INFORMATIQUE THÉORIQUE ET APPLICATIONS

VILIAM GEFFERT

Normal forms for phrase-structure grammars

Informatique théorique et applications, tome 25, n° 5 (1991), p. 473-496.

<http://www.numdam.org/item?id=ITA_1991__25_5_473_0>

Informatique théorique et Applications/Theoretical Informatics and Applications
(vol. 25, n° 5, 1991, p. 473 à 496)

NORMAL FORMS FOR PHRASE-STRUCTURE GRAMMARS (*)

by Viliam GEFFERT (1)

Abstract. – *Some new normal forms for the phrase-structure grammars are presented. Each phrase-structure grammar can be replaced by an equivalent grammar with all of the context free rules being of the from* $S \to v$, *where* S *is the initial nonterminal, what concerns non context free rules five different situations may occur: either two extra rules of the form* $AB \to \varepsilon$, $CD \to \varepsilon$, *or two extra rules* $AB \to \varepsilon$, $CC \to \varepsilon$, *or two extra rules* $AA \to \varepsilon$, $BBB \to \varepsilon$, *or even a single extra rule* $ABBBA \to \varepsilon$, *or a single extra rule* $ABC \to \varepsilon$. *In all cases, no additional nonterminal symbols are required.*

Résumé. – *Quelques nouvelles formes normales pour grammaires de type* 0 *sont présentées. Chaque langage récursivement énumérable peut être engendré par une grammaire où les règles « context-free » sont de forme* $S \to v$, *où* S *est le symbole initial non terminal. En ce qui concerne les règles « non context-free », on a l'une des cinq situations suivantes :ou bien deux règles du type* $AB \to \varepsilon$, $CD \to \varepsilon$, *ou deux règles du type* $AB \to \varepsilon$, $CC \to \varepsilon$, *ou deux règles du type* $AA \to \varepsilon$, $BBB \to \varepsilon$, *ou une règle du type* $ABBBA \to \varepsilon$, *ou règle du type* $ABC \to \varepsilon$. *Dans tous les cas aucun symbole non terminal additionnel n'est nécessaire.*

1. INTRODUCTION

Problems concerning normal forms of various devices generating or recognizing languages have turned out to be of crucial importance in the development of the formal language theory. Using normal forms, we shall obtain a mentally simpler manipulation with the phrase-structure grammars, while preserving their generative power. In spite of the fact that much research has been done on the comparison of many different models of grammars and automata, the central position of the context-free languages (and grammars) remains. One of the important advantages in dealing with the context-free grammars is the fact that each derivation can be represented by a derivation tree, in contrast to more powerful types of grammars. A similar characterization of the recursively enumerable languages would also be useful. It was

(*) Received 1990..
(1) University of P. J. Šafárik, Department of Computer Science, Jesenná 5, 04154 Košice, Czechoslovakia.

Informatique théorique et Applications/Theoretical Informatics and Applications
0988-3754/91/05 473 24/$4.40/© AFCET-Gauthier-Villars

shown in [13], that each phrase-structure grammar is equivalent to a grammar with rules in one of the following forms:

(i) $A \to v$, (ii) or $AB \to \varepsilon$.

Some similar results can also be found in [6], and [12]. We are going to establish a stronger result; namely, that the context-free rules can be of the form $S \to v$, where S is the initial nonterminal, using only two extra non-context-free rules $AB \to \varepsilon$, $CD \to \varepsilon$. Clearly, S, A, B, C, D are the only nonterminal symbols used by this type of grammar. Then we shall show that each recursively enumerable language may also be generated by a grammar having all of its rules context-free, of the form $S \to v$, and two extra rules $AB \to \varepsilon$, $CC \to \varepsilon$, or $AA \to \varepsilon$, $BBB \to \varepsilon$, or even a single extra rule; either $ABBBA \to \varepsilon$, or $ABC \to \varepsilon$. In all cases, no additional nonterminal symbols are used. These normal forms have already been announced in [2] and [3]. Problems concerning grammars with a single extra non-context-free rule of the form $AB \to \varepsilon$ (but using a slightly different approach, so-called Dyck_1-reductions of the context-free languages) can be found in [8].

The paper is organized as follows: We begin in the Section 2 by giving some basic definitions. This section also gives some auxiliary theorems concerning a characterization of the recursively enumerable languages by a pair of homomorphisms presented in [1], which is necessary to prove the main results. Section 3 proves the main theorem – the representation of the recursively enumerable languages by a grammar with only two non-context-free rules $AB \to \varepsilon$, $CD \to \varepsilon$. Section 4 concerns the other types of normal forms and Section 5 discusses the time and space complexities of grammars in these normal forms.

2. THE HOMOMORPHIC REPRESENTATION – A VARIANT OF THE POST CORRESPONDENCE PROBLEM

We are going to define the main notions here. (The reader is assumed to be familiar with the basic definitions and notation of formal language theory – this may be found in [5] or [7].) Then we shall establish a characterization of the recursively enumerable languages by a pair of homomorphisms [1], which is necessary to prove the main results. It is based on the notion of a g-system, introduced by Rovan [10, 11] in order to unify the theory of grammars. The g-system is a generalization of the notion of grammar, it is an iterated rewriting of the sentential form by a nondeterministic finite state 1-a-transducer [4]. (We could use the classical definition of the phrase-structure grammar as well, but using the g-systems, we shall obtain better

time and space complexities for the resulting phrase-structure grammars in normal form, since the g-systems give us a more natural correspondence between various types of devices and the phrase-structure grammar in normal form.)

DEFINITION: *A generative system* (*g-system*, for short) is a 4-tuple $G=(N, T, P, S)$, where N and T are finite aplhabets of nonterminal and terminal symbols, S in N is an initial symbol, and P represents a binary relation over $V^* \times V^*$ (where $V=N \cup T$). P is given in the form of a 1-a-transducer [4] (from V^+ to V^*), *i.e.*, $P=(K, V, V, H, q_I, q_F)$, where K is a finite set of states, q_I, q_F in K are initial and final states, respectively, and H is a finite subset of $K \times V \times V^* \times K$ (the set of *transitions*, or *edges*).

$u \in V^+$ is said to directly generate v, written $u \Rightarrow v$, if P is able to rewrite u to v, *i.e.*, there exists a path of transitions

$$(q_I, s_1, v_1, q_1)(q_1, s_2, v_2, q_2) \ldots (q_{n-1}, s_n, v_n, q_F) \in H^+,$$

such that $s_1 \ldots s_n = u$, and $v_1 \ldots v_n = v$.

Finally, *the language genrated by* G is the set $L(G)=\{ w \in T^*; S \Rightarrow^* w \}$, where $\Rightarrow^*$ is the transitive and reflexive closure of the relation $\Rightarrow$.

As is usual in the theory of grammars, G is said to be of time complexity $T(n)$ if, for each $w \in L(G)$ of length n, there exists a derivation of at most $T(n)$ steps generating w. Similarly, G is of space complexity $S(n)$ if, for each $w \in L(G)$ of length n, there is some derivation of w in which each sentential form is of length at most $S(n)$.

It can be easily seen that for each phrase-structure grammar G there exists a g-system G' such that $L(G')=L(G)$. Intuitively, each rule $A_1 \ldots A_n \rightarrow v$ in G will be replaced by a path of transitions in the 1-a-transducer of G' (from q_I to q_F), rewriting $A_1 \ldots A_n$ to v. (*See* [1] for the detailled proof.)

We shall now establish a characterization of the recursively enumerable languages by a pair of homomorphisms and a quotient [1]. (*The quotient* is understood as an operation inverse to concatenation, *i.e.*, $u \backslash uv = v$ for each u, v. This implies that $u \backslash v$ is defined only if u is a prefix of v.) The pair of homomorphisms h_1, h_2: $\Sigma_A^* \rightarrow \Sigma_B^*$ can be used to generate a language $L \subseteq \Sigma_L^*$ (where Σ_A, Σ_B, and Σ_L are some alphabets, $\Sigma_L \subseteq \Sigma_B$) as follows: For each string $\alpha \in \Sigma_A^+$, check if $h_1(\alpha) \backslash h_2(\alpha)$ is defined [*i.e.*, if $h_1(\alpha)$ is a prefix of $h_2(\alpha)$], and then if $h_1(\alpha) \backslash h_2(\alpha)$ is in Σ_L^*. If this is the case, then generate $w = h_1(\alpha) \backslash h_2(\alpha)$ to the output, it is a word in L.

THEOREM 1: *For each effectively given recursively enumerable language L one can effectively construct a pair of homomorphisms $h_1, h_2: \Sigma_A^* \to \Sigma_B^*$ such that*

$$L = \{ w \in \Sigma_L^*;\ w = h_1(\alpha) \backslash h_2(\alpha) \text{ for some } \alpha \in \Sigma_A^+ \}.$$

(Where Σ_A, Σ_B are some alphabets, $\Sigma_B \supseteq \Sigma_L$.) Thus, $w \in \Sigma_L^*$ is a word in L if and only if there exists some $\alpha \in \Sigma_A^+$ such that $h_2(\alpha) = h_1(\alpha) w$.

Construction and informal idea of the proof: The proof is based on the fact that g-systems are capable of generating any recursively enumerable language. The construction is similar to the construction of the Post Correspondence Problem imitating the computation of a Turing machine (and its halting problem). [9] But, instead of a Turing machine, we shall rather simulate the derivation of a word by a g-system generating the language L:

Let $L = L(G)$ for some g-system $G = (N, \Sigma_L, P, S)$, where P is a 1-a-transducer, *i.e.* $P = (K, V, V, H, q_I, q_F)$, where $V = N \cup \Sigma_L$. Define

$$\Sigma_A = \{ a_0, a_1, a_2, a_3 \} \cup K \times V \cup H,$$
$$\Sigma_B = \{ b_0, b_1, b_2 \} \cup K \cup V \cup K \times V,$$

where $a_0, a_1, a_2, a_3, b_0, b_1, b_2$ are new symbols. We now define h_1 and h_2:

TABLE

$x \in \Sigma_A$	$h_1(x)$	$h_2(x)$	Remark
a_0	b_0	$b_0 b_1 S$	
a_1	b_1	b_2	
a_2	$b_2 q_I$	$q_F b_1$	
a_3	b_1	ε	
(q, A)	A	$q(q, A)$	for $(q, a) \in K \times V$
(q, A, v, q')	$(q, A) q'$	v	$(q, A, v, q') \in H$

First, we shall briefly show that if $S \Rightarrow_G^* v$ then there exists $\alpha \in \Sigma_A^+$ such that

$$h_2(\alpha) = h_1(\alpha) b_1 v, \tag{2.1}$$

by induction on the length of a derivation in G:

A: If $v = S$ (the length of derivation is zero), then (2.1) holds for $\alpha = a_0$:

$$h_1(\alpha) = b_0$$
$$h_2(\alpha) = b_0 b_1 S$$

B: Now assume (2.1) holds for derivations of length k. Let $S \Rightarrow_G^* u \Rightarrow_G v$ be the derivation of length $k+1$. We have, by induction, $\alpha_1 \in \Sigma_A^+$ such that

$$\left.\begin{aligned} h_1(\alpha_1) &= y \\ h_2(\alpha_1) &= y b_1 u, \end{aligned}\right\} \tag{2.2}$$

for some $y \in \Sigma_B^*$. Since $u \Rightarrow_G v$, there exists a path of transitions $(k_1, s_1, v_1, k_1') \ldots (k_n, s_n, v_n, k_n')$ in H^+ such that

$$\left.\begin{aligned} u &= s_1 \ldots s_n, \\ v &= v_1 \ldots v_n, \\ k_1 &= q_I, \\ k_{i+1} = k_i', &\quad \text{for } i = 1, \ldots, n-1 \\ k_n' &= q_F. \end{aligned}\right\} \tag{2.3}$$

We can construct α by appending certain symbols to α_1. First, let $\alpha_2 = \alpha_1 a_1$. From (2.2) and (2.3) we have

$$\begin{aligned} \alpha_2 &= \alpha_1 a_1 \\ h_1(\alpha_2) &= y b_1 \\ h_2(\alpha_2) &= y b_1 \underbrace{s_1 \ldots s_n}_{u} b_2 \end{aligned}$$

Now we extend α_2 by $(k_1, s_1) \ldots (k_n, s_n)$:

$$\begin{aligned} \alpha_3 &= \alpha_1 a_1 (k_1, s_1) \ldots (k_n, s_n) \\ h_1(\alpha_3) &= y b_1 s_1 \ldots s_n \\ h_2(\alpha_3) &= y b_1 s_1 \ldots s_n b_2 k_1 (k_1, s_1) \ldots k_n (k_n, s_n) \end{aligned}$$

Next, we append the symbol a_2:

$$\begin{aligned} \alpha_4 &= \alpha_1 a_1 (k_1, s_1) \ldots (k_n, s_n) a_2 \\ h_1(\alpha_4) &= y b_1 s_1 \ldots s_n b_2 q_I \\ h_2(\alpha_4) &= y b_1 s_1 \ldots s_n b_2 k_1 (k_1, s_1) \ldots k_n (k_n, s_n) q_F b_1 \end{aligned}$$

$h_1(\alpha_4)$ remains a prefix of $h_2(\alpha_4)$, because $q_I = k_1$, by (2.3). Finally, let us append $(k_1, s_1, v_1, k_1') \ldots (k_n, s_n, v_n, k_n')$:

$$\begin{aligned} \alpha_5 &= \alpha_1 a_1 (k_1, s_1) \ldots (k_n, s_n) a_2 (k_1, s_1, v_1, k_1') \ldots (k_n, s_n, v_n, k_n') \\ h_1(\alpha_5) &= \ldots b_2 q_I (k_1, s_1) k_1' (k_2, s_2) k_2' \ldots (k_n, s_n) k_n' \\ h_2(\alpha_5) &= \ldots b_2 k_1 (k_1, s_1) k_2 (k_2, s_2) \ldots k_n (k_n, s_n) q_F b_1 v_1 v_2 \ldots v_n \end{aligned}$$

By (2.3) we have $k'_i = k_{i+1}$ for $i = 1, \ldots, n-1$, $k'_n = q_F$, and also $v = v_1 \ldots v_n$. Then the strings $\alpha = \alpha_5$ and $v = v_1 \ldots v_n$ again satisfy (2.1) and the claim is verified.

Thus, for each v such that $S \Rightarrow^* v$ we have $\alpha' \in \Sigma_A^+$ such that $h_2(\alpha') = h_1(\alpha') b_1 v$. Let $\alpha = \alpha' a_3$. Clearly $h_2(\alpha) = h_1(\alpha) v$, *i.e.* $v = h_1(\alpha) \backslash h_2(\alpha)$.

Moreover, we have also shown that, having a derivation $S = w_0 \Rightarrow w_1 \Rightarrow \ldots \Rightarrow w_k \Rightarrow w$, α can always be chosen in the form

$$a_0 \left(\prod_{i=0}^{k} (a_1 (K \times V)^{n_i} a_2 H^{n_i}) \right) a_3,$$

where $n_i = |w_i|$. Therefore, we can always find an α satisfying $w = h_1(\alpha) \backslash h_2(\alpha)$ such that

$$|\alpha| \leqq 2 + \sum_{i=0}^{k} (2 + 2|w_i|) \in O\left(\sum_{i=0}^{k} |w_i| \right). \tag{2.4}$$

We shall not detail the long technical verification showing that this is the only way of forming an α with the desired properties, since the complete formal proof can be found in [1]. □

Now, by suitably encoding the alphabet Σ_B into a two-letter alphabet, we shall establish a different version of Theorem 1 – a representation of recursively enumerable languages by a modification of the Post Correspondence Problem:

DEFINITION: Let $\Sigma_L = \{a_1, \ldots, a_{n_L}\}$ be an alphabet. *The Extended Post Correspondence* (*EPC*, for short) is

$$P = (\{(u_1, v_1), \ldots, (u_r, v_r)\}, (z_{a_1}, \ldots, z_{a_{n_L}})),$$

where $u_i, v_i, z_a \in \{0, 1\}^*$ for each $i = 1, \ldots, r$, and each $a \in \Sigma_L$.

The language represented by P in Σ_L^*, written $L(P)$, is the set:

$$L(P) = \{x_1 \ldots x_n \in \Sigma_L^*; \exists s_1, \ldots, s_l \in \{1, \ldots, r\}, \\ l \geqq 1 \text{ such that } v_{s_1} \ldots v_{s_l} = u_{s_1} \ldots u_{s_l} z_{x_1} \ldots z_{x_n}\}.$$

Note, that the classical Post Correspondence Problem [9] is to determine whether or not $\varepsilon \in L(P)$, where P is an *EPC* for $\Sigma_L = \varnothing$ [*i.e.* $L(P) \subseteq \varnothing^* = \{\varepsilon\}$].

THEOREM 2: *For each recursively enumerable language L there exists an Extended Post Correspondence P such that* $L(P) = L$.

Proof: By the use of Theorem 1, $w \in \Sigma_L^*$ is a word in L if and only if there exists an $\alpha \in \Sigma_A^+$ such that $h_2(\alpha) = h_1(\alpha) w$. (Where h_1, h_2 are some homomorphisms from Σ_A^* to Σ_B^*, $\Sigma_B \supseteq \Sigma_L$.)

If $\Sigma_A = \{e_1, \ldots, e_{n_A}\}$, $\Sigma_L = \{a_1, \ldots, a_{n_L}\}$, and $\Sigma_B = \{a_1, \ldots, a_{n_B}\}$, $n_B \geqq n_L$,

then $w = x_1 \ldots x_n \in \{a_1, \ldots, a_{n_L}\}^*$ is a word in L if and only if there exists an $\alpha = e_{s_1} \ldots e_{s_l} \in \{e_1, \ldots, e_{n_A}\}^+$ such that

$$h_2(e_{s_1}) \ldots h_2(e_{s_l}) = h_1(e_{s_1}) \ldots h_1(e_{s_1}) x_1 \ldots x_n. \tag{2.5}$$

Our next task is now to encode the symbols of Σ_B into the strings over a two-letter alphabet, *i.e.* to define a homomorphism $c: \Sigma_B^* \to \{0, 1\}^*$. Let $k = \lceil \log_2(n_B) \rceil + 1$, and

$$c(a_i) = b_k^i \ldots b_0^i, \quad \text{for each } i = 1, \ldots, n_B,$$

where $b_k^i \ldots b_0^i$ is the number i written in binary notation (with leading zeroes if needed). This encoding is unambiguous, *i.e.* $c(u) = c(v)$ if and only if $u = v$, for each $u, v \in \Sigma_B^*$. Now we define

$$P = (\{(u_1, v_1), \ldots, (u_{n_A}, v_{n_A})\}, (z_{a_1}, \ldots, z_{a_{n_L}})),$$

where

$$\begin{aligned} u_i &= c(h_1(e_i)), \\ v_i &= c(h_2(e_i)), \end{aligned} \quad \text{for each } i = 1, \ldots, n_A$$

$$z_{a_i} = c(a_i), \quad \text{for each } a_i \in \Sigma_L = \{a_1, \ldots, a_{n_L}\}.$$

Now, since c is unambiguous, the condition (2.5) is satisfied if and only if

$$c(h_2(e_{s_1})) \ldots c(h_2(e_{s_l})) = c(h_1(e_{s_1})) \ldots c(h_1(e_{s_l})) c(x_1) \ldots c(x_n),$$

and hence if and only if

$$v_{s_1} \ldots v_{s_l} = u_{s_1} \ldots u_{s_l} z_{x_1} \ldots z_{x_n}. \tag{2.6}$$

Thus, $w = x_1 \ldots x_n \in \Sigma_L^*$ is a word in L if and only if there exist $s_1, \ldots, s_l \in \{1, \ldots, n_A\}$, $l \geqq 1$ satisfying (2.6), which proves the theorem. □

We also have, by (2.4), the correspondence between time/space complexities of g-system and the extended Post correspondence: If $w \in L(G) = L(P)$ is generated by a derivation $S = w_0 \Rightarrow_G \ldots \Rightarrow_G w_k \Rightarrow_G w$, then we can find a

solution of P for w of length

$$l \leqq c \cdot \sum_{i=0}^{k} |w_i|, \tag{2.7}$$

for some constant c. Before passing to our main results, we need two more technical lemmas which will be applied later.

LEMMA 1: *Let $h_1, h_2 : \Sigma_A^* \to \Sigma_B^*$ be a pair of homomorphisms representing a language L (as it was shown in Theorem 1). Then, for any $\alpha \in \Sigma_A^*$ if $h_1(\alpha)$ is a prefix of $h_2(\alpha)$, then $h_1(\alpha) \backslash h_2(\alpha)$ does not contain a substring $xh_1(t)$, where $x \in \Sigma_L$, and $t \in \Sigma_A$ such that $h_2(t) = \varepsilon$.*

Proof: Suppose lemma does not hold for h_1, h_2 of Theorem 1. Then, there exists an $\alpha \in \Sigma_A^*$ such that $h_1(\alpha)$ is a prefix of $h_2(\alpha)$, and $w = h_1(\alpha) \backslash h_2(\alpha)$ does contain a substring $xh_1(t)$, for some $x \in \Sigma_L$ and for some $t \in \Sigma_A$ such that $h_2(t) = \varepsilon$.

Since $h_2(\alpha) = h_1(\alpha) w$, we may conclude that $h_2(\alpha)$ also contains a substring $xh_1(t)$. If $h_2(t) = \varepsilon$, then either $t = a_3$ or $t = (q, A, \varepsilon, q') \in H$, by Table in Theorem 1. Therefore $h_1(t) = b_1$, or $h_1(t) = (q, A) q'$.

But this implies that $h_2(\alpha)$ contains a substring xb_1, or $x(q, A) q'$, for some $x \in \Sigma_L \subseteq V$. A contradiction in either case, since the symbol b_1 must be preceded either by b_0 or by q_F in $h_2(\alpha)$, but never by $x \in V$ (*see* Table). Similarly, $(q, A) \in K \times V$ must be preceded by $q \in K$, but never by $x \in V$. □

LEMMA 2: *Let $P = (\{(u_1, v_1), \ldots, (u_r, v_r)\}, (z_{a_1}, \ldots, z_{a_{n_L}}))$ be an EPC representing a language L. Then, there is no loss of generality in assuming that for any $s_1, \ldots, s_l \in \{1, \ldots, r\}$ if $u_{s_1} \ldots u_{s_l}$ is a prefix of $v_{s_1} \ldots v_{s_l}$, then $u_{s_1} \ldots u_{s_l} \backslash v_{s_1} \ldots v_{s_l}$ does not contain a substring $z_x u_t$, where $x \in \Sigma_L$, and $t \in \{1, \ldots, r\}$ such that $v_t = \varepsilon$.*

Proof: We may assume, without loss of generality, that an *EPC* representing a language L has been designed by the use of Theorem 1 and 2. Now, the argument merely mirrors the proof of Lemma 1, since the symbols of Σ_B were unambiguously encoded by strings in $\{0, 1\}^*$. □

3. THE MAIN RESULT

We can now state and prove the main theorem. We shall show that any recursively enumerable language may be generated by a grammar having all of its rules context-free, of the form $S \to v$, where S is the initial nonterminal,

but two extra rules $AB \to \varepsilon$, $CD \to \varepsilon$. The proof is based on the representation by an extended Post correspondence: We have shown that for each recursively enumerable language L there exists an *EPC*

$$P = (\{(u_1, v_1), \ldots, (u_r, v_r)\}, (z_{a_1}, \ldots, z_{a_{n_L}}))$$

such that $L(P) = L$, *i. e.* $w = x_1 \ldots x_n \in \Sigma_L^*$ is a word in L if and only if there exists $s_1, \ldots, s_l \in \{1, \ldots, r\}$, $l \geqq 1$ such that

$$v_{s_1} \ldots v_{s_l} = u_{s_1} \ldots u_{s_l} z_{x_1} \ldots z_{x_n}.$$

Since u_i, v_i, $z_a \in \{0, 1\}^*$ for each $i = 1, \ldots, r$, and for each $a \in \Sigma_L$, we can use the following way of generating $w = x_1 \ldots x_n$:

1st stage:

$$S \Rightarrow z'^R_{x_n} S x_n \Rightarrow z'^R_{x_n} z'^R_{x_{n-1}} S x_{n-1} x_n \Rightarrow \ldots$$
$$\Rightarrow z'^R_{x_n} \ldots z'^R_{x_1} S x_1 \ldots x_n \Rightarrow z'^R_{x_n} \ldots z'^R_{x_1} A x_1 \ldots x_n$$

2nd stage:

$$\Rightarrow z'^R_{x_n} \ldots z'^R_{x_1} u'^R_{s_l} A v''_{s_l} x_1 \ldots x_n$$
$$\Rightarrow z'^R_{x_n} \ldots z'^R_{x_1} u'^R_{s_l} u'^R_{s_{l-1}} A v''_{s_{l-1}} v''_{s_l} x_1 \ldots x_n \Rightarrow \ldots$$
$$\Rightarrow z'^R_{x_n} \ldots z'^R_{x_1} u'^R_{s_l} \ldots u'^R_{s_2} A v''_{s_2} \ldots v''_{s_l} x_1 \ldots x_n$$
$$\Rightarrow z'^R_{x_n} \ldots z'^R_{x_1} u'^R_{s_l} \ldots u'^R_{s_1} v''_{s_1} \ldots v''_{s_l} x_1 \ldots x_n,$$

where z'_{x_i}, u'_{s_j}, v''_{s_j} denote strings over some new alphabets $\{0', 1'\}$, and $\{0'', 1''\}$ corresponding to z_{x_i}, u_{s_j}, v_{s_j}, respectively. Formally, define two auxiliary homomorphisms b', b'' by $b'(0) = 0'$, $b'(1) = 1'$, $b''(0) = 0''$, $b''(1) = 1''$. Then $u'_s = b'(u_s)$, $v''_s = b''(v_s)$, and $z'_x = b'(z_x)$, for each $s = 1, \ldots, r$, $x \in \Sigma_L$. [Similarly, φ', φ'' will be simplified notations for $b'(\varphi)$, $b''(\varphi)$, respectively, for each $\varphi \in \{0, 1\}^*$.]

Note that, the only thing we should be able to do in the 1st stage of derivation is to rewrite the symbol S to $z'^R_x S x$, for each $x \in \Sigma_L$ (and also to A in the last step). Similarly, the 2nd stage will be just repeated rewriting of A to $u'^R_s A v''_s$, for each $s = 1, \ldots, r$ (and to $u'^R_s v''_s$ in the last step). It should be clear that, for the first two stages, we need context-free rules only.

3rd stage:

$$z'^R_{x_n} \ldots z'^R_{x_1} u'^R_{s_l} \ldots u'^R_{s_1} v''_{s_1} \ldots v''_{s_l} x_1 \ldots x_n$$

$$= (u_{s_1} \ldots u_{s_l} z_{x_1} \ldots z_{x_n})'^R (v_{s_1} \ldots v_{s_l})'' x_1 \ldots x_n$$

$$= \varphi'^R_1 \varphi''_2 x_1 \ldots x_n \Rightarrow^* x_1 \ldots x_n$$

if and only if $\varphi_1 = \varphi_2$.
Now we have to check, whether or not our *EPC* has a solution for $x_1 \ldots x_n$ (or, more exactly, whether the sentential form we have generated represents a solution to P for $x_1 \ldots x_n$), and, if and only if we have found a solution (*i. e.* $\varphi_1 = \varphi_2$), we have to erase $\varphi'^R_1 \varphi''_2$, which gives the terminal string $w = x_1 \ldots x_n$. The 3rd stage is therefore a cancellation of substrings $0'0''$, $1'1''$ (by rules $0'0'' \to \varepsilon$, and $1'1'' \to \varepsilon$). Clearly, the only place where we can apply these rules is the "frontier" between φ'^R_1 and φ''_2, since $\varphi'^R_1 \in \{0', 1'\}^*$, $\varphi''_2 \in \{0'', 1''\}^*$, and $x_1 \ldots x_n \in \Sigma_L^*$. It now follows easily that $\varphi'^R_1 \varphi''_2 w \Rightarrow^* w$ if and only if $\varphi_1 = \varphi_2$. It is obvious, that this way of rewriting generates exactly the language $L(P)$, and we can now construct a phrase-structure grammar with six nonterminal symbols, namely S, A, $0'$, $0''$, $1'$, $1''$ and with only two non-context-free rules $0'0'' \to \varepsilon$, $1'1'' \to \varepsilon$, for each recursively enumerable language L.

We are now going to eliminate the symbol "A", and to replace it everywhere by "S". The only thing causing problems is that we can now use a 2nd stage rule (*i. e.* $S \to u'^R_s S v''_s$, for some $s = 1, \ldots, r$) before applying the last rule of the 1st stage ($S \to z'^R_{x_1} S x_1$). In the next theorem, it will be seen that if we violate the correct ordering, then we shall not be able to derive a terminal string in the 3rd stage of derivation, because the extended Post correspondence, constructed in Theorem 2, has some "nice" technical properties such that the condition $\varphi_1 = \varphi_2$ cannot be satisfied in this case. We are now ready for the main theorem:

THEOREM 3: *Each recursively enumerable language L can be generated by a phrase-structure grammar with 5 nonterminal symbols, using only context-free rules of the form $S \to v$, where S is the initial symbol, but two extra rules $AB \to \varepsilon$, $CD \to \varepsilon$.*

Proof: Let

$$P = (\{(u_1, v_1), \ldots, (u_r, v_r)\}, (z_{a_1}, \ldots, z_{a_{n_L}}))$$

be an *EPC* such that $L(P)=L$. We may also assume that P satisfies the conditions of Lemma 2. Define a grammar $G=(T, \Sigma_L, P, S)$, by

$$N=\{S, 0', 1', 0'', 1''\},$$
$$P=GEN \cup EPC \cup TEST,$$

where

$$GEN=\{S \to z_x'^R S x;\ x \in \Sigma_L\},$$

/generating terminal symbols in the 1st stage,

$$EPC=\{S \to u_s'^R S v_s'';\ s=1, \ldots, r\}$$

/generating a solution of P in the 2nd stage,

$$\cup \{S \to u_s'^R v_s'';\ s=1, \ldots, r\},$$

/terminating the 2nd stage,

$$TEST=\{0'\,0'' \to \varepsilon,\ 1'\,1'' \to \varepsilon\},$$

/checking and erasing the solution of P in the 3rd stage of derivation.

(Recall that u_s', v_s'', z_x' are simplified notations for the strings over $\{0', 1'\}$, $\{0'', 1''\}$, corresponding to u_s, v_s, z_x, respectively.) Let $x_1 \ldots x_n \in L$. Since $L=L(P)$, then there exist $s_1, \ldots, s_l \in \{1, \ldots, r\}$, $l \geqq 1$ such that $v_{s_1} \ldots v_{s_l} = u_{s_1} \ldots u_{s_l} z_{x_1} \ldots z_{x_n}$.

Then

$$S \Rightarrow_G^* z_{x_n}'^R \ldots z_{x_1}'^R S x_1 \ldots x_n$$

/using $S \to z_{x_i}'^R S x_i \in GEN$, for $i=n, \ldots, 1$,

$$\Rightarrow_G^* z_{x_n}'^R \ldots z_{x_1}'^R u_{s_l}'^R \ldots u_{s_2}'^R S v_{s_2}'' \ldots v_{s_l}'' x_1 \ldots x_n$$
$$\Rightarrow_G z_{x_n}'^R \ldots z_{x_1}'^R u_{s_l}'^R \ldots u_{s_1}'^R v_{s_1}'' \ldots v_{s_l}'' x_1 \ldots x_n$$

/using $S \to u_{s_i}'^R S v_{s_i}'' \in EPC$, for $i=l, \ldots, 2$, and $S \to u_{s_1}'^R v_{s_1}'' \in EPC$,

$$= \varphi_1'^R \varphi_2'' x_1 \ldots x_n \Rightarrow_G^* x_1 \ldots x_n$$

/using $0'\,0'' \to \varepsilon$, $1'\,1'' \to \varepsilon \in TEST$, since $\varphi_1 = \varphi_2$,
hence it follows that $x_1 \ldots x_n \in L(G)$.

Conversely, let $S \Rightarrow_G^* x_1 \ldots x_n \in \Sigma_L^*$. There are three ways of rewriting the symbol S in the sentential form:

$$S \Rightarrow z_x'^R S x,$$

for each $x \in \Sigma_L$, using rules in *GEN*,

$$S \Rightarrow u_s'^R S v_s'',$$
$$S \Rightarrow u_s'^R v_s'',$$

for each $s = 1, \ldots, r$, using rules in *EPC*.

Thus if $S \Rightarrow^* p S q$, then $p \in \{0', 1'\}^*$, and $q \in (\{0'', 1''\} \cup \Sigma_L)^*$. Using any rule in *TEST* is disabled until the symbol S disappears, since the sentential form does not contain any substring of the form $0'0''$, or $1'1''$. On the other hand, after using some $S \to u_s'^R v_s'' \in EPC$, there is no possibility to use any rule in *GEN* or *EPC*, since the sentential form does not contain the symbol S and the rules in *TEST* are not capable of generating it.

Then the derivation $S \Rightarrow_G^* x_1 \ldots x_n$ is of the form

$$S \quad \Rightarrow^* \quad p S q$$

/by rules in $GEN \cup EPC$,

$$\Rightarrow \quad p u_{s_1}'^R v_{s_1}'' q$$

/the symbol S is annihilated by an *EPC* rule, for some $s_1 \in \{1, \ldots, r\}$,

$$\Rightarrow^* \quad x_1 \ldots x_n$$

/by rules in *TEST*.

Since $p u_{s_1}'^R \in \{0', 1'\}^*$, $v_{s_1}'' q \in (\{0'', 1''\} \cup \Sigma_L)^*$, and the rules in *TEST* are able only to cancel the substrings $0'0''$, $1'1''$, we have $p u_{s_1}'^R = \underline{\varphi}'^R$, and $v_{s_1}'' q = \underline{\varphi}'' x_1 \ldots x_n$, for some $\underline{\varphi} \in \{0, 1\}^*$:

If there were a substring $x_i 0''$ in $v_{s_1}'' q$, for some $i = 1, \ldots, n$, then we would not be able to generate a terminal string, since there is no possibility to cancel the symbol $0''$ because we cannot generate the symbol $0'$ between x_i and $0''$. The same argument holds for $x_i 1''$. This implies that, before using $S \to z_{x_1}'^R S x_1$, we cannot apply any *EPC* rule rewriting S to $u_t'^R S v_t''$, for $v_t'' \neq \varepsilon$. Then the derivation must be of the form:

1st stage:

$$S \quad \Rightarrow^* \quad \varphi'^R S x_1 \ldots x_n$$

/using $S \to z_{x_i}'^R S x_i \in GEN$, or $S \to u_t'^R S v_t'' \in EPC$, for $v_t'' = \varepsilon$,

2nd stage:

$$\Rightarrow^* \quad \varphi'^R u_{s_l}'^R \ldots u_{s_2}'^R S v_{s_2}'' \ldots v_{s_l}'' x_1 \ldots x_n$$

$$\Rightarrow \quad \varphi'^R u_{s_l}'^R \ldots u_{s_1}'^R v_{s_1}'' \ldots v_{s_l}'' x_1 \ldots x_n$$

/using $S \to u_{s_i}'^R S v_{s_i}'' \in EPC$, and $S \to u_{s_1}'^R v_{s_1}'' \in EPC$, for some $s_1, \ldots, s_l \in \{1, \ldots, r\}$, $l \geqq 1$,

3rd stage:

$$\Rightarrow^* \quad x_1 \ldots x_n$$

/using $0'0'' \to \varepsilon$, $1'1'' \to \varepsilon \in TEST$.

Now we are going to elimate the use of *EPC* rules in the 1st stage. It is easy to verify that $v_{s_1} \ldots v_{s_l} = u_{s_1} \ldots u_{s_l} \varphi$, or, equivalently,

$$\varphi = u_{s_1} \ldots u_{s_l} \backslash v_{s_1} \ldots v_{s_l},$$

since we were able to derive a terminal string in the 3rd stage. If, in the first stage before applying $S \to z_{x_1}'^R S x_1 \in GEN$, we rewrote S to $u_t'^R S v_t''$, for $v_t'' = \varepsilon$ (and hence also $v_t = \varepsilon$), then φ'^R would contain a substring $u_t'^R z_{x_i}'^R$, for some $i \in \{1, \ldots, n\}$. But then φ would contain a substring $z_{x_i} u_t$.

Then we would have $s_1, \ldots, s_l \in \{1, \ldots, r\}$ such that

$$\varphi = u_{s_1} \ldots u_{s_l} \backslash v_{s_1} \ldots v_{s_l}$$

would contain a substring $z_{x_i} u_t$, for some $x_i \in \Sigma_L$, and some $t \in \{1, \ldots, r\}$ such that $v_t = \varepsilon$. But that would contradict our assumption that the *EPC* P we have used satisfies the conditions of Lemma 2. Consequently, using an *EPC* rule in the 1st stage of derivation, we obtain a sentential form from which we are not able to derive any terminal string in the 3rd stage. It thence appears that each derivation generating a terminal string must be of the form:

1st stage:

$$S \quad \Rightarrow^* \quad z_{x_n}'^R \ldots z_{x_1}'^R S x_1 \ldots x_n$$

/by rules $S \to z_{x_i}'^R S x_i \in GEN$,

2nd stage:

$$\Rightarrow^* \quad z'^R_{x_n} \ldots z'^R_{x_1} u'^R_{s_l} \ldots u'^R_{s_2} S v''_{s_2} \ldots v''_{s_l} x_1 \ldots x_n$$
$$\Rightarrow \quad z'^R_{x_n} \ldots z'^R_{x_1} u'^R_{s_l} \ldots u'^R_{s_1} v''_{s_1} \ldots v''_{s_l} x_1 \ldots x_n$$

/by rules $S \to u'^R_{s_i} S v''_{s_i} \in EPC$, and $S \to u'^R_{s_1} v''_{s_1} \in EPC$, for some $s_1, \ldots, s_l \in \{1, \ldots, r\}$, $l \geqq 1$,

3rd stage:

$$\Rightarrow^* \quad x_1 \ldots x_n$$

/by rules $0'0'' \to \varepsilon$, $1'1'' \to \varepsilon \in TEST$.

Then, since we were able to derive a terminal string in the 3rd stage, we get

$$v_{s_1} \ldots v_{s_l} = u_{s_1} \ldots u_{s_l} z_{x_1} \ldots z_{x_n},$$

and hence we have $s_1 \ldots s_l \in \{1, \ldots, r\}$, $l \geqq 1$ such that $s_1 \ldots s_l$ is a solution of P for $x_1 \ldots x_n$, but this holds only if $x_1 \ldots x_n \in L(P) = L$. □

It can be easily seen that several known results are simple consequences of this theorem. For example, any recursively enumerable set can be recognized by a nondeterministic 1-state machine having two 1-turn pushdown stores. We also obtain that each recursively enumerable language L can be expressed in the form $L = p(L_1)$, where L_1 is a deterministic linear language, and p is a cancellation of well-balanced parentheses subwords over alphabet consisting of two pairs of parentheses. We can use a cancellation of a palindrome prefix over two-letter alphabet as well. If we restrict p to be a polynomially-bounded erasing then we shall obtain a characterization of the class of *NP*-languages by deterministic linear (and hence also by context-free) languages.

4. SOME OTHER NORMAL FORMS

In this section, we shall be interested in a number of other normal forms which generate all the recursively enumerable languages. We shall consider several normal forms, but constructions will be given uniformly, by showing the transformations for putting the grammar exhibited in Theorem 3 into

these forms. Firstly, we present an "improved" version of Theorem 3, reducing the number of nonterminals by one, and simplifying one of the non-context-free rules:

THEOREM 4: *Each recursively enumerable language L can be generated by a phrase-structure grammar with 4 nonterminal symbols, using only context-free rules of the form $S \to v$, where S is the initial symbol, and two extra rules $AB \to \varepsilon$, $CC \to \varepsilon$.*

Proof: We may use Theorem 3 to construct a phrase-structure grammar $G=(N, T, P, S)$ generating a language L and then to encode the symbols $0'$, $1'$, $0''$, $1'' \in N$ by symbols $1, \emptyset, 0$. Formally, define a homomorphism $h: \{0', 1', 0'', 1''\}^* \to \{1, \emptyset, 0\}^*$, where:

$$h(0')=1\emptyset\emptyset, \qquad h(0'')=001,$$
$$h(1')=1\emptyset, \qquad h(1'')=01.$$

But then we have to modify also the grammar G, which gives $G'=(N', T, P', S)$, where

$$N'=\{S, 1, \emptyset, 0\},$$

/instead of $N=\{S, 0', 1', 0'', 1''\}$,

$$P'=GEN' \cup EPC' \cup TEST',$$

the rules are changed as follows:

$$GEN'=\{S \to h(z_x'^R)\, Sx;\ x \in \Sigma_L\},$$

/instead of $S \to z_x'^R\, Sx$,

$$EPC'=\{S \to h(u_s'^R)\, Sh(v_s'');\ s=1, \ldots, r\} \cup \{S \to h(u_s'^R)\, h(v_s'');\ s=1, \ldots, r\},$$

/instead of $S \to u_s'^R\, S\, v_s''$, $S \to u_s'^R\, v_s''$.

The cancellation of substrings $0'0''$, $1'1''$ in the 3rd stage of the derivation will now correspond to the cancellation of $1\emptyset\emptyset001$, and $1\emptyset01$, respectively. But

$$TEST'=\{\emptyset 0 \to \varepsilon,\ 11 \to \varepsilon\}$$

will do as well.

A: The first two stages of the derivation: If G rewrites the symbol S to $z'^R_x Sx$ by a rule in GEN, or to $u'^R_s S v''_s$ by a rule in EPC, then G' can rewrite S to $h(z'^R_x) S x$, or to $h(u'^R_s) S h(v''_s)$, by the corresponding rule in GEN' or EPC', respectively.

On the other hand, neither $TEST$ rules in G, nor $TEST'$ rules in G' are applicable, since the sentential forms contain the symbol S: In G, the sentential form structure is $S \Rightarrow^*_G pSqw \in \{0', 1'\}^* S \{0''1''\}^* \Sigma_L^*$, there are no substrings $0'0''$, $1'1''$ in $pSqw$. This corresponds to $S \Rightarrow^*_{G'} h(p) S h(q) w \in \{1\emptyset\emptyset, 1\emptyset\}^* S \{001, 01\}^* \Sigma_L^*$ in G', the sentential form does not contain any substrings $\emptyset 0$, or 11.

Hence, by a straightforward induction on the length of the derivation, we have that $S \Rightarrow^*_G p S qw$ if and only if $S \Rightarrow^*_{G'} h(p) S h(q) w$. Similarly, the annihilation of the symbol S itself will be carried out by rewriting S to $h(u'^R_s) h(v''_s)$, instead of $u'^R_s v''_s$.

B: The third stage of the derivation: It is obvious that if $\varphi'^R_1 \varphi''_2 w \Rightarrow^*_G w$, then $\varphi_1 = \varphi_2$, and also $h(\varphi'^R_1) h(\varphi''_2) w \Rightarrow^*_{G'} w$. The cancellation of zeroes by the rule $0'0'' \to \varepsilon$ (or cancellation of ones by $1'1'' \to \varepsilon$) corresponds to the cancellation of $1\emptyset\emptyset 001$ (or $1\emptyset 01$), respectively, by the use of $\emptyset 0 \to \varepsilon$, $11 \to \varepsilon \in TEST'$.

Now, consider the case in which no terminal string can be generated from $\varphi'^R_1 \varphi''_2 w$ in G, because $\varphi_1 \neq \varphi_2$. We shall show that neither can we derive a terminal string from $h(\varphi'^R_1) h(\varphi''_2) w$ in G'.

B1: If φ_1 and φ_2 differ in at least one bit (at the same position) then the derivation will be blocked after some steps, because we shall obtain different symbols at the "frontier" between $\{0', 1'\}^*$ and $\{0'', 1''\}^*$:

Either

$$\varphi'^R_1 \varphi''_2 w \Rightarrow^*_G \underline{\psi}'^R_1 0' 1'' \underline{\psi}''_2 w,$$

or

$$\varphi'^R_1 \varphi''_2 w \Rightarrow^*_G \underline{\psi}'^R_1 1' 0'' \underline{\psi}''_2 w.$$

There is only one possible derivation here, since there is always at most one substring that can be modified, by at most one rule. From now on, the derivation cannot continue (in either case), since there are no substrings $0'0''$, $1'1''$, or S in the sentential form.

In G', this corresponds to the only possible derivation, *i.e.*,

either

$$h(\varphi_1'^R)\,h(\varphi_2'')\,w \quad \Rightarrow_G^*, \quad h(\underline{\psi}_1'^R)\,1\emptyset\emptyset\;01\,h(\underline{\psi}_2'')\,w,$$

or

$$h(\varphi_1'^R)\,h(\varphi_2'')\,w \quad \Rightarrow_G^*, \quad h(\underline{\psi}_1'^R)\,1\emptyset\;001\,h(\underline{\psi}_2'')\,w,$$

respectively. There is possibility to continue, but the only possible step is either

$$h(\underline{\psi}_1'^R)\,1\emptyset\emptyset\,01\,h(\underline{\psi}_2'')\,w \quad \Rightarrow_G, \quad h(\underline{\psi}_1'^R)\,1\emptyset 1\,h(\underline{\psi}_2'')\,w \in \{1\emptyset\emptyset,\ 1\emptyset\}^*\,1\emptyset 1\,\{001,\ 01\}^*\,\Sigma_L^*,$$

or

$$h(\underline{\psi}_1'^R)\,1\emptyset\,001\,h(\underline{\psi}_2'')\,w \quad \Rightarrow_G, \quad h(\underline{\psi}_1'^R)\,101\,h(\underline{\psi}_2'')\,w \in \{1\emptyset\emptyset,\ 1\emptyset\}^*\,101\,\{001,\ 01\}^*\,\Sigma_L^*.$$

The derivation is blocked in both cases, there is no substring to be modified. Hence, no terminal string can be derived in G' either.

B2: If $\varphi_1 \neq \varphi_2$, but there is no position with different bits, then φ_2 is a proper prefix of φ_1, or vice versa. The derivation in G will then produce either

$$\varphi_1'^R\,\varphi_2''\,w \quad \Rightarrow_G^* \quad \underline{\psi}'^R\,w \in \{0',\ 1'\}^+\,\Sigma_L^*,$$

or

$$\varphi_1'^R\,\varphi_2''\,w \quad \Rightarrow_G^* \quad \underline{\psi}''\,w \in \{0'',\ 1''\}^+\,\Sigma_L^*,$$

corresponding to

$$h(\underline{\psi}'^R)\,w \in \{1\emptyset\emptyset,\ 1\emptyset\}^+\,\Sigma_L^*,$$

or to

$$h(\underline{\psi}'')\,w \in \{001,\ 01\}^+\,\Sigma_L^*,$$

respectively. Neither here can we derive a terminal string.

Thus, we have shown that $\varphi_1'^R\,\varphi_2''\,w \Rightarrow_G^* w$ if and only if $h(\varphi_1'^R)\,h(\varphi_2'')\,w \Rightarrow_G^*, w$, which completes the proof. □

We shall now present a normal form using only three nonterminals, but a little longer non-context-free rules:

THEOREM 5: *Each recursively enumerable language L can be generated by a phrase-structure grammar with 3 nonterminal symbols, using only context-free rules of the form $S \to v$, where S is the initial symbol, but two extra rules $AA \to \varepsilon$, $BBB \to \varepsilon$.*

Proof: We shall again simulate the phrase-structure grammar $G=(N, T, P, S)$ of Theorem 3, but now a homomorphism h will be defined by

$$h(0')=100, \qquad h(0'')=01,$$
$$h(1')=10, \qquad h(1'')=001.$$

Define a grammar $G'=(N', T, P', S)$ by $N'=\{S, 0, 1\}$,

$$P'=GEN' \cup EPC' \cup TEST',$$

where

$$GEN'=\{S \to h(z_x'^R)\, S\, x;\ x \in \Sigma_L\},$$
$$EPC'=\{S \to h(u_s'^R)\, S\, h(v_s'');\ s=1, \ldots, r\} \cup \{S \to h(u_s'^R)\, h(v_s'');\ s=1, \ldots, r\}.$$

Cancellation of $0'0''$, and $1'1''$ will now correspond to

$$TEST'=\{000 \to \varepsilon,\ 11 \to \varepsilon\}.$$

A: For the first two stages, we can state the exact correspondence between derivations in G and in G':

$$S \Rightarrow_G^* \alpha_1'^R S \beta_1'' w \Rightarrow_G \alpha_1'^R \alpha_2'^R \beta_2'' \beta_1'' w = \varphi_1'^R \varphi_2'' w$$

will correspond to

$$S \Rightarrow_{G'}^* h(\alpha_1'^R)\, S\, h(\beta_1'') w \Rightarrow_{G'} h(\alpha_1'^R)\, h(\alpha_2'^R)\, h(\beta_2'')\, h(\beta_1'') w = h(\varphi_1'^R)\, h(\varphi_2'') w.$$

The $TEST'$ rules of G' are not applicable, since the sentential form is in $\{100, 10\}^* S \{01, 001\}^* \Sigma_L^*$, and hence there are no substrings of the form 000, or 11.

B: In the third stage of the derivation, we have $h(\varphi_1'^R)\, h(\varphi_2'') w \in \{100, 10\}^* \{01, 001\}^* \Sigma_L^*$, so the only place we can modify is at the frontier

between $h(\varphi_1'^R)$ and $h(\varphi_2'')$, by *TEST'* rules. The argument is therefore similar to the proof of Theorem 4, there are the same cases to consider.

Both cancellation of zeroes and ones correspond to the cancellation of 1 0001:

$$h(0')\,h(0'') = 100\,01 \quad \Rightarrow_G, \quad 11 \quad \Rightarrow_G, \quad \varepsilon,$$
$$h(1')\,h(1'') = 10\,001 \quad \Rightarrow_G, \quad 11 \quad \Rightarrow_G, \quad \varepsilon,$$

by the rules $000 \to \varepsilon$, $11 \to \varepsilon \in TEST'$. Therefore, if $\varphi_1'^R \varphi_2'' w \Rightarrow_G^* w$, then $\varphi_1 = \varphi_2$, and also $h(\varphi_1'^R)\,h(\varphi_2'')\,w \Rightarrow_G^*, w$.

B1: If $\varphi_1 \neq \varphi_2$ and we get a "prohibited" combination of different frontier symbols, *i.e.*,
either

$$\varphi_1'^R \varphi_2'' w \quad \Rightarrow_G^* \quad \underline{\psi}_1'^R\, 1'\, 0''\, \underline{\psi}_2'' w,$$

or

$$\varphi_1'^R \varphi_2'' w \quad \Rightarrow_G^* \quad \underline{\psi}_1'^R\, 0'\, 1''\, \underline{\psi}_2'' w,$$

then either

$$h(\varphi_1'^R)\,h(\varphi_2'')\,w \quad \Rightarrow_G^*, \quad h(\underline{\psi}_1'^R)\,10\,01\,h(\underline{\psi}_2'')\,w,$$

or

$$h(\varphi_1'^R)\,h(\varphi_2'')\,\mathrm{w} \quad \Rightarrow_G^*, \quad h(\underline{\psi}_1'^R)\,100\,001\,h(\underline{\psi}_2'')\,w,$$

respectively. The derivation has already been blocked in the first case, since the sentential form is in $\{100, 10\}^* 1001 \{01, 001\}^* \Sigma_L^*$. In the second case, the derivation can continue, but the only possible step is

$$h(\underline{\psi}_1'^R)\,100\,001\,h(\underline{\psi}_2'')\,w \quad \Rightarrow_G, \quad h(\underline{\psi}_1'^R)\,101\,h(\underline{\psi}_2'')\,w \in \{100, 10\}^* 101 \{01, 001\}^* \Sigma_L^*.$$

No further steps are possible and therefore the derivation is blocked.

B2: If $\varphi_1 \neq \varphi_2$ because φ_2 is a proper prefix of φ_1, or vice versa, then we get a sentential form in $\{0', 1'\}^+ \Sigma_L^*$, or in $\{0'', 1''\}^+ \Sigma_L^*$. In G', this corresponds to sentential forms in $\{100, 10\}^+ \Sigma_L^*$, or $\{01, 001\}^+ \Sigma_L^*$, respectively. No terminal string can be generated in either case. □

Now, we shall show a grammar with a single extra rule:

THEOREM 6: *Each recursively enumerable language L can be generated by a grammar with 3 nonterminal symbols, using only context-free rules of the form $S \to v$, where S is the initial symbol, and a single extra rule $ABBBA \to \varepsilon$.*

Proof: The phrase-structure grammar of the previous theorem used two extra rules, namely $000 \to \varepsilon$, and $11 \to \varepsilon$. A careful study of the proof reveals that, in each derivation generating a terminal string, these rules were always applied in pairs, rewriting by $000 \to \varepsilon$ was immediately followed by the use of $11 \to \varepsilon$. Furthermore, they were always applied at the same place, *i.e.*

$$\ldots \Rightarrow \alpha' 10001 \beta'' w \Rightarrow \alpha' 11 \beta'' w \Rightarrow \alpha' \beta'' w \Rightarrow \ldots$$

Therefore, a single extra rule $10001 \to \varepsilon$ will do as well. □

Finally, we are going to present another normal form using only a single non-context-free rule:

THEOREM 7: *Each recursively enumerable language L can be generated by a phrase-structure grammar with 4 nonterminal symbols, using only context-free rules of the form $S \to v$, where S is the initial symbol, and a single extra rule $ABC \to \varepsilon$.*

Proof: The idea is again to imitate the phrase-structure grammar of Theorem 3. The detailed proof is very similar to the previous ones, so we content ourselves with a construction. Define a homomorphism h as follows:

$$h(0') = AB, \qquad h(0'') = C,$$
$$h(1') = A, \qquad h(1'') = BC.$$

Next, construct a grammar $G' = (N', T, P', S)$ by $N' = \{S, A, B, C\}$, $P' = GEN' \cup EPC' \cup TEST'$, where

$$GEN' = \{ S \to h(z_x'^R) S x;\ x \in \Sigma_L \},$$

$$EPC' = \{ S \to h(u_s'^R) S h(v_s'');\ s = 1, \ldots, r \} \cup \{ S \to h(u_s'^R) h(v_s'');\ s = 1, \ldots, r \},$$

$$TEST' = \{ ABC \to \varepsilon \}. \quad \square$$

5. TIME AND SPACE COMPLEXITY

First, we shall briefly review the time and space complexities of grammars constructed in Section 3 and 4. Then we shall show that the simulation of nondeterministic k-tape Turing machine by a grammar in normal form is quadratic, *i.e.* it is not less time efficient than the simulation by an arbitrary phrase-structure grammar.

Let L be a recursively enumerable language generated by some g-system G. By Theorems 1 and 2, we can construct an extended Post correspondence P such that $L(P)=L(G)$. If a word $w=x_1 \ldots x_n \in L(G)$ is generated by a derivation

$$S=w_0 \Rightarrow_G w_1 \Rightarrow_G \ldots \Rightarrow_G w_k \Rightarrow_G w_{k+1}=w$$

then, by (2.7), there exists a solution of P for w, *i.e.* there exist $s_1, \ldots, s_l \in \{1, \ldots, r\}$, $l \geqq 1$ satisfying $v_{s_1} \ldots v_{s_l}=u_{s_1} \ldots u_{s_l} z_{x_1} \ldots z_{x_n}$ such that $l \leqq c \sum_{i=0}^{k} |w_i|$. It can be shown that the simulation of P by a phrase-structure grammar of Theorem 3 is linear: The first stage of its derivation consists of n steps, $n=|w|=|w_{k+1}|$, the second stage requires l steps, $l \leqq c \sum_{i=0}^{k} |w_i|$. The third one requires $|\varphi_2''|$ steps, since each step cancels exactly one symbol of φ_2'', together with one symbol of $\varphi_1'^R$. Note that $|\varphi_2''| = \sum_{i=1}^{l} |v_{s_i}''| = \sum_{i=1}^{l} |v_{s_i}| \leqq M.l$, where $M=\max_{j=1}^{r} |v_j|$ (a constant dependent only on P). Thus the total number of steps is at most

$$n+l+M.l \leqq d'.l \leqq d \sum_{i=0}^{k+1} |w_i|,$$

for suitable constants d', d. The space complexity is also bounded by $O(\sum |w_i|)$.

All other grammars, constructed in Section 4, simulated the grammar of Theorem 3. Each nonterminal symbol was encoded by a string of length at most three, similarly, each derivation step was imitated by at most three steps. Thus all these grammars have the same time and space bound $O(\sum |w_i|)$.

In order to establish our normal forms, we could use the classical definition of the phrase-structure grammar as well, instead of introducting a notion of

a g-system. However, an iterated rewriting of the sentential form by a nondeterministic 1-a-transducer (*i.e.* g-system) makes manipulation much easier and mentally simpler. For example, we have shown in [1] that every regular set can be generated by logarithmically time-bounded g-system having $\sum |w_i| \in O(n)$, which gives a linear time-bound for the phrase-structure grammar in our normal form, for each regular set. We are now going to show that, if a language L is recognized by a k-pushdown non-deterministic automaton A of time complexity $T_A(n)$ and space complexity $S_A(n)$, then L is also generated by a g-system of the same time and space complexity (for each k). From this we shall obtain a phrase-structure grammar in normal form with the time and space complexity bounded by $\sum_{t=0}^{T_A(n)} p_t$, where p_t is the number of symbols saved onto the pushdown stores of A at time t.

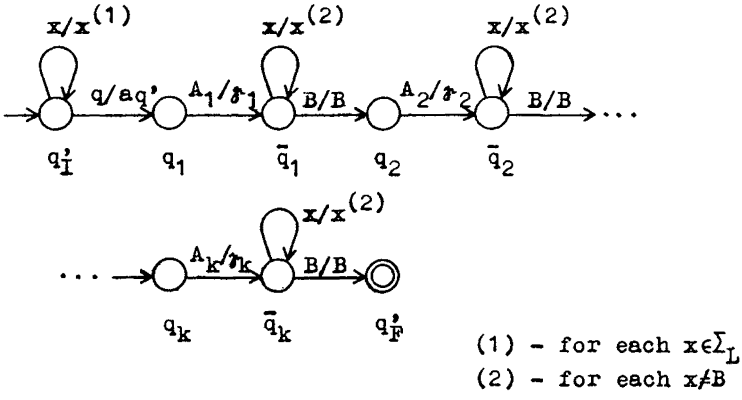
The proof is quite intuitive: Let $x_1 \ldots x_n$ be an input of A. To describe the computational history of A at any given time we need to know three things; the state q it is in, what has been scanned on the input tape, *i.e.* $x_1 \ldots x_i$, and what is on its pushdown stores, *i.e.* strings $A_j \beta_j$ (with the symbol A_j on top, for each $j = 1, \ldots, k$). In g-system this will correspond to the sentential form

$$x_1 \ldots x_i q A_1 \beta_1 B A_2 \beta_2 B \ldots A_k \beta_k B,$$

where B is a new symbol, used as a bottom-of-the-stack marker. A move by a particular transition $(q', \gamma_1, \ldots, \gamma_k) \in \delta(q, a, A_1, \ldots, A_k)$ – the automaton changes its state from q to q' and replaces the topmost symbol A_j by the string γ_j on its j-th pushdown (for each $j = 1, \ldots, k$), providing the input head scans the symbol a-will be imitated by rewriting the above sentential form to

$$x_1 \ldots x_i a q' \gamma_1 \beta_1 B \gamma_2 \beta_2 B \ldots \gamma_k \beta_k B.$$

(Where $a = x_{i+1}$, or $a = \varepsilon$.) The following edges in G are needed:



(For each transition of A, we use some new distinct states $q_1, \ldots, q_k$, $\bar{q}_1, \ldots, \bar{q}_k$, so the only states shared by different paths are q'_I, and q'_F.) The derivation will be initiated by rewriting S to $q_I (Z_I B)^k$, which corresponds to the initial computational history of A, and terminated by a path of edges from q'_I to q'_F erasing $q_F B^k$. (q_I, q_F, and Z_I denote the initial and final states of A, and its initial pushdown symbol, respectively.) It should be clear that A and G represent the same language, and that their time and space complexities are asymptotically equal.

Since k tapes can be easily replaced by $2k$ pushdowns, it follows immediately that each k-tape nondeterministic Turing machine can be replaced by a phrase-structure grammar in normal form of time complexity $O(T_A(n) \cdot S_A(n)) \subseteq O(T_A^2(n))$. This simulation result can be extended up to the multihead multidimensional multitape machines, to the L-systems with interactions, and so forth.

We would like to conclude this paper by an interesting question concerning a phrase-structure grammar having all of its rules context-free, but one pair of canceled parentheses, *i.e.*, one extra rule of the form $(\ ) \to \varepsilon$, which was presented in [2]. [The two extra rules of Theorem 3 can be viewed as a cancellation of two pairs of parentheses, *i.e.*, $(\ ) \to \varepsilon$, $\langle\ \rangle \to \varepsilon$.] This grammar is equivalent to a grammar with context dependency restricted to synchronization, a monotonic variant of this grammar corresponds to an abstract family of languages (not full) lying between the context-free and the context-sensitive languages. The same class of languages was obtained independently in [8], by the use of the Dyck_1-reductions on the context-free languages.

ACKNOWLEDGEMENTS

The author thanks Branislav Rovan for several helpful discussions, suggestions, and comments concerning this work.

This work was performed as a part of SPZV I-1-5/08 grant.

REFERENCES

1. V. GEFFERT, A Representation of Recursively Enumerable Languages by two Homomorphisms and a Quotient, *Theoret. Comput. Sci.*, 1988, *62*, pp. 235-249.
2. V. GEFFERT, Grammars with Context Dependency Restricted to Synchronization, *Proc. of M.F.C.S'* 86, *L.N.C.S.*, 1986, *233*, Springer-Verlag, pp. 370-378.
3. V. GEFFERT, Context-Free-Like Forms for the Phrase-Structure Grammars, *Proc. of M.F.C.S.'* 88, *L.N.C.S.*, 1988, *324*, Springer-Verlag, pp. 309-317.
4. S. GINSBURG, Algebraic and Automata-Theoretic Properties of Formal Languages, *North-Holland*, Amsterdam, 1975.

5. M. A. HARRISON, Introduction to Formal Language Theory, *Addison-Wesley*, 1978.
6. M. A. HARRISON and M. SCHOLNICK, A Grammatical Characterization of One Way Nondeterministic Stack Languages, *Ibid.*, 1971, *18*, pp. 148-172.
7. J. E. HOPCROFT and J. D. ULLMAN, Formal Languages and Their Relation to Automata, *Addison-Wesley*, 1969.
8. M. JANTZEN, M. KUDLEK, K. L. LANGE and H. PETERSEN, Dyck_1-Reductions of Context-Free Languages, *Comput. Artificial Intelligence*, 1990, *9*, No. 1, pp. 3-18.
9. E. POST, A Variant of a Recursively Unsolvable Problem, *Bull. Amer. Math. Soc.*, 1946, *52*, pp. 264-268.
10. B. ROVAN, A Framework for Studying Gramars, *Proc. of M.F.C.S'* 81, *L.N.C.S.*, 1981, *118*, Springer-Verlag, pp. 473-482.
11. B. ROVAN, Complexity Classes of g-Systems are AFL, Univ. Comeniana, *Acta Math. Univ. Comenian.*, *XLVIII-XLIX*, 1986, pp. 283-297.
12. E. S. SANTOS, A Note on Bracketed Grammars, *J. Assoc. Comput. Mach.*, 1972, *19*, pp. 222-224.
13. W. J. SAVITCH, How to Make Arbitrary Grammars Look Like Context-Free Grammars, *S.I.A.M. J. Comput.*, *2*, No. 3, 1973.