stichting
mathematisch
centrum

AFDELING ZUIVERE WISKUNDE ZW 44/75 JUNE

T.M.V. JANSSEN

AN ARITHMETIZATION OF VAN WIJNGAARDEN GRAMMAR

2e boerhaavestraat 49 amsterdam

Printed at the Mathematical Centre, 49, 2e Boerhaavestraat, Amsterdam.

The Mathematical Centre, founded the 11-th of February 1946, is a non-profit institution aiming at the promotion of pure mathematics and its applications. It is sponsored by the Netherlands Government through the Netherlands Organization for the Advancement of Pure Research (Z.W.O), by the Municipality of Amsterdam, by the University of Amsterdam, by the Free University at Amsterdam, and by industries.

AMS(MOS) subject classification scheme (1970): 68A30, 02F43, 02F99

An Arithmetization of van Wijngaarden Grammar

by

T.M.V. Janssen

ABSTRACT

An arithmetization is given of the concept of a van Wijngaarden grammar, by means of which it is proved that each van Wijngaarden grammar generates a recursively enumerable language.

KEYWORDS & PHRASES: *van Wijngaarden grammar, arithmetization, generative capacity, recursively enumerable languages.*

INTRODUCTION

Every recursively enumerable language can be generated by a van Wijngaarden grammar. This has been proved by SINTZOFF [9]; an elegant proof has been given by VAN WIJNGAARDEN [12] who showed that only one metanotion is needed to do so. The present report contains a formal proof of the reverse that each van Wijngaarden grammar generates a recursively enumerable language; no such proof has yet been given, as this fact is belied to be obvious by Church's thesis. The method of our proof is sketched as follows.

A van Wijngaarden grammar concerns strings of symbols; it generates strings of special symbols and it gives rise to relations between strings (such as string s_1 can be obtained from string s_2 by applying some rule of the grammar). The problem we consider, is the question whether a set of special strings is recursively enumerable. We shall, however, not directly deal with this problem, but rather with a translation of it. We use a 1-1 encoding of strings of symbols in numbers. Relations between strings, become relations between numbers, and the problem becomes the question whether a set of special numbers is recursively enumerable. By solving the translated problem we have indirectly solved the original problem since the encoding is 1-1. This method, which is called arithmetization, is well known in recursion theory.

We actually will obtain the result by defining a predicate *wordgeneration* (X,w,d), where X is the code number of a van Wijngaarden grammar, w is the code number of a string and d encodes all the information about the derivation of w. From the definition of this predicate it will become clear that this predicate is primitive recursive. Apparently $w \in L(X)$ iff $\exists d[$*wordgeneration*$(X,w,d)]$, thus L(X) is recursively enumerable. The predicate *wordgeneration* can actually be shown to be elementary recursive; this shows that the arithmetization is of the same complexity as the well known arithmetizations of Turing machines and of Markov algorithms.

FROM FORMAL LANGUAGE THEORY

We recall the following from formal language theory; for details see HOPCROFT & ULLMAN [6]. Let V be a finite set. Then V^+ is the set of all non empty strings consisting of elements from V, thus $V^+ = \{v_1 \ldots v_n \mid n \geq 1, v_i \in V\}$. We indicate the empty string with λ and put $V^* = V^+ \cup \{\lambda\}$. For $w \in V^*$, let $|w|$ denote the length of w.

We designate a *Chomsky grammar* (shortly grammar) by $G = (V,\Sigma,P,S)$ where V is a finite set, $\Sigma \subset V$, $S \in V-\Sigma$ and $P \subset V^+ \times V^*$ is a finite set: the rules or productions. If there are no restrictions on P, then G is of Chomsky type 0 or shortly type-0. If $|u| \leq |v|$ for each $(u,v) \in P$ then G is context-sensitive. If $u \in V-\Sigma$ for each $(u,v) \in P$ then G is context-free. If $(u,v) \in P$ and $x,y \in V^*$ then $xuy \underset{G}{\Rightarrow} xvy$. We denote by $\overset{*}{\underset{G}{\Rightarrow}}$ the transitive, reflexive closure of $\underset{G}{\Rightarrow}$. We omit subscripts G when no confusion is likely. The *language generated by* G is $L(G) = \{w \in \Sigma^* \mid S \overset{*}{\Rightarrow} w\}$.

A *λ-rule* is a rule $(u,v) \in P$ with $v = \lambda$.

A *homomorphism* h is a mapping $V^* \to W^*$, where V and W are finite sets, and $h(xy) = h(x)h(y)$ for all $x,y \in V^*$.

FROM RECURSION THEORY

We recall the following facts from recursion theory; for details and proofs see DAVIS [3].

(1) The operation *composition* associates with the functions
$\lambda y_1 \ldots y_m\ f(y_1,\ldots,y_m),\ \lambda x_1 \ldots x_n\ g(x_1,\ldots,x_n),\ \ldots\ ,\ \lambda x_1 \ldots x_n\ g(x_1,\ldots,x_n)$
the function

$$\lambda x_1 \ldots x_n\ f(g(x_1,\ldots,x_n),\ \ldots\ ,g(x_1,\ldots,x_n)).$$

(2) The operation of *primitive recursion* associates with the given total functions $\lambda x_1 \ldots x_n\ f(x_1,\ldots,x_n)$ and $\lambda zyx_1 \ldots x_n\ g(z,y,x_1,\ldots,x_n)$ the function $\lambda zx_1 \ldots x_n\ h(z,x_1,\ldots,x_n)$ where for all $z,x_1,\ldots,x_n$ the following holds:

$$h(0,x_1,\dots,x_n) = f(x_1,\dots,x_n) ,$$
$$h(z+1,x_1,\dots,x_n) = g(z,h(z,x_1,\dots,x_n),x_1,\dots,x_n).$$

(3) A function is called a *primitive recursive function* if it can be obtained by a finite number of applications of the operations of composition and primitive recursion, beginning with functions from the following list:

(i) $\lambda x(x+1)$,

(ii) $\lambda x(0)$,

(iii) $\lambda x_1\dots x_n(x_i)$.

(4) The following functions are primitive recursive

$$\lambda xy(x+y) \quad \lambda xy(xy) \quad \lambda xy(x^y).$$

(5) An n-place predicate P is called a *primitive recursive predicate* if the characteristic function of the set $\{(x_1,\dots,x_n) \mid P(x_1,\dots,x_n)\}$ is primitive recursive.

(6) If P and Q are primitive recursive predicates, then so are the conjunction $P \wedge Q$, the disjunction $P \vee Q$, the implication $P \rightarrow Q$ and the negated predicate $\neg P$.

(7) The predicates $<,\leq,>,\geq,=,\neq$ are primitive recursive.

(8) If $P(y,x_1,\dots,x_n)$ is a primitive recursive predicate, then the predicates

$$R(z_1,z_2,x_1,\dots,x_n)\text{, meaning } \exists y[(z_1 \leq y \leq z_2 \wedge P(y,x_1,\dots,x_n)]$$

and

$$S(z_1,z_2,x_1,\dots,x_n)\text{, meaning } \forall y[(z_1 \leq y \leq z_2 \rightarrow P(y,x_1,\dots,x_n)]$$

are again primitive recursive.

The usual notation for these two predicates are

$$\overset{z_2}{\underset{z_1}{\exists y}}\ P(y,x_1,\dots,x_n)$$

and

$$\overset{z_2}{\underset{z_1}{\forall y}}\ P(y,x_1,\dots,x_n)$$

respectively.

(9) We use the following definition of recursively enumerable set (this definition is equivalent to the one given by DAVIS [3]):
A *set* S is called *recursively enumerable* if there exists a primitive recursive predicate $P(x,y)$ such that $S = \{x \mid \exists y\ P(x,y)\}$.

CODING

In the sequel we encode sequences of natural numbers in natural numbers; therefore, we need a primitive recursive function from $\bigcup_{n=0}^{\infty} \mathbb{N}^n$ to $\mathbb{N}$. For convenience we require this function to be onto. The value of the function for the sequence $x_1,\ldots,x_n$ is denoted by $\langle x_1,\ldots,x_n\rangle$. The value for the empty sequence is denoted by $\langle\ \rangle$. We require this function to have the property that for all $1 \le i \le j \le n$ holds:

$$\langle x_i,\ldots,x_j\rangle \le \langle x_1,\ldots,x_i,\ldots,x_j,\ldots,x_n\rangle.$$

We require the following functions and predicate to be primitive recursive:

the function L , which gives the length of an encoded sequence; thus $L(\langle x_1,\ldots,x_n\rangle) = n$;
the function $(..)_i$, which gives the i-th element of a sequence; thus $(\langle x_1,\ldots,x_n\rangle)_i = x_i$ for $1 \le i \le n$;
the function $*$, which gives the code number of the concatenation of two sequences; thus $x * y = \langle (x)_1,\ldots,(x)_{L(x)}, (y)_1,\ldots,(y)_{L(y)}\rangle$;
the predicate *Elem*, where *Elem*(X,x) holds iff for some i the i-th element in the sequence encoded by X, equals x.

We define the primitive recursive function $\lambda xy \cdot it(x,y)$ by

$$\begin{cases} it(x,0) = \langle\ \rangle \\ it(x,n+1) = it(x,n) * (\langle x\rangle). \end{cases}$$

The primitive recursive function $\lambda x\ It(x)$ is now defined by $It(x) = it(x,L(x))$.
The function It has the following usefull property. Let $a = \langle a_1,\ldots,a_n\rangle$, $b = \langle b_1,\ldots,b_m\rangle$, where $m \le n$, and let $b_1,\ldots$ and b_m be equal to subsequences of $a_1,\ldots,a_n$ (with $b_i = \langle\ \rangle$ for at most one i). Then $b \le It(a)$. We will use this estimate to bound quantifications.

There are several codings which have the desired properties and for which the required functions and predicate are primitive recursive. For example, see TROELSTRA [10] (proofs concerning the properties of this code are found in §2 of KREISEL & TROELSTRA [7]) or VAN WIJNGAARDEN [13]. The actual choice of this code is, for our aims, arbitrarily. Even the use of an onto coding is not essential (for a code which is not onto see DAVIS [3]), but then one needs a primitive recursive predicate which tells whether a number is the code number of a sequence, and the definitions of the functions and predicate have to be adjusted in order to remain total.

DEFINITIONS CONCERNING VAN WIJNGAARDEN GRAMMARS

A *van Wijngaarden grammar* (W-grammar) is an ordered sixtuple $G = (V_M, \Pi, \Sigma, P_M, P_H, S)$ where the following holds:

V_M is a finite set, the elements of which are called *metavariables*;
Σ is a finite set, the elements of which are called *terminal symbols* or *terminals*;
Π is a finite set, the elements of which are called *protovariables*;
S is a special symbol in Π, called the *start symbol.*

We require V_M, Σ and Π to be disjoint. Let V_P denote $\Sigma \cup \Pi$, and let V denote $V_P \cup V_M$. The elements of V are called the *symbols of the grammar.* Let $<$ and $>$ be two symbols not in V. The set of *hypernotions* H is defined by $H := \{<\alpha> \mid \alpha \in V^+\}$.

$P_M \subset V^+ \times V^*$ is a finite set, the elements of which are called *metarules*;
$P_H \subset (H \cup \Pi) \times (H \cup V)^+$ is a finite set, the set of *hyperrules* or *rule schemata.*

We notice that for each $M \in V_M$ the grammar $G_M := (V_M, V_P, P_M, M)$ is a type-0 grammar. The grammar G_M is called a *metagrammar* of G. If $u \underset{G_M}{\Rightarrow} v$, then we say that v is a *direct metaproduction* of u, and if $u \underset{G_M}{\overset{*}{\Rightarrow}} v$, then that v is a *metaproduction* of v. If $M \underset{G_M}{\overset{*}{\Rightarrow}} v$, where M is the startsymbol of G_M and $v \in V_P^*$, then we call v a *terminal metaproduction* of M.

A *metaassignment* h is a homomorphism $h: (V \cup \{<,>\})^* \to (V_P \cup \{<,>\})^*$ such that $h(<) = <$, $h(>) = >$, $h(P) = P$ for $P \in V_P$ and $h(M) \in L(G_M)$ for $M \in V_M$.

Thus h(M) is a terminal metaproduction of the metanotion M.
The set of *production rules* of G, denoted by $Prod_G$, is defined by

$$Prod_G := \{(a,b) \mid \text{there is a hyperrule } (u,v) \text{ in } P_H, \text{ and there is a metaassignment } h \text{ with } h(u) = a \text{ and } h(v) = b\}.$$

Let $\alpha \underset{G}{\Rightarrow} \beta$ iff $\exists w_1,w_2,w_3,w_4 \in V_P^*$ such that $\alpha = w_1w_2w_3$, $\beta = w_1w_4w_3$ and $(w_2,w_4) \in Prod_G$. Again $\overset{*}{\underset{G}{\Rightarrow}}$ is the transitive, reflexive closure of $\underset{G}{\Rightarrow}$. We will omit subscript G when no confusion is likely. The *language generated by a W-grammar* G, denoted by L(G), is the set $\{w \in \Sigma^* \mid S \overset{*}{\Rightarrow} w\}$.

EXAMPLES

To illustrate the definitions concerning a van Wijngaarden grammar, we consider two examples. We will write metarules and hyperrules in the form $a \to b$ instead of (a,b).

Example 1. (Due to GREIBACH [4]).
Let $G = (V_M,\Pi,\Sigma,P_M,P_H,S)$ where

$V_M = \{N\}$, $\Pi = \{S\}$, $\Sigma = \{a\}$,

$P_M = \{N \to aN,\ N \to a\}$,

$P_H = \{S \to \langle a\rangle,\ \langle N\rangle \to \langle NN\rangle,\ \langle N\rangle \to N\}$.

Notice that $L(G_N) = \{a\}^+$. So any terminal metaproduction of N is of the form a^n, and any metaassignment h is of the form $h(N) = a^n$. Thus

$$Prod_G = \{S \to \langle a\rangle\} \cup \{\langle a^n\rangle \to \langle a^{2n}\rangle \mid n \geq 1\} \cup \{\langle a^n\rangle \to a^n \mid n \geq 1\}.$$

Consequently, all derivations of strings of L(G) are of the form

$$S \Rightarrow \langle a\rangle \Rightarrow \langle aa\rangle \Rightarrow \langle aaaa\rangle \Rightarrow \ldots \Rightarrow \langle a^{2n}\rangle \Rightarrow a^{2n}.$$

Example 2.
In order to make the role of the brackets < and > clearer, we change

example 1 a little. Let all sets except P_H remain the same, and let P_H be defined as follows:

$$P_H = \{S \to \langle a\rangle,\ \langle N\rangle \to \langle N\rangle\langle N\rangle,\ \langle N\rangle \to N\}.$$

Again, any metaassignment h is of the form $h(N) = a^n$. Thus

$$\mathrm{Prod}_G = \{S \to \langle a\rangle\} \cup \{\langle a^n\rangle \to \langle a^n\rangle\langle a^n\rangle \mid n \geq 1\} \cup \{\langle a^n\rangle \to a^n \mid n \geq 1\}.$$

Consequently, all derivations of strings of L(G) are now of the form

$$S \Rightarrow \langle a\rangle \Rightarrow \langle a\rangle\langle a\rangle \Rightarrow \langle a\rangle\langle a\rangle\langle a\rangle \Rightarrow \langle a\rangle \ \ldots\ \langle a\rangle \Rightarrow a^n.$$

We notice that the effect of using brackets is that the string between the brackets is considered as a whole, and that this string is kept together in the derivation.

REMARKS ON THE DEFINITION

There are several descriptions of what a van Wijngaarden grammar is. The definition we give is almost the same as that of GREIBACH [4]. The main point where our definition differs from other ones concerns the type of the grammars $G_M = (V_M, V_P, P_M, M)$. GREIBACH [4] requires each G_M to be a context-free grammar without λ-rules. The ALGOL 68 definition in VAN WIJNGAARDEN [11] has context-free metarules and EMPTY::. as the only λ-rule. Also DE CHASTELLIER & COLMERAURER [2] allow λ-rules among the context-free metarules. On the other hand, BAKER [1] requires that each G_M is a context-sensitive grammar, and allows λ-rules only under special conditions.

For our goal it does not make much difference which type of grammar we require for G_M. Therefore, we choose the most unrestricted one, i.e. type-0.

CODE NUMBER OF A VAN WIJNGAARDEN GRAMMAR

We define the following standard code for the symbols of a W-grammar: $\mathrm{code}(<) = 1$, $\mathrm{code}(>) = 2$.
Let $a_0, a_1, a_2, ..$ be an enumeration of Π, where S equals a_0. Then $\mathrm{code}(a_i) = 3i$.
Let $b_1, b_2, ..$ be an enumeration of V_M, then $\mathrm{code}(b_i) = 3i + 1$.
Let $c_1, c_2, ..$ be an enumeration of Σ, then $\mathrm{code}(c_i) = 3i + 2$.

Let E be the expression $s_1 s_2 ... s_n$ (thus E consists of the symbols $s_1, s_2, ..., s_n$ arranged in this order). The code number $\mathrm{cn}(E)$ of this expression is defined by $\mathrm{cn}(E) = \mathrm{cn}(s_1, ..., s_n) = \nless \mathrm{code}(s_1), \mathrm{code}(s_2), ..., \mathrm{code}(s_n) \ngtr$.

The code number of a metarule or hyperrule (u,v) is defined by

$$\mathrm{cn}((u,v)) = \nless \mathrm{cn}(u), \mathrm{cn}(v) \ngtr.$$

Example:
consider the hyperrule $(<S>, S<S>)$. The code number of this hyperrule is:

$$\mathrm{cn}((<S>\ S<S>)) = \nless \mathrm{cn}(<S>), \mathrm{cn}(S<S>) \ngtr =$$

$$= \nless \nless \mathrm{code}(<), \mathrm{code}(S), \mathrm{code}(>) \ngtr, \nless \mathrm{code}(S), \mathrm{code}(<), \mathrm{code}(S), \mathrm{code}(>) \ngtr \ngtr =$$

$$= \nless \nless 1,0,2 \ngtr, <1,0,1,2 \ngtr \ngtr.$$

The code number of a W-grammar with metarules $m_1, ..., m_k$ and hyperrules $h_1, ..., h_\ell$ is defined as

$$\mathrm{cn}(\text{W-grammar}) = \nless \nless \mathrm{cn}(m_1)_1, ..., \mathrm{cn}(m_k) \ngtr, \nless \mathrm{cn}(h_1)_1, ..., \mathrm{cn}(h_\ell) \ngtr \ngtr.$$

We note that each rearrangement of the sets Π, V_M, Σ and of P_M and P_H leads to a different code number for the same W-grammar. On the other hand: if a W-grammar G_1 has symbols which do not occur in the metarules or hyperrules, then G_1 may have the same code number as the W-grammar G_2 which is obtained from G_1 be deleting all these non-used symbols. The two grammars G_1 and G_2, however, are equivalent in the sense that they have exactly the same derivations and thus generate the same language.

PREDICATES CONCERNING THE STRUCTURE OF A VAN WIJNGAARDEN GRAMMAR

(1) $\mathit{open\ bracket}(x) \leftrightarrow x = 1;$

open bracket(x) holds iff x is the code of the open bracket <.

(2) $\mathit{close\ bracket}(x) \leftrightarrow x = 2;$

close bracket(x) holds iff x is the code of the close bracket >.

(3) $\mathit{start\ symbol}(x) \leftrightarrow x = 0;$

start symbol(x) holds iff x is the code of the start symbol.

(4) $\mathit{protovariable}(x) \leftrightarrow \overset{x}{\underset{0}{\exists}} i\ [x = 3i];$

protovariable(x) holds iff x is the code of some protovariable.

(5) $\mathit{metavariable}(x) \leftrightarrow \overset{x}{\underset{1}{\exists}} i\ [x = 3i+1];$

metavariable(x) holds iff x is the code of some metavariable.

(6) $\mathit{terminal}(x) \leftrightarrow \overset{x}{\underset{1}{\exists}} i\ [x = 3i+2];$

terminal(x) holds iff x is the code of some terminal symbol.

(7) $\mathit{symbol}(x) \leftrightarrow \mathit{protovariable}(x) \vee \mathit{metavariable}(x) \vee \mathit{terminal}(x);$

symbol(x) holds iff x is the code of some symbol of the grammar.

(8) $\mathit{mfree\ string}(x) \leftrightarrow \overset{L(x)}{\underset{1}{\forall}} i\ [\mathit{protovariable}((x)_i) \vee \mathit{terminal}((x)_i)];$

mfree string(x) holds iff x is the code number of a string which contains no metavariables, and no brackets.

(9) $\mathit{hypernotion}(x) \leftrightarrow \mathit{open\ bracket}((x)_1) \wedge \overset{L(x)-1}{\underset{2}{\forall}} i\ [\mathit{symb}((x)_i)] \wedge$

$\wedge\ \mathit{close\ bracket}((x)_{L(x)});$

hypernotion(x) holds iff x is the code number of some hypernotion.

(10) $\mathit{mrule}(x) \leftrightarrow L(x) = 2 \wedge \overset{L((x)_1)}{\underset{1}{\forall}} i\ [\mathit{symb}(((x)_1)_i)] \wedge$

$\wedge\ \overset{L((x)_2)}{\underset{1}{\forall}} i\ [\mathit{symb}(((x)_2)_i)] \wedge (x)_1 \neq \langle\ \rangle;$

mrule(x) holds iff x is the code number of some possible metarule.

(11) $hrule(x) \leftrightarrow L(x) = 2 \wedge ((L((x)_1) = 1 \wedge protovariable(((x)_1)_1)) \vee$

$\vee\ hypernotion((x)_1)) \wedge \overset{It((x)_1)}{\underset{0}{\exists a}}\ \overset{It((x)_1)}{\underset{0}{\exists b}} \Big[L(a) = L(b)+1 \wedge a_{L(a)} = (x_2) \wedge$

$\wedge\ (a)_1 = \nmid\ \nmid \wedge \overset{L(b)}{\underset{1}{\forall i}} [protovariable((b)_i) \vee hypernotion((b)_i)) \wedge$

$\wedge\ (a)_{i+1} = (a)_i * (b)_i] \Big];$

hrule(x) holds iff x is the code number of some possible hyperrule. Intuitively, *hrule*(x) implies that $(x)_2$ can be read as $(b)_1 * (b)_2 * \ldots * (b)_n$, where each $(b)_i$ is a protovariable or a hypernotion. The string $(a)_i$ equals $(b)_1 * \ldots * (b)_{i-1}$, thus $(a)_i$ is an initial substring of $(x)_2$, and $(a)_{L(a)}$ equals $(x)_2$.

Remark. It is not impossible that for the code number x of a metarule, also hrule(x) holds. However, this will not lead to an unintended use of the rule, since we shall only speak about metarules and hyperrules of a given W-grammar. Then we will always check whether x was given as a hyperrule or as a metarule.

(12) $W\text{-}grammar(X) \leftrightarrow L(X) = 2 \wedge \overset{(X)_1}{\underset{0}{\forall x}} [Elem((X)_1,x) \rightarrow mrule(x)] \wedge$

$\wedge\ \overset{(X)_2}{\underset{0}{\forall x}} [Elem((X)_2,x) \rightarrow hrule(x)];$

W-grammar(X) holds iff X is the code number of some W-grammar.

(13) $metarule(X,x) \leftrightarrow W\text{-}grammar(X) \wedge mrule(x) \wedge Elem((X)_1,x);$

metarule(X,x) holds iff metarule x is a metarule in W-grammar X.

(14) $hyperrule(X,x) \leftrightarrow W\text{-}grammar(X) \wedge hrule(x) \wedge Elem((X)_2,x);$

hyperrule(X,x) holds iff hyperrule x is a hyperrule in W-grammar X.

PREDICATES CONCERNING THE DERIVATIONAL PROCESS IN A VAN WIJNGAARDEN GRAMMAR

(1) *direct metaproduction*(X,u,v) $\leftrightarrow$

$$\overset{u}{\underset{0}{\exists}} u_1 \overset{u}{\underset{0}{\exists}} u_2 \overset{u}{\underset{0}{\exists}} u_3 \overset{v}{\underset{0}{\exists}} u_4 \; [u = u_1 * u_2 * u_3 \wedge v = u_1 * u_4 * u_3 * \wedge \mathit{metarule}(X, \nmid u_2, u_4 \nmid)];$$

direct metaproduction(X,u,v) holds iff v is a direct metaproduction of v.

(2) *metaproduction*(X,u,v,d) $\leftrightarrow$

$$u = (d)_1 \wedge v = (d)_2 \wedge \overset{L(d)}{\underset{2}{\forall}} i \; [\mathit{direct\ metaproduction}(X, (d)_{i-1}, (d)_i)];$$

metaproduction(X,u,v,d) holds iff v is a metaproduction of v and d encodes a derivation of v from u.

(3) *terminal metaproduction*(X,m,tm,d) $\leftrightarrow$ *metaproduction*(X,m,tm,d) $\wedge$

$$\wedge\ L(m) = 1 \wedge \mathit{metavariable}((m)_1) \wedge \mathit{mfree\ string}(d_{L(d)});$$

terminal metaproduction(X,m,tm,d) holds iff tm is a terminal metaproduction of m and d encodes a derivation of tm from m.

(4) *metasubstitution*(X,h,sh,m,tm,d) $\leftrightarrow$ *terminal metaproduction*(X,m,tm,d) $\wedge$

$$\wedge \overset{It(sh)}{\underset{0}{\exists}} b \Big[L(b) = L(h)+1 \wedge (b)_1 = \nmid \nmid \wedge$$
$$\wedge \overset{L(b)}{\underset{1}{\forall}} i \; [((h)_i = m \rightarrow (b)_{i+1} = (b)_i * tm) \wedge ((h)_i \neq m \rightarrow (b)_{i+1} = (b)_i * \nmid (h)_i \nmid)] \Big];$$

metasubstitution holds iff the string sh is obtained from string h by replacing each occurrence of metavariable m by its terminal metaproduction tm, where d codes the derivation of tm from m.

Intuitively this definition states that $b_1, b_2, \ldots$ represents the sequence of those initial substrings of sh, which are the result of substituting tm for m in all the initial substrings of h; so $b_{L(b)}$ = sh.

(5) *substitution sequence*(hseq,mseq,tmseq,dseq) $\leftrightarrow$

$$L(hseq) = L(mseq)+1 \wedge L(mseq) = L(tmseq) \wedge L(mseq) = L(dseq) \wedge$$
$$\wedge \overset{L(mseq)}{\underset{1}{\forall}} i \; [\mathit{metasubstitution}(X, (hseq)_i, (hseq)_{i+1}, (mseq)_i, (tmseq)_i, (dseq)_i)];$$

substitution sequence holds when in the string $(hseq)_1$ all occurrences of the metanotions $m_1, m_2, \ldots$ are replaced by their terminal metaproductions $tm_1, tm_2, \ldots$.

(6) *derivation of a rule*(X,l,r,lseq,rseq,mseq,tmseq,dseq) $\leftrightarrow$

substitution sequence(X,lseq,mseq,tmseq,dseq) $\wedge$

$\wedge$ *substitution sequence*(X,rseq,mseq,tmseq,dseq) $\wedge$

$\wedge$ *hyperrule*$(X, \nmid (lseq)_1, (rseq)_1 \nmid) \wedge (lseq)_{L(lseq)} = l \wedge$

$\wedge\ (rseq)_{L(rseq)} = r \wedge$ *mfree string*(l) $\wedge$ *mfree string*(r);

derivation of a rule holds when (l,r) is a productionrule of W-grammar X.

(7) *rule*(X,d) $\leftrightarrow$

$L(d) = 7 \wedge$ *derivation of a rule*$(X, (d)_1, (d)_2, (d)_3, (d)_4, (d)_5, (d)_6 (d)_7)$;

rule is an abbreviation of *derivation of a rule*.

(8) *derivationstep*(X,r,u,v) $\leftrightarrow$ *rule*(X,r) $\wedge$

$\wedge \overset{u}{\underset{0}{\exists}} u_1 \overset{u}{\underset{0}{\exists}} u_2 \overset{u}{\underset{0}{\exists}} u_3 \overset{v}{\underset{0}{\exists}} u_4 \, [u = u_1 * u_2 * u_3 \wedge v = u_1 * u_4 * u_3 \wedge u_2 = (r)_1 \wedge u_4 = (r)_2]$;

derivationstep(X,r,u,v) holds iff $u \Rightarrow v$ in W-grammar X, where r codes all information about the productionrule that is used.

(9) *derivation*(X,r,s) $\leftrightarrow$

$L(s) = L(r)+1 \wedge \overset{L(r)}{\underset{1}{\forall}} i \ [derivationstep(X, (r)_i, (s)_i, (s)_{i+1})] \wedge$

$\wedge\ L(s_1) = 1 \wedge startsymbol(((s)_1)_1) \wedge \overset{L(s)}{\underset{1}{\forall}} i \ [terminal(((s)_{L(s)})_i)]$;

derivation(X,r,s) holds iff $(s)_1 \overset{*}{\Rightarrow} (s)_{L(s)}$ in W-grammar X, the productionrules used are $(r)_1, (r)_2, \ldots, (r)_{L(r)}$; *startsymbol* $((s)_1)$ holds, and $(s)_{L(s)}$ is a string of terminals.

(10) $\mathit{wordgeneration}(X,d,w) \leftrightarrow$

$$L(d) = 2 \wedge \mathit{derivation}(X,(d)_1,(d)_2) \wedge ((d)_2)_{L((d)_2)} = w;$$

wordgeneration(X,d,w) holds iff d codes the derivation of word w in W-grammar X.

RESULT

All the predicates introduced up till now are primitive recursive. The next one is the only recursively enumerable predicate.

$\mathit{belongs\ to}(X,w) \leftrightarrow \exists d[\mathit{word\ generation}(X,w,d)];$

belongs to(X,w) holds iff $w \in L(X)$.

According to the given definition of recursively enumerable set, this means that L(X) is recursively enumerable. So we have proved: each van Wijngaarden grammar generates a recursively enumerable language.

We remark that, for a suitable choice of the encoding function $\nmid\ \nmid$, all predicates except the last are even elementary recursive (for a definition see e.g. GRZEGORCZYK [5]). This shows that the above arithmetization is not more complex in the sense of GRZEGORCZYK [5] than the well known arithmetization of Turing machines (see e.g. DAVIS [3]) or the arithmetization of Markov algorithms (see e.g. MENDELSON [8]).

REFERENCES

[1] J.L. BAKER, *Grammars with Structured Vocabulary: a Model for the ALGOL-68 Definition*, Inf. and Control 20 (1972) 351-395.

[2] G. DE CHASTELLIER & A. COLMERAUER, *W-Grammar*, Proc. ACM National Conference (1969) 511-518.

[3] M. DAVIS, *Computability & Unsolvability*, Mc Graw-Hill, New York (1958).

[4] S.A. GREIBACH, *Some Restrictions on W-Grammars*, Intern. J.Comput. Information Sci. 3 (1974) 289-327.

[5] A. GRZEGORCZYK, *Some classes of recursive functions*, Rozprawy matematyczne no. 4, Instytut Matematyczny Polskiej Akedemie Nauk, Warsaw.

[6] J.E. HOPCROFT & J.D. ULLMAN, *Formal Languages and their relation to Automata*, Addison-Wesley, Reading Mass. (1969).

[7] G. KREISEL & A.S. TROELSTRA, *Formal Systems for some branches of Intuitionistic Analysis*, Ann. of Math. Log. (1970) 229-387.

[8] E. MENDELSON, *Introduction to Mathematical Logic*, Van Nostrand, Princeton (1964).

[9] M. SINTZOFF, *Existence of a van Wijngaarden syntax for every recursively enumerable set*, Extr. Ann. Soc. Sci. Bruxelles T.81, II (1967) 115-118.

[10] A.S. TROELSTRA, (ed.), *Metamathematical Investigations of Intuitionistic Arithmetic and Analysis*, Springer, Berlin (1973).

[11] A. VAN WIJNGAARDEN, (ed.), *Report on the Algorithmic Language ALGOL 68*, Num. Math. 14 (1969) 79-218.

[12] A. VAN WIJNGAARDEN, *The Generative Power of Two-Level Grammars*, pp. 9-16, in: *Languages and Programming* (J. LOECKX (ed.)), 2nd colloquium University of Saarbrucken, 1974, Springer, Berlin (1974).

[13] A. VAN WIJNGAARDEN, *On the coding of sequences of natural numbers*, M.C. Report ZW 38, Mathematical Centre, Amsterdam.