quad \{ \bar{\Psi}^{(t)}_{f(i)} \mid t,i \in N \}$ is decidable.

Then one can find ρ, $\kappa \in R_1$, $\gamma \in R_2$ such that $(\lambda p,x)(\rho[\bar{\Psi},\kappa]\gamma(p,x))$ is a Gödel numbering.

Remark: In addition to the necessary condition ($\bar{\Psi}1$) we need some assumption on the effective constructibility of the "tracks" to obtain sufficient conditions for "universal" $\bar{\Psi}$.

Until now we looked to the possible ρ, $\bar{\Psi}$, κ, γ separately. What would of course be most interesting is the interplay of $\bar{\Psi}$ and κ for universal $\bar{\Psi}$, κ. We give some necessary conditions. For this we introduce the following

Definition 2.1: Let $\xi_1,\xi_2 \in N$.

$$\xi_1 \overset{\bar{\Psi},\kappa}{\sim} \xi_2 :\leftrightarrow (Et_1,t_2)(\bar{\Psi}^{(t_1)}(\xi_1) = \bar{\Psi}^{(t_2)}(\xi_2) \;\&$$
$$(\tau < t_1)(\kappa\bar{\Psi}^{(\tau)}(\xi_1) \neq 0) \;\&$$
$$(\tau < t_2)(\kappa\bar{\Psi}^{(\tau)}(\xi_2) \neq 0)).$$

Remark: $\overset{\bar{\Psi},\kappa}{\sim}$ is an equivalence relation on N. (Even for universal $\bar{\Psi}$, κ it may happen that this relation is decidable, see example after Theorem 2.2).

Notation:

$[\xi]$... equivalence class of ξ with respect to $\overset{\bar{\Psi},\kappa}{\sim}$. In the notation $[\xi]$ we don't make any reference to the $\bar{\Psi}$, κ used any more since, in the

following, no confusion will arise. Also, we trust that the two-fold use of the brackets in $[\xi]$ and $[\bar{\psi},\kappa]$ will not trouble the reader.

$[N]$... set of all equivalence classes with respect to $\bar{\psi},\kappa$.

Remarks: All $[\xi]$ are recursively enumerable. There are examples (see example after Theorem 2.2) where all $[\xi]$ are even recursive. $[N]$ divides into three disjoint subsets:

$[N] = [N_t] \cup [N_c] \cup [N_\infty]$ where

$[N_t] := \{ [\xi] \mid (E\xi')(\xi' \in [\xi] \;\&\; \kappa(\xi') = o) \}$,

$[N_c] := \{ [\xi] \mid [\xi] \notin [N_t] \;\&\; (E\xi', t)(\xi' \in [\xi] \;\&\; t \neq o \;\&\; \bar{\psi}^{(t)}(\xi') = \xi') \}$,

$[N_\infty] = [N] - [N_t] - [N_c]$, i.e.

$[N_t]$ consists of all $[\xi]$ that contain terminal states ξ', $[N_c]$ consists of all $[\xi]$ that contain "cycles" and $[N_\infty]$ collects the remaining $[\xi]$. Although one's experience with the usual programming languages might not suggest it, there are examples of universal $\bar{\psi}$, κ where $[N_c] = \emptyset$ (see example after Theorem 2.2).

Definition 2.2: Let $[X] \subset [N]$.
A function f is $[X]$-generating $:\Longleftrightarrow [X] = \{ [f(i)] \mid i \in N \}$.

Remarks: For $\bar{\psi}$, $\kappa \in R_1$ there exist f_t, $f_c \in R_1$ such that f_t is $[N_f]$-generating and f_c is $[N_c]$-generating.

Theorem 2.13: Let $\bar{\Psi}$, κ be universal. Then

($\bar{\Psi}$,κ1) there does not exist a recursive $[N_\infty]$-generating function

($\bar{\Psi}$,κ2) for all n:

there exist infinitely many distinct equivalence classes $[\xi_0]$, $[\xi_1]$, ... in $[N_t]$ such that

$$(i)(E\xi')(\xi' \in [\xi_i] \;\&\; (\tau<n)(\bar{\Psi}^{(\tau)}(\xi_i) \in [\xi_i])).$$

Remarks: ($\bar{\Psi}$,κ1) is easily deduced from Rice's Theorem (see, for instance, ROGERS 67), ($\bar{\Psi}$,κ2) needs simple results of Blum's complexity theory which, by Theorem 2.2, are available for our investigation. Intuitively ($\bar{\Psi}$,κ2) says that $\bar{\Psi}$ must contain enough "tracks" of every length that lead to a terminal state.

We have a strong feeling that for $\bar{\Psi}$, $\kappa \in R_1$ the conditions ($\bar{\Psi}$,κ1) and (an effective version of) ($\bar{\Psi}$,κ2) are also sufficient for guaranteeing the universality of $\bar{\Psi}$, κ. However, we have not been able to prove this.

Finally, we want to give an interesting result on the reducibility of universality to normal universality.

Theorem 2.14: Let f be $[N_\infty]$-generating and $\bar{\Psi}$, κ universal. Then one can find ρ, $\gamma \in R_1$ and an f-pairing function τ such that

$(\lambda p,x)(\rho\tau_2[\bar{\Psi},\kappa]\tau(p,\gamma(x)))$ is a Gödel numbering, and

$\bar{\Psi}\tau(p,n) = \tau(p,\bar{\Psi}(p,n))$ for $\bar{\Psi}(p,n) := \tau_2\bar{\Psi}\tau(p,n)$.

Remarks: By an f-pairing function we mean a 1-1, binary, f-recursive function τ for which $\tau(N,N)$ is f-recursive. Thus, Theorem 2.14 says that the states of a "universal automaton" $\bar{\Psi}$, κ can always be decomposed in such a way that one component (which plays the role of the "program") is not altered during execution. Unfortunately, Theorem 2.4 is not constructive in that the τ obtained depends on the non-recursive f (see Theorem 2.13). It is open whether one could find a constructive version of this theorem.

3. Future problems

A further investigation on the topics given in this note will center around the following problems:

1. A further detailed study of universal $\bar{\Psi}$, κ, especially of the questions left open by the Theorems 2.11 - 2.14. Construction of simple examples of universal $\bar{\Psi}$, κ.

2. An exact definition of the concepts "compilation", "simulation", "dependence of syntax on semantics" etc. in terms of the concept of an automaton based on the recursion scheme (CI).

3. Study of the whole hierarchy of possible notions of universal functions between the general notion of a Gödel numbering and our notion of universal $\bar{\Psi}$, κ and such notions which may be obtained by further specializing our notion.

4. Study of automata whose basic action principle is a more complicated recursion scheme than (C1), e.g. such that they could "directly" function according to a definition like that of "apply" in McCARTHY 62. Such automata would have to consist of infinitely many universal automata. Thus the cellular automata in CODD 68 would probably not suffice. Also Gilmore's "computer with a LISP-like machine language" is not such an automaton. However, a detailed examination of Gilmore's work reveals that his computer essentially functions still according to our recursion scheme (C1).

5. Recently it has been pointed out (MOSCHOVAKIS 71, FENSTAD 72) that generalized recursive function theories should include notions like "computation", "length of computation", "subcomputations", "computational steps" etc. as primitive notions. We think that a better understanding of the power contained in one step of an ordinary computability formalism (reflected by the power of $\bar{\Psi}$ in our terminology) could help in reasonably axiomatizing these notions.

Acknowledgement: Most of the material in Section 2. was obtained by a constant and most pleasing collaboration with Dr.B.Rolder (Univ.Innsbruck). To him I want to express my sincere gratitude.

R e f e r e n c e s :

ASSER 60, G.Asser, Rekursive Wortfunktionen, Zeitschrift für mathematische Logik und Grundlagen der Mathematik 6, pp.258-278.

BEKIĆ/WALK 71, H.Bekić, K.Walk, Formalization of storage properties, in: Symposium on Semantics of Algorithmic Languages (E.Engeler ed.), Springer Lecture Notes 188, 1971.

BLUM 67, M.Blum, A machine-independent theory of the complexity of recursive functions, J.ACM 14/2, 1967.

BUCHBERGER 71, B.Buchberger, Associating functions and the operator of conditioned iteration (Russian), Communications of the JINR Dubna, P5-5788, (English translation: Bericht Nr.71-1, Inst.f.num. Math., Univ.Innsbruck, 1971).

BUCHBERGER 72, B.Buchberger, A basic problem in the theory of programming languages, Bericht Nr.72-1, Inst.f.num.Math., Univ.Innsbruck, 1972.

BUCHBERGER/ROIDER 72, B.Buchberger, B.Roider, A study on universal functions, Bericht Nr.72-5, Inst.f.num.Math., Univ.Innsbruck, to appear.

CODD 68, E.F.Codd, Cellular automata, Academic Press, 1968.

FENSTAD 72, J.E.Fenstad, On axioms for computation theories, Lecture given at the conference on mathematical logic, Oberwolfach, Germany, April 1972.

GILMORE 67, P.C.Gilmore, An abstract computer with a LISP-like machine language, in: Computer Programming and Formal Systems (P.Braffort/D.Hirschberg ed.), North-Holland, 1967.

LUCAS/LAUER/STIGLEITNER 68, P.Lucas, P.Lauer, H.Stigleitner, Method and notation for the formal definition of programming languages, TR 25.o87, IBM Laboratory Vienna, 1968.

McCARTHY 62, J.McCarthy, LISP 1.5 programmer's manual, MIT Press, 1962.

McCARTHY/PAINTER 67, J.McCarthy, J.Painter, Correctness of a compiler for arithmetic expressions, Proc.of Symp.in Applied Math. 19 (J.T. Schwartz ed.), Amer.Math.Soc., pp.33-41.

MARKOV 47, A.A.Markov, On the representation of recursive functions (Russian), Doklady Ak.Nauk SSSR, n.s.58, pp.1891-1892.

MOSCHOVAKIS 71, Axioms for computation theories - first draft, in: Logic Colloquium '69 (R.O.Gandy, C.M.E.Yates ed.), North-Holland, 1971.

ROGERS 58, H.Rogers, Jr., Gödel numberings of partial recursive functions, J.Symbolic Logic 23/3, pp.331-341, 1958.

ROGERS 67, H.Rogers, Jr., Theory of recursive functions and effective computability, McGraw-Hill 1967.

SCHWENKEL 66, F.Schwenkel, Semantische Theorie der Programmiersprachen, Dissertation, Univ.Tübingen, Germany, 1966.

SCOTT 67, D.Scott, Some definitional suggestions for automata theory, J.Comp.System Sci. 1, pp.187-212, 1967.

SCOTT 71, D.Scott, The lattice of flow diagrams, in: Symposium on Semantics of Algorithmic Languages (E.Engeler ed.), Springer Lecture Notes 188, 1971.

SHOENFIELD 71, J.R.Shoenfield, Degees of unsolvability, North-Holland/ American Elsevier, 1971.

USPENSKII 56, V.A.Uspenskii, Computable operators and the notion of program (Russian), Uspehi Mat.Nauk 11/4, pp.172-176.

USPENSKII 6o, V.A.Uspenskii, Lectures on computable functions (Russian), Gos.Izd.Fiz.Mat.Lit., Moscow, 196o.