Y. BAR-HILLEL, M. PERLES and E. SHAMIR, JERUSALEM

On formal properties of simple phrase structure grammars *

Section 0: Introduction

This paper deals with mathematical problems concerning simple phrase structure grammars (SPGs, for short). The interest in these systems arose from linguistic considerations, as SPGs are regarded as formal models which are able to describe the syntactic structure of natural languages with good approximation. The problem to what degree SPGs are adequate to describe natural language grammars will not be dealt with here. However, a satisfactory answer to this adequacy problem requires, inter alia, a deep knowledge of the formal (mathematical) properties of the model. Chomsky himself, who introduced the SPGs as well as other formal grammar-models, has published several works on the formal properties of the models [C. M., C. 1, C. 2] (cf. also [B. G. S.]).

The problems studied in this paper include: representations of SPGs, relations with other formal systems (in particular with finite automata), characterizations of the languages obtainable by SPGs, closure under Boolean operations and decision problems for various properties of SPGs.

In Sec. 6 we prove that several properties of SPGs are *recursively undecidable*. For the exact meaning of this notion we refer the reader to Davis' book [D.]. This book contains also a chapter (no. 6) on combinatorial systems, of which the SPGs are a rather special kind.

Among the more significant results of Sec. 6 we would like to mention Th. 6.3. c establishing the undecidability of the problem whether a given SPG is equivalent to a finite automaton.

Section 1: Notation, terminology and basic definitions

Let V be a given set – the *vocabulary*. Elements of V will be called *symbols* and denoted by capital Latin letters – X, Y, Z etc. Finite sequences of symbols of V – including the empty sequence – will be called *strings* over V and denoted by small Latin letters – x, y, z etc. The empty string will be denoted by Λ. Sets of strings over V will be called *languages* over V and denoted by Latin capitals – generally L with subscripts. If $x \in L$, we say that x is *a sentence* of L. The set of all strings over V will be denoted by W_v. A symbol $X \in W_v$ will be identified with the string of length 1 composed of X alone, hence we take $V \subseteqq W_v$. The length of the string x (i. e. the number of its symbol occurrences) will be denoted by $l(x)$.

Reflection: If $x = X_{i_1} X_{i_2} \cdots X_{i_{n-1}} X_{i_n}$, then the reflection of x is the string:

$$x^* = X_{i_n} X_{i_{n-1}} \cdots X_{i_2} X_{i_1} \quad (\Lambda^* = \Lambda).$$

* This work was supported by the U.S, Office of Naval Research, Information Systems Branch, under Contract N 62558–2214.

Concatenation: If $x = X_{i_1} \cdots X_{i_m}$, $y = X_{j_1} \cdots X_{j_n}$, then the concatenate of x and y is the string:

$$x \cdot y = X_{i_1} \cdots X_{i_m} X_{j_1} \cdots X_{j_n} \qquad (\Lambda \cdot x = x \cdot \Lambda = x).$$

(For $x \cdot y$ we shall usually write simply xy.)

Powers: $x^0 = \Lambda$, $x^1 = x$, and generally $x^n = x \cdot x^{n-1}$ $(n = 1, 2, \ldots)$.

Substrings: x is a substring of y $(x \subseteqq y)$ if there are strings u and v (possibly empty), such that $y = uxv$. x is a *proper* substring of y $(x \subset y)$ if $x \subseteqq y$ and $x \neq y$.

W_v is the free semi-group with identity $-\Lambda-$ over the set of generators V, under the operation of concatenation.

Reflection, products and powers of languages: The reflection of a language L is the language $L^* = \{x^* | x \in L\}$; The product of two languages L_1 and L_2 is the language $L_1 \cdot L_2 = \{x \cdot y | x \in L_1,\ y \in L_2\}$. Powers of a language L are defined as follows:

$L^0 = \{\Lambda\}$, $L^1 = L$ and generally $L^n = L \cdot L^{n-1}$ $(n = 1, 2, \ldots)$. $\{x\} \cdot L$ and $L \cdot \{x\}$ will be abbreviated as xL and Lx, respectively.

Multiplication of languages is the ordinary multiplication of complexes in semi-group theory.

Closure of a language L: $cl(L) = \bigcup_{n=0}^{\infty} L^n$.

$\bar{cl}(L)$ is the sub-semi-group with identity of W_v generated by L.

Grammars: By a grammar we understand a finite system of rules determining a language. Grammars will be denoted by Gothic capitals. The language determined by $\mathfrak{A}$ will be denoted by $L(\mathfrak{A})$. If $L(\mathfrak{A}) = L(\mathfrak{B})$, then $\mathfrak{A}$ and $\mathfrak{B}$ are called *equivalent.*

If $\mathfrak{A}$ is a device generating the sentences of $L(\mathfrak{A})$, we call $\mathfrak{A}$ a *production grammar.* If $\mathfrak{A}$ is a device which recognizes, given a string x, whether $x \in L(\mathfrak{A})$ or not, it is called a *recognition grammar.* If $\mathfrak{A}$ is a recognition grammar and $x \in L(\mathfrak{A})$, we say that x is *accepted* by $\mathfrak{A}$.

In the sequel we assume the vocabulary V to be finite.

Definition 1.1. a: A *simple phrase structure system* (SPS) is an ordered couple (V, P), where V is a finite vocabulary, and P is a finite set of *productions* of the form

$$X \to x \qquad (x \neq X,\ X \in V,\ x \in W_v, \text{ and possibly } x = \Lambda).$$

1.1. b: y directly generates z $(y \Rightarrow z)$, if $y = uXv$, $z = uxv$, and $X \to x$ is a production of P.

1.1. c: y generates z $(y \overset{*}{\Rightarrow} z)$, if there exists a sequence of strings $z_0, z_1, \ldots, z_r$ $(r \geq 0)$, such that $y = z_0$, $z_r = z$, and $z_{i-1} \Rightarrow z_i$ $(i = 1, \ldots, r)$. (In case $r = 0$, we have $y = z_0 = z_r = z$, which corresponds to the relation $y \overset{*}{\Rightarrow} y$.) The sequence $z_0, \ldots, z_r$ will be called a *generation tree* of z from y. It is generally not uniquely determined by y and z.

Obviously, if $x \overset{*}{\Rightarrow} y$ and $y \overset{*}{\Rightarrow} z$, then $x \overset{*}{\Rightarrow} z$. Also, if $x_1 \overset{*}{\Rightarrow} y_1$, $x_2 \overset{*}{\Rightarrow} y_2$, $\ldots$, $x_n \overset{*}{\Rightarrow} y_n$, then $x_1 x_2 \cdots x_n \overset{*}{\Rightarrow} y_1 y_2 \cdots y_n$. Conversely, if $x \overset{*}{\Rightarrow} y$ and $x_1 = x_1 \cdots x_n$, then there are strings $y_1, \ldots, y_n$, such that $x_i \overset{*}{\Rightarrow} y_i$ $(i = 1, \ldots, n)$, and

$y = y_1 \cdots y_n$. The last assertion can easily be verified by induction on the length of a generation tree of y from x.

Definition 1.2. a: A *simple phrase structure grammar* (SPG) is an ordered quadruple $\mathfrak{G} = (V, P, T, S)$, where (V, P) is a SPS (see Def. 1.1. a), T (the *terminal* vocabulary) is a subset of V, none of whose elements occur on the left side of a production of P, and S (the *initial* symbol) is a distinguished element of $V-T$. $V-T$ is called the *auxiliary* vocabulary.

1.2. b: x is a *sentence* of $\mathfrak{G}$, if x is a string over T (a *terminal string*), and $S \overset{*}{\Rightarrow} x$ (in the SPS (V, P)). $L(\mathfrak{G})$ is the set of all sentences of $\mathfrak{G}$:

$$L(\mathfrak{G}) = \{x \mid x \in W_T \;\&\; S \overset{*}{\Rightarrow} x\}.$$

1.2. c: A language L is *representable* by an SPG, or is a *simple phrase structure language* (SPL), if there exists an SPG $\mathfrak{G}$, such that $L = L(\mathfrak{G})$.

Section 2: Representation of finite automata by SPGs

In this section we show how to construct SPGs equivalent or otherwise closely related to one-tape and two-tape finite automata, as defined in [R. S.].

The following definition is adopted from [R. S., Defs. 1,2,3].

Definition 2.1. a: A *finite automaton* (FA) is a system $\mathfrak{A} = (T, \Sigma, M, \sigma_0, \Phi)$, where T is a finite vocabulary, Σ is a finite set (the *internal states* of $\mathfrak{A}$), M is a function defined on the Cartesian product $\Sigma \times T$ with values in Σ (the *transition table*), $\sigma_0 \in \Sigma$ (the *initial state*), and $\Phi \subseteq \Sigma$ (the set of *designated final states*).

M can be extended from $\Sigma \times T$ to $\Sigma \times W_T$ by a definition by recursion as follows:

$$M(\sigma, \Lambda) = \sigma \quad \text{for } \sigma \in \Sigma;$$
$$M(\sigma, xX) = M(M(\sigma, x), X), \quad \text{for } \sigma \in \Sigma,\ x \in W_T,\ X \in T.$$

The extended function M has the following property:

$$M(\sigma, x \cdot y) = M(M(\sigma, x), y) \text{ for } \sigma \in \Sigma,\ x, y \in W_T.$$

This can easily be proved by induction on the length of y. $M(\sigma, x) = \tau$ means that if the automaton reads through the whole string x from left to right, symbol by symbol, beginning in state σ and changing states according to the transition table M, then it will end in state τ.

2.1. b: Let $x \in W_T$. x is *accepted* by $\mathfrak{A}$, if $M(\sigma_0, x) \in \Phi$.

$$L(\mathfrak{A}) = \{x \mid M(\sigma_0, x) \in \Phi\}.$$

2.1. c: A language L is *representable* by a FA, or is an FAL, if $L = L(\mathfrak{A})$ for some FA $\mathfrak{A}$.

The class of all FALs over a vocabulary T is closed under Boolean operations as well as under reflection, multiplication and formation of closure. It is the least class of languages over T containing the finite languages and closed under the formation of unions, products and closures of languages. For proofs, as well as for more information about FALs and their representations, see [R. S.] and also [B-H. S.].

Theorem 2.1: Let $\mathfrak{A} = (T, \Sigma, M, \sigma_0, \Phi)$ be a FA.

a) It is possible to construct an SPG $\mathfrak{G}$ equivalent to $\mathfrak{A}$.

b) It is possible to construct an SPG $\mathfrak{G}'$, such that

$$L(\mathfrak{G}') = \{y \mid y = x\,E\,x^*,\ x \in L(\mathfrak{A})\},$$

where E is a fixed additional symbol.

Proof: a) Define $\mathfrak{G} = (V, P, T, \sigma_0)$, where $V = T \cup \Sigma$ (we may assume $\Sigma \cap T = \emptyset$), $P = \{\sigma \to X \cdot M(\sigma, X) \mid \sigma \in \Sigma,\ X \in T\} \cup \{\sigma \to \Lambda \mid \sigma \in \Phi\}$. If $x \in W_T$, then $\sigma_0 \overset{*}{\Rightarrow} x$ in $\mathfrak{G}$ if and only if either $x = y\,\sigma$, where $y \in W_T$ and $\sigma = M(\sigma_0, y)$, or $x \in W_T$ and $M(\sigma_0, x) \in \Phi$. Hence, if $x \in W_T$, then $\sigma_0 \overset{*}{\Rightarrow} x$ in $\mathfrak{G}$ if and only if $M(\sigma_0, x) \in \Phi$, i. e., $L(\mathfrak{A}) = L(\mathfrak{G})$.

b) Define $\mathfrak{G}' = (V', P', T', \sigma_0)$, where $V' = T \cup \{E\} \cup \Sigma$ (we may assume that $T \cap \Sigma = \emptyset$ and that $E \notin \Sigma$), $T' = T \cup \{E\}$,

$$P = \{\sigma \to X \cdot M(\sigma, X) \cdot X \mid \sigma \in \Sigma,\ X \in T\} \cup \{\sigma \to E \mid \sigma \in \Phi\}.$$

$\sigma_0 \overset{*}{\Rightarrow} y$ in $\mathfrak{G}'$, if and only if either $y = x\,\sigma\,x^*$, where $x \in W_T$ and $\sigma = M(\sigma_0, x)$, or $y = xEx^*$, where $x \in W_T$ and $M(\sigma_0, x) \in \Phi$. Hence,

$$L(\mathfrak{G}') = \{y \mid y \in W_{T'} \ \&\ \sigma_0 \overset{*}{\Rightarrow} y\} = \{y \mid y = xEx^*,\ x \in L(\mathfrak{A})\}.$$

Remarks: a) In order to get an SPG $\mathfrak{G}''$, such that

$$L(\mathfrak{G}'') = \{y \mid y = xx^*,\ x \in L(\mathfrak{A})\},$$

without an auxiliary symbol E, define $\mathfrak{G}'' = \{V, P'', T, \sigma_0\}$, where $V = T \cup \Sigma$,

$$P'' = \{\sigma \to X \cdot M(\sigma, X) \cdot X \mid \sigma \in \Sigma,\ X \in T\} \cup \{\sigma \to \Lambda \mid \sigma \in \Phi\}.$$

b) If $\mathfrak{A} = \{T, \Sigma, M, \sigma_0, \Phi\}$ is a FA, $E \notin T$, and $L(\mathfrak{A})$ is infinite, then the language $L' = \{y \mid y = xEx^*,\ x \in L(\mathfrak{A})\}$ is not an FAL. This follows easily from [R. S., Lemma 8]. This remark, together with Th. 2.1. b, demonstrates the existence of SPGs which are not equivalent to any FA.

c) The following converse of Th. 2.1. a holds true: Let $\mathfrak{G} = (V, P, T, S)$ be an SPG, all of whose productions are of the form $A_i \to A_j X_k$, $A_i \to X_k$ or $A_i \to \Lambda$ (or of the form $A_i \to X_k A_j$, $A_i \to X_k$ or $A_i \to \Lambda$), where A_i, A_j, $A_k \in V\text{-}T$, $X_k \in T$. It is possible to construct a FA $\mathfrak{A}$ equivalent to $\mathfrak{G}$.

Method of proof: First construct a nondeterministic FA $\mathfrak{A}'$ (see [R. S., Defs. 9–10]) equivalent to $\mathfrak{G}$. Then construct a deterministic FA $\mathfrak{A}$ equivalent to $\mathfrak{A}'$, by [R. S., Def. 11 and Th. 11]. For a detailed proof of a similar result see [B-H. S., § 3].

d) The following converse of Th. 2.1. b will be proved in Sec. 7, Th. 7.2: Let L be a language over a vocabulary T, and E a symbol not contained in T. Let $L' = \{y \mid y = xEx^*,\ x \in L\}$. If L' is an SPL, then L is an FAL.

The following definition is adopted from [R. S., Defs. 15–16]. Def. 16 has been slightly corrected. The intuitive motivation for this definition is given in [R. S., preceding Def. 15].

Definition 2.2. a: A *two-tape finite automaton* (TTA) is a system

$$\mathfrak{A}^2 = (T, E, \Sigma, M, \sigma_0, \Phi, \Sigma_1, \Sigma_2),$$

where $(T \cup \{E\}, \Sigma, M, \sigma_0, \Phi)$ is an ordinary FA, $E \notin T$, and the sets Σ_1, Σ_2 form a *partition* of Σ, i. e., $\Sigma_1 \cap \Sigma_2 = \emptyset$, $\Sigma_1 \cup \Sigma_2 = \Sigma$. E is an auxiliary symbol (the *end-marker*).

A TTA accepts or rejects ordered pairs of strings over T. The following notation will be useful for defining when a pair of strings is accepted.

Let $\sigma_0, \sigma_1, \ldots, \sigma_n$ be a sequence of internal states of a TTA

$$\mathfrak{A}^2 = (T, E, \Sigma, M, \sigma_0, \Phi, \Sigma_1, \Sigma_2).$$

A pair of *associated sequences* of integers $k_0, \ldots, k_n$; $l_0, \ldots, l_n$ is defined as follows:

1. k_i is 1 or 2 according as $\sigma_i \in \Sigma_1$ or $\sigma_i \in \Sigma_2$;
2. l_i is the number of indices $j \leqq i$ such that $k_j = k_i$.

2.2. b: Let $\mathfrak{A}^2$ be a TTA, and (x_1, x_2) an ordered pair of strings over T. Let

$$(x_1 E, x_2 E) = (X_{1,1} X_{1,2} \cdots X_{1,m}, X_{2,1} X_{2,2} \cdots X_{2,n}), \text{ i. e.,}$$
$$x_1 = X_{1,1} \cdots X_{1,m-1},\ x_2 = X_{2,1} \cdots X_{2,n-1}, \text{ and } X_{1,m} = X_{2,n} = E.$$

The pair (x_1, x_2) is accepted by $\mathfrak{A}^2$ if and only if there is a (unique) sequence of states $\sigma_0, \sigma_1, \ldots, \sigma_p$, and associated sequences of integers $k_0, \ldots, k_p$; $l_0, \ldots, l_p$, such that:

1. σ_0 is the initial state of $\mathfrak{A}^2$;
2. $\sigma_i = M(X_{k_{i-1}, l_{i-1}}, \sigma_{i-1})$ for $i = 1, \ldots, p$;
3. for $i = 0, 1, \ldots, p-1$, if $k_i = 1$, then $l_i \leqq m$, and if $k_i = 2$, then $l_i \leqq n$;
4. if $k_p = 1$, then $l_p = m + 1$, and if $k_p = 2$, then $l_p = n + 1$;
5. $\sigma_p \in \Phi$.

$L(\mathfrak{A}^2)$ is the set of all pairs of strings over T accepted by $\mathfrak{A}^2$.

The intuitive picture is as follows: the two strings $x_1 E$ and $x_2 E$ are printed on two tapes and fed into the machine. The machine begins to operate in the initial state σ_0. When it is in a state $\sigma \in \Sigma_i$, it reads on the i-th tape ($i = 1,2$). The machine reads through both tapes from left to right, symbol after symbol, changing states according to the transition table M, and switching from one tape to another according to the partition (Σ_1, Σ_2) of the internal states. Finally, one of the strings is exhausted, and the machine scans on one of the tapes the first empty square following the symbol E. Then it stops, and the pair (x_1, x_2) is accepted if and only if the machine is in a designated final state $\sigma_p \in \Phi$. Thus it may happen that the machine accepts or rejects a pair (x_1, x_2) without having scanned all the symbols of x_1 or all those of x_2. The end-markers E enable the machine to notice the end of the strings x_1 and x_2 before getting off the printed tape.

Theorem 2.2: Let $\mathfrak{A}^2 = (T, E, \Sigma, M, \sigma_0, \Phi, \Sigma_1, \Sigma_2)$ be a TTA. It is possible to construct an SPG $\mathfrak{G}$, such that

$$L(\mathfrak{G}) = \{z \mid z = xEy^*,\ (x, y) \in L(\mathfrak{A}^2)\}.$$

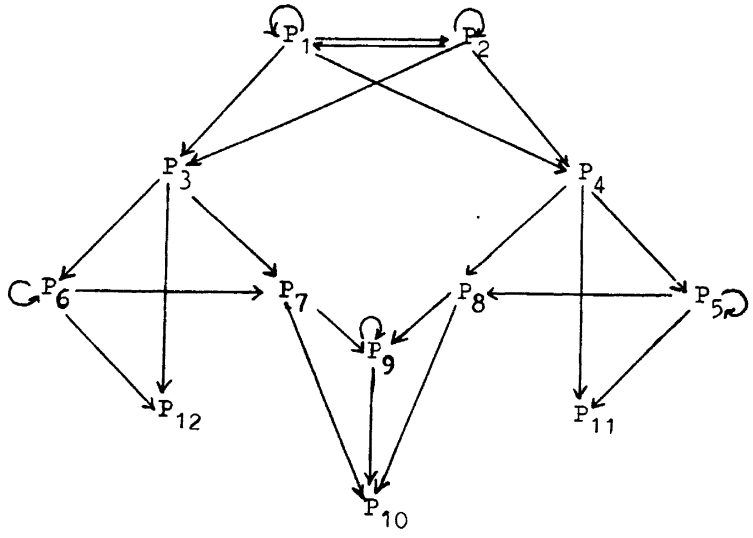
In principle the construction is similar to those in the proof of Th. 2.1. But the construction and the proof of this theorem are longer and more difficult, due to the complications in the definition of acceptance of pairs (x, y) by $\mathfrak{A}^2$.

Proof: Define $\mathfrak{G} = (V, P, T', \sigma_0)$ with $V = T \cup \{E\} \cup \Sigma \cup \Sigma' \cup \Sigma'' \cup \{\lambda\}$ where $\Sigma' = \{\sigma' \mid \sigma \in \Sigma\}$, $\Sigma'' = \{\sigma'' \mid \sigma \in \Sigma\}$ (we may assume that the sets T,

$\{E\}$, Σ, Σ', Σ'' and $\{\lambda\}$ are mutually disjoint); $T' = T \cup \{E\}$; $P = \bigcup_{i=1}^{12} P_i$, and

$$
\begin{aligned}
P_1 &= \{\sigma \to X \cdot M(\sigma, X) \mid \sigma \in \Sigma_1, X \in T\},\\
P_2 &= \{\sigma \to M(\sigma, X) \cdot X \mid \sigma \in \Sigma_2, X \in T\},\\
P_3 &= \{\sigma \to E\tau'' \mid \sigma \in \Sigma_1, \tau = M(\sigma, E)\},\\
P_4 &= \{\sigma \to \tau' E \mid \sigma \in \Sigma_2, \tau = M(\sigma, E)\},\\
P_5 &= \{\tau' \to X\varrho' \mid \tau \in \Sigma_1, X \in T, \varrho = M(\tau, X)\},\\
P_6 &= \{\tau'' \to \varrho'' X \mid \tau \in \Sigma_2, X \in T, \varrho = M(\tau, X)\},\\
P_7 &= \{\tau'' \to \lambda \mid \tau \in \Sigma_1 \cap \Phi\},\\
P_8 &= \{\tau' \to \lambda \mid \tau \in \Sigma_2 \cap \Phi\},\\
P_9 &= \{\lambda \to X\lambda \mid X \in T\},\\
P_{10} &= \{\lambda \to \Lambda\},\\
P_{11} &= \{\tau' \to \Lambda \mid \tau \in \Sigma_1, M(\tau, E) \in \Phi\},\\
P_{12} &= \{\tau'' \to \Lambda \mid \tau \in \Sigma_2, M(\tau, E) \in \Phi\}.
\end{aligned}
$$

The following diagram illustrates the order in which the twelve sets of productions P_i $(1 \leq i \leq 12)$ can be applied.



It may easily be verified that $\sigma_0 \overset{*}{\Rightarrow} z$ in $\mathfrak{G}$ if and only if either

1. $z = x\sigma y^*$, $x \in W_T$, $y \in W_T$ and σ is the state of the automaton $\mathfrak{A}^2$ after having scanned all the symbols of the pair (x, y), beginning in the initial state σ_0 (P_1, P_2); or

2. $z = xE\tau'' y^*$, $x \in W_T$, $y \in W_T$, and τ is the state of $\mathfrak{A}^2$ after having scanned all the symbols of the pair (xE, y), beginning in σ_0 (P_1, P_2, P_3, P_6); or

3. $z = x\tau' E y^*$, $x \in W_T$, $y \in W_T$, and τ is the state of $\mathfrak{A}^2$ after having scanned all the symbols of the pair (x, yE), beginning in σ_0 (P_1, P_2, P_4, P_5); or

4. $z = xE\lambda y^*$, and $\mathfrak{A}^2$ accepts the pair (x, y) after having scanned the whole string xE and only a part of the string yE $(P_1, P_2, P_3, P_6, P_7, P_9)$; or

5. $z = x\lambda Ey^*$, and $\mathfrak{A}^2$ accepts the pair (x, y) after having scanned the whole string yE and only a part of the string xE $(P_1, P_2, P_4, P_5, P_8, P_9)$; or

6. $z = xEy^*$, and $\mathfrak{A}^2$ accepts the pair (x, y) after having scanned the whole string xE and only a part of the string yE ($P_1, P_2, P_3, P_6, P_7, P_9, P_{10}$); or
7. $z = xEy^*$, and $\mathfrak{A}^2$ accepts the pair (x, y) after having scanned the whole string yE and only a part of the string xE ($P_1, P_2, P_4, P_5, P_8, P_9, P_{10}$); or
8. $z = xEy^*$, and $\mathfrak{A}^2$ accepts the pair (x, y) after having scanned the whole string xE *and* the whole string yE ($P_1, P_2, P_3, P_6, P_{12}$, or $P_1, P_2, P_4, P_5, P_{11}$).

Since $T' = T \cup \{E\}$, we get:

$$L(\mathfrak{G}) = \{xEy^* \mid (x, y) \in L(\mathfrak{A}^2)\}.$$

Remark: Let $\mathfrak{A}^2 = (T, E, \Sigma, M, \sigma_0, \Phi, \Sigma_1, \Sigma_2)$ be a TTA, and let

$$\mathfrak{A}^{2'} = (T, E, \Sigma, M, \sigma_0, \Sigma - \Phi, \Sigma_1, \Sigma_2).$$

Then $L(\mathfrak{A}^{2'}) = W_T \times W_T - L(\mathfrak{A}^2)$, where $W_T \times W_T$ is the set of all ordered pairs of strings over T. Hence, given a TTA $\mathfrak{A}^2$, we can construct an SPG $\mathfrak{G}'$, such that

$$L(\mathfrak{G}') = \{z \mid z = xEy^*, (x, y) \notin L(\mathfrak{A}^2)\} = W_T \cdot E \cdot W_T - \{z \mid z = xEy^*, (x, y) \in L(\mathfrak{A}^2)\}.$$

It is also easy to construct an SPG $\bar{\mathfrak{G}}$, such that $L(\bar{\mathfrak{G}})$ is the set of all strings over $T \cup \{E\}$ containing either no E at all or more than one E, i. e.,

$$L(\bar{\mathfrak{G}}) = W_{T \cup \{E\}} - W_T \cdot E \cdot W_T.$$

(This set is even representable by a FA.) In Sec. 3, Th. 3.1. e, we shall see how to construct an SPG $\mathfrak{G}''$, such that $L(\mathfrak{G}'') = L(\mathfrak{G}') \cup L(\bar{\mathfrak{G}})$. But

$$L(\mathfrak{G}') \cup L(\bar{\mathfrak{G}}) = (W_T \cdot E \cdot W_T - \{z \mid z = xEy^*, (x, y) \in L(\mathfrak{A}^2)\})$$
$$\cup (W_{T \cup \{E\}} - W_T \cdot E \cdot W_T) = W_{T \cup \{E\}} - \{z \mid z = xEy^*, (x, y) \in L(\mathfrak{A}^2)\}.$$

Thus, given a TTA $\mathfrak{A}^2 = (T, E, \Sigma, M, \sigma_0, \Phi, \Sigma_1, \Sigma_2)$, we can construct an SPG $\mathfrak{G}''$, such that $L(\mathfrak{G}'') = W_{T \cup \{E\}} - \{z \mid z = xEy^*, (x, y) \in L(\mathfrak{A}^2)\}$.

Section 3: Closure under Boolean and other operations

Summary of results:

The class of SPLs is effectively closed under reflection, as well as under the formation of unions, products and closures of languages (Th. 3.1). ("Effectively closed under reflection" means that, given an SPG $\mathfrak{G}$, we can effectively construct an SPG $\mathfrak{G}^*$, such that $L(\mathfrak{G}^*) = L(\mathfrak{G})^*$, and similarly for the other operations.)

The class of SPLs over a vocabulary T containing at least two symbols is not closed under the formation of intersections, nor under complementation with respect to W_T (Th. 3.2).

The class of SPLs is effectively closed under a quite general kind of substitution (Th. 3.3).

Lemma 3.1: Let $\mathfrak{G} = (V, P, T, S)$ be an SPG, let $X \in V - T$ be an auxiliary symbol and let Y be a new symbol not contained in V. If we replace X by Y in V and in both sides of all productions of P, and if, in case $X = S$, we also replace S by Y in $\mathfrak{G}$, then we get an equivalent SPG. (In other words: renaming an auxiliary symbol of $\mathfrak{G}$ does not change $L(\mathfrak{G})$.)

The lemma can easily be verified by induction on the length of generation trees (see Def. 1.1. c, with $z_0 = S$).

Theorem 3.1:

a) Every finite language is an SPL.

b) If L is an SPL, so is L^*.

c) If L_1 and L_2 are SPLs, so is $L_1 \cdot L_2$.

d) If L is an SPL, so is $cl(L)$.

e) If L_1 and L_2 are SPLs, so is $L_1 \cup L_2$.

Moreover, (1) given a finite language $L = \{x_1, \ldots, x_n\}$, an SPG representing L can be effectively constructed;

(2) given an SPG representing an SPL L, we can effectively construct SPGs representing L^* and $cl(L)$, respectively;

(3) given two SPGs representing two SPLs L_1 and L_2, we can effectively construct SPGs representing $L_1 \cdot L_2$ and $L_1 \cup L_2$, respectively.

Proof: a) Let $L = \{x_1, \ldots, x_n\}$ be a finite language over a vocabulary T. Take a new symbol $S \notin T$ and define: $\mathfrak{G} = (V, P, T, S)$, where $V = T \cup \{S\}$ and $P = \{S \to x_1, \ldots, S \to x_n\}$. Obviously $L(\mathfrak{G}) = L = \{x_1, \ldots, x_n\}$.

b) Let $L = L(\mathfrak{G})$, $\mathfrak{G} = (V, P, T, S)$. Define: $\mathfrak{G}^* = (V, P^*, T, S)$, where $P^* = \{X \to x^* \mid X \to x \in P\}$. Obviously, $L(\mathfrak{G}^*) = L(\mathfrak{G})^* = L^*$.

c) Let $L_i = L(\mathfrak{G}_i)$, $\mathfrak{G}_i = (V_i, P_i, T_i, S_i)$ $(i = 1,2)$.
Without loss of generality, assume that $(V_1 - T_1) \cap V_2 = V_1 \cap (V_2 - T_2) = \emptyset$; otherwise, by Lemma 3,1, replace some of the symbols of $V_1 - T_1$ and $V_2 - T_2$ by new auxiliary symbols. Take a symbol $S \notin V_1 \cup V_2$, and define

$$\mathfrak{G} = (V, P, T, S),$$

where $= V_1 \cup V_2 \cup \{S\}$, $P = P_1 \cup P_2 \cup \{S \to S_1 S_2\}$, $T = T_1 \cup T_2$. Obviously,

$$L(\mathfrak{G}) = L(\mathfrak{G}_1) \cdot L(\mathfrak{G}_2) = L_1 \cdot L_2.$$

d) Let $L = L(\mathfrak{G})$, $\mathfrak{G} = (V, P, T, S)$. Take a symbol $S_0 \notin V_1$ and define $\mathfrak{G}' = (V', P', T, S_0)$, where $V' = V \cup \{S_0\}$, $P' = P \cup \{S_0 \to SS_0, S_0 \to \Lambda\}$.

$$L(\mathfrak{G}') = cl(L(\mathfrak{G})) = cl(L).$$

as the reader will readily verify.

e) Let $L_i = L(\mathfrak{G}_i)$, $\mathfrak{G}_i = (V_i, P_i, T_i, S_i)$ $(i = 1,2)$. As in c), assume that $(V_1 - T_1) \cap V_2 = V_1 \cap (V_2 - T_2) = \emptyset$. Take a symbol $S \notin V_1 \cup V_2$, and define $\mathfrak{G} = (V, P, T, S)$, where $V = V_1 \cup V_2 \cup \{S\}$, $P = P_1 \cup P_2 \cup \{S \to S_1, S \to S_2\}$, $T = T_1 \cup T_2$. Obviously, $L(\mathfrak{G}) = L(\mathfrak{G}_1) \cup L(\mathfrak{G}_2) = L_1 \cup L_2$.

Remark: The class of FALs is also effectively closed under reflection and multiplication, as well as under the formation of closures and unions [R. S., Ths. 4, 5, 6, 12, 13]. However, whereas the class of FALs over a vocabulary T is effectively closed also under intersection and complementation with respect to W_T [R. S., Ths. 5, 6], this is not the case for SPLs, as shown by the following

Theorem 3.2: Let T be a vocabulary containing more than one symbol.

a) The class of SPLs over T is not closed under formation of intersections.

b) The class of SPLs over T is not closed under complementation with respect to W_T.

Proof: a) Let $T = \{0,1\}$, $L = \{0^n10^n10^k \mid n = 0, 1, 2, \ldots;\ k = 0, 1, 2, \ldots\}$. L is an SPL. In fact, define $\mathfrak{G} = (V, P, T, S)$, where $V = \{0, 1, S, A\}$, $T = \{0, 1\}$, $P = \{S \to S0,\ S \to A1,\ A \to 0A0,\ A \to 1\}$. Obviously. $L(\mathfrak{G}) = L$.

Now, $L^* = \{0^k10^n10^n \mid n = 0, 1, 2, \ldots;\ k = 0, 1, 2, \ldots\}$. L^* is an SPL too, by Th. 3.1. b. $L \cap L^* = \{0^n10^n10^n \mid n = 0, 1, 2, \ldots\}$. In order to prove that $L \cap L^*$ is not an SPL, we use Th. 4.1 of Sec. 4. Indeed, if $L \cap L^*$ were an SPL, then by Th. 4.1 we would have for a sufficiently large n a decomposition: $0^n10^n10^n = xuwvy$, with $u \neq \Lambda$ or $v \neq \Lambda$, and $xu^2wv^2y \in L \cap L^*$, i. e., $xuuwvvy = 0^m10^m10^m$, for some $m > n$. The duplicated substrings u and v can contain only zeros, so that the effect of the duplication is to increase the length of one or two of the blocks 0^n in $0^n10^n10^n$, while one block remains unchanged. Hence $xuuwvvy$ cannot have the form $0^m10^m10^m$ with three equal blocks 0^m, contrary to the requirement that $xuuwvvy \in L \cap L^*$. Therefore $L \cap L^*$ is not an SPL, which proves a).

b) The class of SPLs over T is closed under union (Th. 3.1. e). Since

$$L_1 \cap L_2 = W_T - [(W_T - L_1) \cup (W_T - L_2)],$$

if the class of SPLs over T were closed under complementation with respect to W_T, then it would also be closed under intersection, contrary to a).

Example: Let

$$T = \{0, 1\},\ L = \{0^n10^n10^k \mid n = 0, 1, 2, \ldots;\ k = 0, 1, 2, \ldots\}.$$

We shall construct an SPG $\mathfrak{G}$ representing $W_T - (L \cap L^*)$.

Define: $\mathfrak{G} = (V, P, T, S)$, where $V = \{0, 1, S, S_0, S_1, S_2\}$, $T = \{0, 1\}$ and

$$P = \left\{ \begin{array}{lll} 1.\ S \to S_0 & 6.\ S_1 \to 0S_1 & 11.\ S \to S_01S_20S_0 \\ 2.\ S \to S_01S_0 & 7.\ S_1 \to 1S_1 & 12.\ S \to S_01S_00S_2 \\ 3.\ S \to S_11S_11S_11S_1 & 8.\ S_1 \to \Lambda & 13.\ S_2 \to 0S_20 \\ 4.\ S_0 \to 0S_0 & 9.\ S \to S_00S_21S_0 & 14.\ S_2 \to 1 \\ 5.\ S_0 \to \Lambda & 10.\ S \to S_20S_01S_0 & \end{array} \right\}$$

Productions no. 1, 2, 4 and 5 yield all strings over T with less than two occurrences of 1. Productions no. 3, 6, 7 and 8 yield all strings over T with more than two occurrences of 1. Productions no. 9–14 yield all strings of the form $0^k10^l10^m$, where $k \neq l$ or $l \neq m$ (9, 13, 14 yield $k > l$, 10, 13, 14 yield $k < l$, 11, 13, 14 yield $l < m$, and 12, 13, 14 yield $l > m$.)

Therefore $L(\mathfrak{G}) = W_T - (L \cap L^*)$. Thus, $W_T - (L \cap L^*)$ is an SPL, while $W_T - (W_T - (L \cap L^*)) = L \cap L^*$ is not an SPL. as proved in Th. 3.2. a. (If we omit from P productions no. 11 and 12, or productions no. 9 and 10, we get a representation of $W_T - L$, or $W_T - L^*$, respectively.)

Theorem 3.3: (Closure under substitution.)

Let M be an SPL over a vocabulary U. Let a function φ assign to every symbol $X \in U$ an SPL $\varphi(X) = L_X$ over a vocabulary T_X. Let L be the set of *all* strings over $\bigcup_{X \in U} T_X$ obtained from sentences $u \in M$ by substituting a sentence of L_X for each occurrence of X in u, i. e.:

$$L = \left\{ y_{X_1} \cdots y_{X_k} \;\middle|\; \begin{array}{l} X_i \in U \text{ and } y_{X_i} \in L_{X_i} \text{ for } 1 \leq i \leq k, \\ 0 \leq k < \infty;\ X_1 \cdots X_k \in M \end{array} \right\}.$$

Then L is an SPL, and an SPG representing L can be effectively constructed from SPGs representing M and $L_X (X \in U)$.

Proof: Let $M = L(\mathfrak{G}_M)$, $\mathfrak{G}_M = (V_M, P_M, U, S_M)$, and let, for each $X \in U$:

$$L_X = L(\mathfrak{G}_X), \quad \mathfrak{G}_X = (V_X, P_X, T_X, S_X).$$

Assume, without loss of generality, that all the auxiliary vocabularies $V_M - U$ and $V_X - T_X$ $(X \in U)$, as well as $\bigcup_{X \in U} T_X$, are mutually exclusive; otherwise apply Lemma 3,1, as in the proof of Th. 3.1. b. (The terminal vocabularies $T_X (X \in U)$ may have elements in common. The vocabulary U itself will not appear explicitly in the SPG to be constructed, so that it may intersect the vocabularies $V_X (X \in U)$.)

Define: $\mathfrak{G} = (V, P, T, S)$, with $V = (V_M - U) \cup (\bigcup_{X \in U} V_X)$, $T = \bigcup_{X \in U} T_X$, and $P = P'_M \cap (\bigcup_{X \in U} P_X)$, where P'_M is obtained from P_M by substituting, in all productions, S_X – the initial symbol of $\mathfrak{G}_X$ – for every occurrence of $X \in U$.

Clearly, every sentence of L is generated by $\mathfrak{G}$: in order to generate $y_{X_i} \cdots y_{X_k}$, where $X_i \in U$, $y_{X_i} \in L_{X_i}$ $(1 \leq i \leq k)$ and $X_1 \cdots X_k \in M$, first generate $S_{X_1} \ldots S_{X_k}$ from S_M, applying P_M, then generate y_{X_i} from S_{X_i}, applying P_{X_i} $(1 \leq i \leq k)$.

Conversely, it is easily verified by induction on the length of generation trees in $\mathfrak{G}$, that for every $A \in V_M - U$, if $A \overset{*}{\Rightarrow} y$ by P and $y \in W_T$, then there is a decomposition $y = y_{X_1} \cdots y_{X_k}$, such that $X_i \in U$, $y_{X_i} \in L_{X_i}$ $(1 \leq i \leq k)$ and $A \overset{*}{\Rightarrow} X_1 \cdots X_k$ in $\mathfrak{G}_M$. By taking $A = S_M$, we see that every string over T generated by $\mathfrak{G}$ is a sentence of L. Therefore, $L(\mathfrak{G}) = L$.

In case the languages L_X $(X \in U)$ consist of single, possibly empty, strings, we get as a special case of Th. 3.3 the following

Corollary 3.1: Let T and T' be two vocabularies, and φ a function from T into $W_{T'}$. φ can be extended naturally to a function from W_T to $W_{T'}$, by defining: if $x = X_1 \cdots X_k \in W_T$, then $\varphi(x) = \varphi(X_1) \cdots \varphi(X_k)$.

If L is an SPL over T, then $\varphi(L)$ is an SPL over T', and an SPG representing $\varphi(L)$ can be effectively constructed from an SPG representing L.

Remark: Theorem 3.3 holds true if we replace throughout "SPL" by "FAL", and "SPG" by "FA".

Method of proof: a) First prove that if L is an FAL over a vocabularly T, and $X \in T$, then the language L' obtained from L by omitting all occurrences of the symbol X from all sentences of L is an FAL, and a FA representing L' can be effectively constructed from a FA representing L. This will be done most conveniently by first constructing a nondeterministic FA $\mathfrak{A}'$ representing L', then reducing $\mathfrak{A}'$ to an equivalent deterministic FA $\mathfrak{A}''$, by [R. S., Def. 11 and Th. 11]. (A similar result is proved in [C. M.].)

b) Let M be an FAL over U, and for each $X \in U$ let L_X be an FAL over T_X. Take new separating symbols A_X, one for each $X \in U$, all different and not contained in $\bigcup_{X \in U} T_X$. Given FAa representing M and L_X $(X \in U)$, construct a

FA $\mathfrak{B}$ which will accept exactly all strings of the form

$$A_{X_1} y_{X_1} A_{X_1} A_{X_2} y_{X_2} A_{X_2} \cdots A_{X_k} y_{X_k} A_{X_k},$$

where $X_i \in U$, $y_{X_i} \in L_{X_i}$ $(1 \leqq i \leqq k)$, and $X_1 \cdots X_k \in M$.

c) By repeated application of a) construct from $\mathfrak{B}$ a FA $\mathfrak{A}$ which will accept exactly the strings obtained from strings of the form

$$A_{X_1} y_{X_1} A_{X_1} A_{X_2} y_{X_2} A_{X_2} \cdots A_{X_k} y_{X_k} A_{X_k}$$
$$(X_i \in U,\ y_{X_i} \in L_{X_i} \quad (1 \leqq i \leqq k), \quad X_1 \cdots X_k \in M)$$

by omitting all occurrences of the separating symbols A_{X_i}. Clearly, $L(\mathfrak{A})$ is the language required in Th. 3.3.

In Section 8 we shall prove that the intersection of an SPL and an FAL is effectively an SPL.

Section 4: Some basic auxiliary properties of SPGs

Definition 4.1: Let $\mathfrak{G} = (V, P, T, S)$ be an SPG.

a) $\mathfrak{G}$ is called a 1-SPG if, for every production $X \to x \in P$, $l(x) \geqq 1$ (i. e., if P does not contain any production of the form $X \to \Lambda$).

b) $\mathfrak{G}$ is called a 2-SPG if, for every production $X \to x \in P$, $l(x) \geqq 2$ (i. e., if P does not contain any production of the form $X \to \Lambda$ or $X \to Y$, with $Y \in V$).

The following two lemmata show how to reduce any SPG to an almost equivalent 1-SPG, and any 1-SPG to an almost equivalent 2-SPG. These reductions will be used later on. In the definition of SPGs given in [B. G. S., § 2], all productions are required to be non-empty, i. e., of the form $X \to x$, $x \neq \Lambda$. Lemma 4.1 shows that the SPGs defined in Section 1 are completely equivalent to the SPGs defined in [B. G. S.], with the possible exception of the empty string Λ.

Lemma 4.1: Let $\mathfrak{G} = (V, P, T, S)$ be an SPG. It can be effectively decided whether $\Lambda \in L(\mathfrak{G})$, and a 1-SPG $\mathfrak{G}'$ can be constructed such that

$$L(\mathfrak{G}') = L(\mathfrak{G}) - \{\Lambda\}.$$

Proof: 1. We define an ascending chain of subsets of V:

$$V_1 = \{X \mid X \in V \,\&\, X \to \Lambda \in P\}$$

$V_{k+1} = V_k \cup \{X \mid X \in V \,\&\, (\exists x)\,[x \in W_{V_k} \,\&\, X \to x \in P]\}$ $\quad (k = 1, 2, \ldots)$, i. e., V_{k+1} contains V_k and all symbols which directly generate strings over V_k.

Obviously, $V_k \subseteqq V_{k+1}$ for all k, and if for some k $V_k = V_{k+1}$, then $V_k = V_l$ for all $l > k$, so that the chain remains constant after n steps at most, where n is the number of symbols in V. If $X \in V_n$, then $X \overset{*}{\Rightarrow} \Lambda$. Conversely, if $X \overset{*}{\Rightarrow} \Lambda$ then $X \in V_k$ for some k, and therefore $X \in V_n$. $\Lambda \in L(\mathfrak{G})$ iff $S \overset{*}{\Rightarrow} \Lambda$, i. e., iff $S \in V_n$.

2. Let $\mathfrak{G}' = (V, P', T, S)$, where P' is defined as follows: $X \to x \in P'$ iff $x \neq \Lambda$ and there is a production $X \to y \in P$, such that x is obtained from y by omission of some (possibly none) occurrences of symbols contained in V_n.

Obviously, if $X \overset{*}{\Rightarrow} x$ in $\mathfrak{G}'$, then $X \overset{*}{\Rightarrow} x$ in $\mathfrak{G}$, because the effect of every production of P' can be obtained by applying a finite number of productions of P. Conversely, if $X \overset{*}{\Rightarrow} x$ in $\mathfrak{G}$ and $x \neq \Lambda$, then $X \overset{*}{\Rightarrow} x$ in $\mathfrak{G}'$. This can easily be proved by induction on the lengths of the generation trees in $\mathfrak{G}$.

In particular, $S \overset{*}{\Rightarrow} x$ in $\mathfrak{G}'$ iff $x \neq \Lambda$ and $S \overset{*}{\Rightarrow} x$ in $\mathfrak{G}$. Therefore,

$$L(\mathfrak{G}') = L(\mathfrak{G}) - \{\Lambda\}.$$

Lemma 4.2: Let $\mathfrak{G} = (V, P, T, S)$ be a 1-SPG. For all $X \in T$ it can be effectively decided whether $X \in L(\mathfrak{G})$, and a 2-SPG $\mathfrak{G}'$ can be constructed, such that $L(\mathfrak{G}') = L(\mathfrak{G}) - T$ (i. e., $x \in L(\mathfrak{G}')$ iff $x \in L(\mathfrak{G})$ and $l(x) \geq 2$).

Proof: 1. We define ascending chains of subsets of V: For each $X \in V$,

$$V_1(X) = \{X\},$$

$$V_{k+1}(X) = V_k(X) \cup \{Y \mid Y \in V \,\&\, (\exists Z)\,[Z \in V_k(X) \,\&\, Z \to Y \in P]\} \quad (k = 1, 2, \ldots).$$

i. e., $V_{k+1}(X)$ contains besides $V_k(X)$ all symbols which are directly generated by symbols of $V_k(X)$.

By an argument similar to that used in the proof of lemma 4.1 we have:

$$V_1(X) \subseteq V_2(X) \subseteq \cdots \subseteq V_n(X) = V_{n+1}(X) = V_{n+2}(X) = \cdots,$$

where n is the number of symbols of V.

$\mathfrak{G}$ is, by assumption, a 1-SPG, so that P contains no length-decreasing productions of the form $X \to \Lambda$. Therefore $Y \in V_n(X)$ iff $X \overset{*}{\Rightarrow} Y$. $X \in L(\mathfrak{G})$ iff $S \overset{*}{\Rightarrow} X$ and $X \in T$, i. e., iff $X \in T \cap V_n(S)$.

2. Let

$$P' = \{X \to x \mid l(x) \geq 2 \,\&\, (\exists Y)\,[Y \in V_n(X) \,\&\, Y \to x \in P]\}.$$

Define P'' as follows:

$X \to Y_1 \cdots Y_r \in P''$, iff there is a production $X \to X_1 \cdots X_r \in P'$, such that $Y_j \in V_n(X_j)$ for $1 \leq j \leq r$. Let $\mathfrak{G}' = (V, P'', T, S)$.

Obviously, if $X \overset{*}{\Rightarrow} x$ in $\mathfrak{G}'$, then $X \overset{*}{\Rightarrow} x$ in $\mathfrak{G}$. Conversely, if $X \overset{*}{\Rightarrow} x$ in $\mathfrak{G}$ and $l(x) \geq 2$, then $X \overset{*}{\Rightarrow} x$ in $\mathfrak{G}'$. This can easily be proved by induction on the lengths of the generation trees in $\mathfrak{G}$.

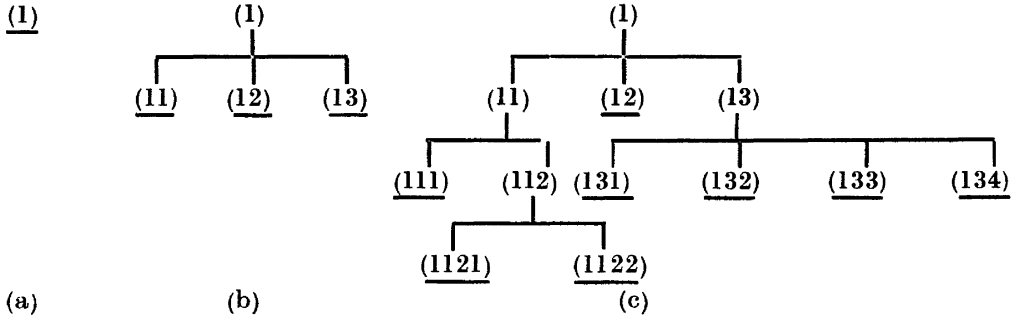
In particular, $S \overset{*}{\Rightarrow} x$ in $\mathfrak{G}'$ iff $l(x) \geq 2$ and $S \overset{*}{\Rightarrow} x$ in $\mathfrak{G}$. Therefore,

$$L(\mathfrak{G}') = L(\mathfrak{G}) - T.$$

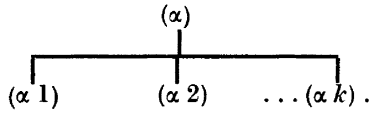
Theorem 4.1: Let $\mathfrak{G} = (V, P, T, S)$ be a 2-SPG. It is possible to determine two natural numbers p and q such that every sentence z of $L(\mathfrak{G})$ with $l(z) > p$ admits a decomposition of the form $z = xuwvy$, where $u \neq \Lambda$ or $v \neq \Lambda$, $l(uwv) \leq q$, and all the strings

$$z_k = xu^k wv^k y \in L(\mathfrak{G}) \qquad (k = 1, 2, 3, \ldots).$$

Proof: The proof relies on a graphic representation of generation trees. By a *tree* we mean a "branching pattern" with a special notation of nodes. The following graphs are examples of trees:



In general, taking in a given tree a *terminal node* (in the examples, the underlined nodes are terminal) which is denoted by the multi-index (α), we obtain a new tree by adding to the old one the "branch"



Each one of the nodes $(\alpha\ 1)$, $(\alpha\ 2)$, . . ., $(\alpha\ k)$ is called *consecutive* to (α). A $\mathfrak{G}$-*tree* is obtained from a tree by attaching a symbol of V to each node of the tree, so that when A is attached to (α), B_j to $(\alpha\ j)$ $(1 \leqq j \leqq k)$, and these $(\alpha\ j)$ are all the consecutive nodes of (α), then $A \to B_1\ B_2 \cdots B_k \in P$.

Note that a tree does not always yield a $\mathfrak{G}$-tree by a suitable attachment of symbols. For instance: in all the $\mathfrak{G}$-trees the components of the multi-indices cannot be greater than the length of the longest production in P.

The node (1) is the vertex of the tree, and the *vertex-symbol* of a $\mathfrak{G}$-tree is the symbol attached to the vertex. The *final string* of a $\mathfrak{G}$-tree is the string obtained by arranging the symbols attached to the terminal nodes in the lexicographic order of the multi-indices. (E. g., the order of the terminal nodes in the tree (c) is: (111), (1121), (1122), (12), (131), (132), (133), (134).)

A *path* in a tree is a sequence of consecutive nodes starting with the vertex and terminating in a terminal node. (E. g., (1), (11), (112), (1121) is a path in (c).) The sequence of symbols attached to the nodes of a path gives a $\mathfrak{G}$-path. Note that every terminal node determines a unique path.

Each node (α) in a tree determines, in a natural manner, a unique *subtree* of the given tree. This subtree contains (α) (as its vertex), the consecutive nodes of (α), the consecutive nodes of these nodes, and so on. (Properly speaking, a subtree is not a tree, but becomes a tree when we rename the node (α) by (1) and the other nodes correspondingly.) In a similar manner, a node in a $\mathfrak{G}$-tree determines a $\mathfrak{G}$-*subtree*. The final string of a $\mathfrak{G}$-subtree is a substring of the final string of the $\mathfrak{G}$-tree.

Now it is easily verified that the vertex-symbol of a $\mathfrak{G}$-tree generates the final string (in the sense of Def. 1.1 c). Conversely, if $A \overset{*}{\Rightarrow} x$ in $\mathfrak{G}$, then a $\mathfrak{G}$-tree with vertex-symbol A and final string x can be constructed. This justifies the assertion that $\mathfrak{G}$-trees are graphic representations of generations in $\mathfrak{G}$.

After introducing the above notions, the actual proof of the theorem runs as follows: Let n be the number of symbols in V, and consider the $\mathfrak{G}$-trees in which all the paths are of length n at most. The number of all such $\mathfrak{G}$-trees is obviously finite and the lengths of their final strings are bounded. Call this bound p (p can easily be estimated). Every sentence z of $L(\mathfrak{G})$ with $l(z) > p$ is then the final string of a $\mathfrak{G}$-tree with vertex-symbol S, which contains a path longer than n. Let $A_1, A_2, \ldots, A_r$ $(r > n)$ be a longest $\mathfrak{G}$-path in that $\mathfrak{G}$-tree. In the sequence $A_{r-n}, \ldots, A_r$, at least one symbol must occur twice. Suppose $A_i = A_j = W$ $(r - n \leqq i < j \leqq r)$. Consider the $\mathfrak{G}$-subtree whose vertex-symbol is $W = A_i$ (i. e., the $\mathfrak{G}$-subtree determined by the node to which $W = A_i$ is attached). Since $W = A_j$ is attached to one of its nodes, the generation corresponding to this $\mathfrak{G}$-subtree has the form:

$$W \overset{*}{\Rightarrow} u' Wv' \overset{*}{\Rightarrow} uwv,$$

where, since we deal with a 2-SPG, u' or v' (and so u or v) $\neq \Lambda$. Here uwv is the final string of the $\mathfrak{G}$-subtree, hence it is a substring of z, the final string of the original $\mathfrak{G}$-tree. This means that z can be decomposed into the form $z = xuwvy$.

Now the generations $W \overset{*}{\Rightarrow} u'Wv'$, $u' \overset{*}{\Rightarrow} u$, $v' \overset{*}{\Rightarrow} v$ can be repeated k times. We obtain $W \overset{*}{\Rightarrow} (u')^k W (v')^k \overset{*}{\Rightarrow} u^k w v^k$, and from the original $\mathfrak{G}$-tree we obtain $S \overset{*}{\Rightarrow} xu^k w v^k y \in L(\mathfrak{G})$ $(k = 1, 2, 3, \ldots)$.

Finally note that $A_{r-n}, \ldots, A_r$ is a longest path of the $\mathfrak{G}$-subtree – hence the length of all its paths is $\leqq n + 1$; thus, with a suitable bound q, we have for its final string: $l(uwv) \leqq q$.

The theorem just proved (combined with Lemma 4.2) is very useful in showing that various languages are *not* SPLs, and was already used in the proof of Theorem 3.2.

Section 5: Solvable decision problems

In this section we shall exhibit decision procedures for two basic properties of SPGs, namely the emptiness and the infiniteness of $L(\mathfrak{G})$, as well as for the basic generation relations of strings in an SPG.

One way of obtaining decision procedures is to consider the graphic representation of generation trees, which was already used in proving Theorem 4.1. This way yields sometimes more economic procedures. However, we preferred the method of ascending chains of sets (of strings or of symbols), which allows for a more uniform and rigorous presentation. This last method was already used in the proofs of Lemmata 4.1 and 4.2.

In general, it is more convenient first to present decision procedures for 1-SPGs or even 2-SPGs, and afterwards use Lemmata 4.1 and 4.2 in order to reduce an arbitrary SPG to one of these special kinds of SPGs.

Theorem 5.1: Let $\mathfrak{G} = (V, P, T, S)$ be a 1-SPG. For $A \in V$ and $x \in W_T$, the following relations are decidable:

a) $A \overset{*}{\Rightarrow} x$
b) $(\exists u)(\exists v)(A \overset{*}{\Rightarrow} uxv)$
c) $(\exists u)(A \overset{*}{\Rightarrow} ux)$ and, similarly, $(\exists v)(A \overset{*}{\Rightarrow} xv)$
d) $(\exists u)(\exists v)(u \neq \Lambda,\ v \neq \Lambda\ \&\ A \overset{*}{\Rightarrow} uxv)$
e) $(\exists u)(u \neq \Lambda\ \&\ A \overset{*}{\Rightarrow} ux)$ and, similarly, $(\exists v)(v \neq \Lambda\ \&\ A \overset{*}{\Rightarrow} xv)$.

Proof: We prove b) first. Suppose that V contains n symbols and let $l(x) = m$. We construct inductively the following sets of strings:

$$A_m^1 = \{z \mid l(z) \leqq m \;\&\; (\exists u)(\exists v)(A \Rightarrow uzv)\}$$
$$A_m^k = \{z \mid l(z) \leqq m \;\&\; [z \in A_m^{k-1} \vee (\exists u)(\exists v)(\exists y)(y \in A_m^{k-1} \;\&\; y \Rightarrow uzv)]\}.$$

The relation $R(x, y) = (\exists u)(\exists v)(y \Rightarrow uxv)$, on which the definition of the A_m^k is based, is clearly decidable, since $\Rightarrow$ denotes *direct* generation. Thus the sets $A_m^1, A_m^2, A_m^3, \ldots$ can be effectively constructed one after the other. By definition we have $A_m^{k-1} \subseteqq A_m^k$. Also, $A_m^{k-1} = A_m^k$ implies $A_m^k = A_m^{k+1}$. Indeed, a z with $l(z) \leqq m$ belongs to A_m^{k+1} if $z \in A_m^k$ or $y \in A_m^k$ and $y \Rightarrow uzv$, for suitable y, u, and v. By assumption, y already belongs to A_m^{k-1}, hence, in any case, $z \in A_m^k$.

Thus the sequence of sets $A_m^1, A_m^2, \ldots$ strictly increases until an equality is obtained. But the total number of non-empty strings not longer than m is $\beta = \dfrac{n^{m+1} - 1}{n - 1} - 1$, hence the sequence must stop increasing when (or before) A_m^β is reached. In other words, $\lim\limits_{k \to \infty} A_m^k = A_m^\beta$.

Now it is readily seen that:

$$(\exists u)(\exists v)(A \overset{*}{\Rightarrow} uxv) \leftrightarrow [(A = x) \vee (A = x) \vee (\exists k)(x \in A_m^k)] \leftrightarrow [(A = x) \vee x \in A_m^\beta]$$

A decision procedure for b) would therefore be: Check whether $x = A$. If not, construct A_m^β and check whether $x \in A_m^\beta$.

The proofs of a and c are obtained analogously, by suitable modifications in the definition of A_m^k. As for d, note that d holds if and only if

$$(\exists u)(\exists v)(A \overset{*}{\Rightarrow} uDxEv)$$

holds for at least one pair of symbols $D, E \in V$. A similar remark proves e.

Corollary: Let $\mathfrak{G}$ be an SPG. The property of sentencehood: $x \in L(\mathfrak{G})$ is decidable.

Proof: For a 1-SPG, $x \in L(\mathfrak{G})$ iff $S \overset{*}{\Rightarrow} x$ and $x \in W_T$, thus the corollary follows from part a of the theorem. For an arbitrary SPG $\mathfrak{G}$, we construct by Lemma 4.1 a 1-SPG $\mathfrak{G}'$ which generates exactly the $x \in L(\mathfrak{G})$ with $l(x) \geqq 1$. Now $\Lambda \in L(\mathfrak{G})$ is decidable (again by Lemma 4.1) and a non-empty $x \in L(\mathfrak{G})$ iff $x \in L(\mathfrak{G}')$.

Next, we shall prove the solvability of the emptiness problem.

Theorem 5.2: Given an SPG $\mathfrak{G} = (V, P, T, S)$, we can effectively decide whether $L(\mathfrak{G})$ is empty.

Proof: Let:

$$T_0 = T, \quad T_j = T_{j-1} \cup \{X \in V \mid (\exists x)(x \in W_{T_{j-1}} \;\&\; X \Rightarrow x)\}.$$

Thus T_j includes T_{j-1} and contains also all the symbols that directly generate strings over T_{j-1}. Clearly, T_j can be effectively constructed from T_{j-1}; $T_{j-1} = T_j$ implies $T_j = T_{j+1}$. If $V\text{-}T$ contains r symbols, the ascending chain $T_0 \subseteqq T_1 \subseteqq T_2 \cdots$ strictly increases r steps at most, and we have

$$T_r = \lim_{k \to \infty} T_k.$$

Since, clearly,

$$(\exists x)\,[x \in L(\mathfrak{G})] \leftrightarrow (\exists x)\,[S \overset{*}{\Rightarrow} x \,\&\, x \in W_T] \leftrightarrow (\exists k)\,[S \in T_k] \leftrightarrow S \in T_r,$$

it follows that $L(\mathfrak{G})$ is empty iff $S \notin T_r$. This provides an effective procedure.

The constructions in the proofs of Theorems 5.1 and 5.2 enable us to present an SPG in a "reduced" form, in which superfluous symbols have been eliminated. We first define:

Definition 5.1: An SPG $\mathfrak{G} = (V, P, T, S)$ is called *reduced* if $L(\mathfrak{G}) = \emptyset$ and $V = \{S\}$, $P = T = \emptyset$; or in case $L(\mathfrak{G}) \neq \emptyset$, if the following two conditions are satisfied:

(i) for every $A \in V$, there are strings u and v such that $S \overset{*}{\Rightarrow} uAv$;

(ii) for every $A \in V - T$, there is a string $t \in W_T$ such that $A \overset{*}{\Rightarrow} t$.

Condition (i) means that every symbol of V actually occurs in some generation from S. Condition (ii) means that every auxiliary symbol generates a terminal string.

Lemma 5.1: Let $\mathfrak{G} = (V, P, T, S)$ be an SPG. It is possible to construct a reduced SPG $\overline{\mathfrak{G}}$ equivalent to $\mathfrak{G}$, such that in $\overline{\mathfrak{G}} = (\overline{V}, \overline{P}, \overline{T}, S)$

$$\overline{V} \subseteq V, \quad \overline{P} \subseteq P, \quad \overline{T} \subseteq T.$$

Proof: If $L(\mathfrak{G}) \neq \emptyset$ (this is decidable, by Th. 5.2), take $\overline{V} = \{S\}$, $\overline{P} = \overline{T} = \emptyset$. If $L(\mathfrak{G}) \neq \emptyset$, we obtain $\overline{\mathfrak{G}}$ in two steps. First we delete from V the symbols which do not generate terminal strings, i. e., the symbols not contained in the set T_r constructed in the proof of Th. 5.2. We denote by P' the set of productions of P which contain only symbols of T_r. Then $\mathfrak{G}' = (T_r, P', T, S)$ is clearly equivalent to $\mathfrak{G}$. $\mathfrak{G}'$ fulfils condition (ii), since $T_r \ni A \overset{*}{\Rightarrow} t$ in $\mathfrak{G}$ implies that all the symbols in the generation tree from A to t belong to T_r, hence $A \overset{*}{\Rightarrow} t$ in $\mathfrak{G}'$.

Next we delete from T_r the symbols which are not generated from S in $\mathfrak{G}'$ (i. e., by the productions of P'). These are the symbols not contained in the set S_1^n constructed in the proof of Th. 5.1. (Note that here $\lim_{k\to\infty} S_1^k = S_1^n$ where n is the number of symbols in T_r.)

Now, if $\overline{V} = S_1^n$ (note that $S_1^n \subseteq T_r \subseteq V$), $\overline{T} = \overline{V} \cap T$ and $\overline{P}$ is the set of productions of P' which contain only symbols of $\overline{V}$, then $\overline{\mathfrak{G}} = (\overline{V}, \overline{P}, \overline{T}, S)$ is the required SPG. Indeed $\overline{\mathfrak{G}}$ is equivalent to $\mathfrak{G}'$, hence also to $\mathfrak{G}$, and it is easily verified that it fulfils condition (i). Regarding condition (ii), note that $S \overset{*}{\Rightarrow} \cdots A \cdots$ in $\mathfrak{G}'$ and $A \overset{*}{\Rightarrow} t$ in $\mathfrak{G}'$ imply that all the symbols in the generation tree from A to t also belong to $S_1^n = \overline{V}$. Hence $A \overset{*}{\Rightarrow} t$ in $\overline{\mathfrak{G}}$, Q. E. D.

Definition 5.2: Let $\mathfrak{G} = (V, P, T, S)$ be an SPG, and let V contain n symbols. A symbol $A \in V$ is called an *embedding symbol* if $A \in A_1^n$. $\mathfrak{G}$ is called an *embedding* SPG, if some symbol $A \in V$ is embedding.

In other words, A is embedding iff there exists a non-trivial generation $A \overset{*}{\Rightarrow} uAv$. Obviously the property of being embedding is effectively decidable, for both symbols and SPGs.

Theorem 5.3: (i) Let $\mathfrak{G}$ be a reduced 2-SPG. $L(\mathfrak{G})$ is infinite iff $\mathfrak{G}$ is embedding. (This does not hold true if $\mathfrak{G}$ is not reduced or is not a 2-SPG.)

(ii) Given an SPG $\mathfrak{G}$, we can effectively decide whether $L(\mathfrak{G})$ is infinite.

Proof: (i) If $\mathfrak{G}$ is an embedding 2-SPG, we have $A \overset{*}{\Rightarrow} uAv$, for a suitable A, with u or $v \neq \Lambda$. If $\mathfrak{G}$ is also reduced, we have $S \overset{*}{\Rightarrow} xAy$, for every A and for suitable x and y. Combining these generations, we have:

$$S \overset{*}{\Rightarrow} xAy \overset{*}{\Rightarrow} xuAvy \overset{*}{\Rightarrow} xu^2Av^2y \overset{*}{\Rightarrow} \cdots \overset{*}{\Rightarrow} xu^kAv^ky \overset{*}{\Rightarrow} \cdots$$

Using again the fact that $\mathfrak{G}$ is reduced, we have:

$$x \overset{*}{\Rightarrow} x',\ y \overset{*}{\Rightarrow} y'\quad u \overset{*}{\Rightarrow} u',\ v \overset{*}{\Rightarrow} v',\ A \overset{*}{\Rightarrow} a,$$

where x', y', u', v' and a are suitable strings over T. Hence

$$S \overset{*}{\Rightarrow} x'(u')^k a(v')^k y',$$

and the right-hand side is a string over T, i.e., it belongs to $L(\mathfrak{G})$, for every $k = 1, 2, 3, \ldots$. We have thus proved that $L(\mathfrak{G})$ is infinite.

Conversely, if none of the symbols of V is embedding, only a finite number of strings are generated from S. (This can also be shown by considering the $\mathfrak{G}$-trees appearing in the proof of Theorem 4.1. For non-embedding $\mathfrak{G}$, the paths of $\mathfrak{G}$-trees are of length $\leq n$, hence the number of different $\mathfrak{G}$-trees, and a fortiori the number of final strings in $\mathfrak{G}$-trees, is finite.) In particular, $L(\mathfrak{G})$ is finite. Note that this converse holds also for any SPG.

(ii) For a 2-SPG, part (ii) follows from (i) and the remark that embedding is a decidable property. For an arbitrary SPG $\mathfrak{G}$, we construct, by Lemmata 4.1, 4.2 and 5.1, a reduced 2-SPG $\overline{\mathfrak{G}}$ which is equivalent to $\mathfrak{G}$, except for a finite number of strings of length ≤ 1. Hence, the infiniteness of $L(\mathfrak{G})$ is equivalent to that of $L(\overline{\mathfrak{G}})$, and therefore decidable.

Section 6: Undecidable properties of SPGs

In this section we prove certain properties of SPGs and relations between SPGs to be effectively *undecidable* (Theorems 6.1, 6.2 and 6.3). We proceed by reducing a known unsolvable decision problem to the decision problems of those properties and relations. This unsolvable problem is the following one, known as

Post's correspondence problem:

Let $(a_1, \ldots, a_n)$, $(b_1, \ldots, b_n)$ be two n-tuples of non-empty strings over a vocabulary V. Does there exist a sequence of indices $i_1, \ldots, i_k$, where $k \geq 1$ and $1 \leq i_j \leq n$ for $j = 1, \ldots, k$, such that $a_{i_1} \cdots a_{i_k} = b_{i_1} \cdots b_{i_k}$?

Post proved in [P.] that this correspondence problem is not effectively solvable, provided that V contains more than one symbol. Since we shall make constant use of Post's correspondence problem and his result, we introduce the following abbreviatory notation.

Definition 6.1: Let (a, b) be a pair of n-tuples of non-empty strings over $\{0,1\}$, $a = (a_1, \ldots, a_n)$, $b = (b_1, \ldots, b_n)$, $n \geq 1$. If there exists a sequence $i_1, \ldots, i_k$ of indices, where $k \geq 1$ and $1 \leq i_j \leq n$ for $j = 1, \ldots, k$, such that $a_{i_1} \cdots a_{i_k} = b_{i_1} \cdots b_{i_k}$, then we write $\mathcal{P}(a, b) = 1$; if there exists no such sequence, we write $\mathcal{P}(a, b) = 0$.

Theorem 6.1: The following decision problems are effectively unsolvable: Given two SPGs $\mathfrak{G}_1$ and $\mathfrak{G}_2$,

a) is $L(\mathfrak{G}_1) \cap L(\mathfrak{G}_2)$ empty?

b) is $L(\mathfrak{G}_1) \cap L(\mathfrak{G}_2)$ finite?

c) is $L(\mathfrak{G}_1) \cap L(\mathfrak{G}_2)$ an FAL?

d) is $L(\mathfrak{G}_1) \cap L(\mathfrak{G}_2)$ an SPL?

Moreover, these problems are unsolvable even in the special case that $\mathfrak{G}_1$ and $\mathfrak{G}_2$ have a common terminal vocabulary T which contains two symbols only, and they remain unsolvable for $\mathfrak{G}_1$, even if $\mathfrak{G}_2$ is held fixed to be a certain SPG $\bar{\mathfrak{G}}_s$ defined below.

Theorem 6.2: The following decision problems are effectivly unsolvable: Given an SPG $\mathfrak{G}$ with a terminal vocabulary T,

a) is $W_T - L(\mathfrak{G})$ empty?

b) is $W_T - L(\mathfrak{G})$ finite?

c) is $W_T - L(\mathfrak{G})$ an FAL?

d) is $W_T - L(\mathfrak{G})$ an SPL?

These problems are unsolvable even if T is restricted to contain two symbols only.

Theorem 6.3: The following decision problems are effectively unsolvable:

a) (Inclusion) Given two SPGs $\mathfrak{G}_1$ and $\mathfrak{G}_2$, is $L(\mathfrak{G}_1) \subseteq L(\mathfrak{G}_2)$?

This problem remains unsolvable for $\mathfrak{G}_2$, if $\mathfrak{G}_1$ is held fixed to be a certain SPG $\mathfrak{G}_W$ defined below, and remains unsolvable for $\mathfrak{G}_1$, if $\mathfrak{G}_2$ is held fixed to be a certain SPG $\bar{\mathfrak{G}}'_s$ defined below.

b) (Equivalence) Given two SPGs $\mathfrak{G}_1$ and $\mathfrak{G}_2$, is $L(\mathfrak{G}_1) = L(\mathfrak{G}_2)$?

This problem remains unsolvable for $\mathfrak{G}_1$, if $\mathfrak{G}_2$ is held fixed to be a certain SPG $\mathfrak{G}_W$ defined below.

c) Given an SPG $\mathfrak{G}$, is $L(\mathfrak{G})$ an FAL?

d) Given an SPG $\mathfrak{G}$ and a FA $\mathfrak{A}$, is $L(\mathfrak{G}) = L(\mathfrak{A})$?

This problem remains unsolvable for $\mathfrak{G}$, if $\mathfrak{A}$ is held fixed to be a certain FA $\mathfrak{G}_W$ defined below.

These four problems are unsolvable even in the special case that the terminal vocabularies of $\mathfrak{G}_1$, $\mathfrak{G}_2$ and $\mathfrak{G}$ and the vocabulary of $\mathfrak{A}$ contain each two symbols only.

The assertions of Theorem 6.3 will be derived as corollaries from Theorems 6.1 and 6.2. The proof of Theorem 6.3 will be given after the proofs of Theorems 6.1 and 6.2.

The reduction of Post's correspondence problem to the decision problems mentioned in Theorems 6.1 and 6.2 will require several auxiliary definitions and lemmata. Lemmata 6.1, 6.3, and 6.6 will be used in the proofs of both Theorem 6.1 and 6.2. Lemmata 6.2, 6.4 and 6.5 are needed for the proof of Theorem 6.2 only.

If we are contented to prove Theorems 6.1, 6.2 and 6.3 only for terminal vocabularies T containing more than two symbols, then we can dispense with Def. 6.5 and Lemma 6.5, and replace Lemma 6.6 by a simplified variant thereof (see Remark b) following the proof of Lemma 6.6).

The construction given here (Defs. 6.2, 6.3 and 6.4) is closely related to that used in [R. S., Theorem 18] for proving the unsolvability of the empty-intersection problem for TTAa.

Definition 6.2: Let $\xi = (x_1, \ldots, x_n)$ be an n-tuple of strings. Define:

$$L(\xi) = \{10^{i_k} \cdots 10^{i_1} E\, x_{i_1} \cdots x_{i_k} \mid k \geq 1, 1 \leq i_\nu \leq n \text{ for } \nu = 1, \ldots, k\}$$

(where 0^i, we recall, is the string $0 \ldots 0$ consisting of i zeros).

If we look upon the string 10^i as a code for the natural number i and denote it by $\hat{i}$, then we have:

$$L(\xi) = \{\hat{i}_k \cdots \hat{i}_1\, E\, x_{i_1} \cdots x_{i_k} \mid k \geq 1,\ 1 \leq i_\nu \leq n \text{ for } \nu = 1, \ldots, k\}.$$

Definition 6.3: Let (a, b) be a pair of n-tuples of non-empty strings over $\{0, 1\}$, $a = (a_1, \ldots, a_n)$, $b = (b_1, \ldots, b_n)$. Define:

$$L(a, b) = L(a) \cdot E \cdot L(b)^*, \text{ i. e.,}$$

$$L(a, b) = \left\{ 10^{i_k} \cdots 10^{i_1} E\, a_{i_1} \cdots a_{i_k} E\, b^*_{j_1} \cdots b^*_{j_1} E\, 0^{j_1} 1 \cdots 0^{j_1} 1 \;\middle|\; \begin{array}{l} k \geq 1,\ l \geq 1, \\ 1 \leq i \leq n \text{ for } \nu = 1, \ldots, k \\ 1 \leq j \leq n \text{ for } \nu = 1, \ldots, l \end{array} \right\}$$

(where x^*, we recall, is the reflection of x).

Lemma 6.1: $L(a, b)$ is an SPL, represented by the SPG:

$$\mathfrak{G}(a, b) = (V, P(a, b), T, S),$$

where

$$V = \{S, A, B, 0, 1, E\},\ T = \{0, 1, E\}$$

$$P(a, b) = \left\{ \begin{array}{ll} S \to AEB, & \\ A \to 10^i A a_i,\ A \to 10^i E a_i & (i = 1, \ldots, n) \\ B \to b^*_j\, B\, 0^j 1,\ B \to b^*_j\, E\, 0^j 1 & (j = 1, \ldots, n) \end{array} \right\}$$

Proof: Obvious.

Lemma 6.2: $W_T - L(a, b)$ is an SPL, represented by the SPG $\mathfrak{G}'(a, b) = (V', P'(a, b), T, S)$, where

$$V' = \{S, S_1, S_2, A, B, 0, 1, E\}, \qquad T = \{0, 1, E\},$$

$$P'(a, b) = P_1 \cup P_2 \cup P_3 \cup P_4 \cup P_5 \cup P_6 \cup P_7(n) \cup P_8(a) \cup P_9(b),$$

and $P_1 = \{S_1 \to \Lambda, S_1 \to 0, S_1 \to 1, S_1 \to 0\, S_1, S_1 \to 1\, S_1\}$,

$P_2 = \{S_2 \to \Lambda, S_2 \to 0, S_2 \to 1, S_2 \to E, S_2 \to 0\, S_2, S_2 \to 1\, S_2, S_2 \to E\, S_2\}$,

$P_3 = \{S \to S_1, S \to S_1\, E\, S_1, S \to S_1\, E\, S_1\, E\, S_1\}$,

$P_4 = \{S \to S_2\, E\ S_2\, E\, S_2\, E\, S_2\, E\, S_2\}$,

$P_5 = \{S \to E\, S_2, S \to S_2\, E, S \to S_2\, E\, E\, S_2, S \to 0\, S_2, S \to S_2\, O\}$,

$P_6 = \{S \to S_1\, 1\, E\, S_2, S \to S_2 E\, 1\, S_1, S \to S_1\, 11\, S_1 E\, S_2, S \to S_2\, E\, S_1\, 11\, S_1\}$,

$P_7(n) = \{S \to S_1\, 0^{n+1}\, S_1\, E\, S_2, S \to S_2\, E\, S_1\, 0^{n+1}_{} S_1\}$

(where n, we recall, is the length of the sequences a and b),

$$P_8(a) = \left\{ \begin{array}{l} S \to AES_2,\ A \to 10^i A\, a_i\ (i = 1, \ldots, n), \\ A \to ES_1 0,\ A \to E S_1 1,\ A \to 1 S_1 E, \\ A \to 10^i E\, \bar{a}_i,\ A \to 10^i 1 S_1 E\, \bar{a}_i\ (i = 1, \ldots, n), \\ A \to 10^i E S_1 \bar{\bar{a}}_i,\ A \to 10^i 1 S_1 E S_1 \bar{\bar{a}}_i\ (i = 1, \ldots, n), \\ \text{where } \bar{a}_i \text{ runs through all strings of } W_{\{0,1\}}, \text{ such that} \\ 1 \leqq l(\bar{a}_i) < l(a_i), \text{ and } \bar{\bar{a}}_i \text{ runs through all strings of} \\ W_{\{0,1\}}, \text{ such that } l(\bar{\bar{a}}_1) = l(a_1), \text{ except } a_i \text{ itself.} \end{array} \right\},$$

$$P_9(b) = \left\{ \begin{array}{l} S \to S_2 E B,\ B \to b_j^* B 0^j 1\ (j = 1, \ldots, n), \\ B \to 0 S_1 E,\ B \to 1 S_1 E,\ B \to E S_1 1, \\ B \to \bar{b}_j^* E 0^j 1,\ B \to \bar{b}_j^* E S_1 10^j 1\ (j = 1, \ldots, n), \\ B \to \bar{\bar{b}}_j^* S_1 E 0^j 1,\ B \to \bar{\bar{b}}_j^* S_1 E S_1 10^j 1\ (j = 1, \ldots, n), \\ \text{where } \bar{b}_i^* \text{ runs through all strings of } W_{\{0,1\}}, \text{ such that} \\ 1 \leqq l(\bar{b}_j^*) < l(b_j) \text{ and } \bar{\bar{b}}_j^* \text{ runs through all strings of} \\ W_{\{0,1\}}, \text{ such that } l(\bar{\bar{b}}_j) = l(b_j^*), \text{ except } b_j^* \text{ itself.} \end{array} \right\}.$$

Proof: From S_1 we get by P_1 every string over $\{0,1\}$. From S_2 we get by P_2 every string over T. By $P_1 \cup P_3$ we get all strings over T containing less than 3 E's. By $P_2 \cup P_4$ we get all strings over T containing more than 3 E's. By $P_2 \cup P_5$ we get all strings over T beginning or ending with E or 0, or containing two consecutive E's. By $P_1 \cup P_2 \cup P_6$ we get all strings over T with two consecutive 1's before the first E or after the last E, or with a 1 immediately preceding the first E or immediately following the last E. By $P_1 \cup P_2 \cup P_7(n)$ we get all strings over T with more than n consecutive zeros before the first E or after the last E.

Up to now we have obtained by $P_1 \cup P_2 \cup P_3 \cup P_4 \cup P_5 \cup P_6 \cup P_7(n)$ all strings over T, except those of the form

$$10^{i_k} \ldots 10^{i_1} ExEyE\, 0^{j_1} 1 \ldots 0^{j_e} 1, \tag{1}$$

where $x \neq \Lambda$, $y \neq \Lambda$, $x \in W_{\{0,1\}}$, $y \in W_{\{0,1\}}$, $k \geqq 1$, $l \geqq 1$, $1 \leqq i_\nu \leqq n$ for $\nu = 1, \ldots, k$, and $1 \leqq j_\nu \leqq n$ for $\nu = 1, \ldots, l$.

$P_1 \cup P_2 \cup P_8(a)$ generates, among others, all strings of the form (1) for which $x \neq a_{i_1} \ldots a_{i_k}$. $P_1 \cup P_2 \cup P_9(b)$ generates, among others, all strings of the form (1) for which $y^* \neq b_{j_1} \ldots b_{j_l}$.

Thus, every string of $W_T - L(a, b)$ can be generated from S by $P'(a, b)$, and it is easily verified that no string over T generated by $P'(a, b)$ from S belongs to $L(a, b)$.

Hence, $L(\mathfrak{G}'(a, b)) = W_T - L(a, b)$.

Definition 6.4:

$$L_s = \{w_1 E w_2 E w_2^* E w_1^* \mid w_1 \in W_{\{0,1\}},\ w_2 \in W_{\{0,1\}}\}.$$

Lemma 6.3: L_s is an SPL, represented by the SPG:

$$\mathfrak{G}_s = (V_s, P_s, T, S),$$

where

$$V_s = \{S, U, 0, 1, E\}, \quad T = \{0, 1, E\},$$

$$P_s = \left\{ \begin{array}{l} S \to 0S0,\ S \to 1S1,\ S \to EUE \\ U \to 0U0,\ U \to 1U1,\ U \to E \end{array} \right\}.$$

Proof: Obvious.

Lemma 6.4: $W_T - L_s$ is an SPL, represented by the SPG:

$$\mathfrak{G}' = (V_s', P_s', T, S),$$

where

$$V_s' = \{S, S_1, S_2, Q, R, 0, 1, E\}, \quad T = \{0, 1, E\},$$
$$P_s' = P_1 \cup P_2 \cup P_3 \cup P_4 \cup P_{10} \cup P_{11}.$$

Here, P_1, P_2, P_3 and P_4 are those defined in Lemma 6.2 and

$$P_{10} = \left\{ \begin{array}{l} S \to Q,\ Q \to 0Q0,\ Q \to 1Q1, \\ Q \to 0S_1 ES_1ES_1E,\ Q \to 1S_1ES_1ES_1E, \\ Q \to ES_1ES_1ES_10,\ Q \to ES_1 ES_1ES_11, \\ Q \to 0S_1ES_1ES_1ES_11,\ Q \to 1S_1ES_1ES_1ES_10 \end{array} \right\},$$

$$P_{11} = \left\{ \begin{array}{l} S \to S_1ERES_1,\ R \to 0R0,\ R \to 1R1, \\ R \to 0S_1E,\ R \to 1S_1E,\ R \to ES_10, R \to ES_11, \\ R \to 0S_1ES_11,\ R \to 1S_1ES_10 \end{array} \right\}.$$

Proof: By $P_1 \cup P_2 \cup P_3 \cup P_4$ we get all strings over T containing more than, or less than 3 E's (see Lemma 4.2). By $P_1 \cup P_{10}$ we get all strings of the form $w_1 Ew_2 Ew_3 Ew_4$, where $w_i \in W_{\{0,1\}}$ $(i = 1, 2, 3, 4)$ and $w_4 \neq w_1^*$. By $P_1 \cup P_{11}$ we get all strings of that form, where $w_3 \neq w_2^*$.

Remark: By Defs. 6.3 and 6.4 we have:

$$L(a, b) \cap L_s = \left\{ 10^{i_k} \cdots 10^{i_1} Ea_{i_1} \cdots a_{i_k} Eb_{i_1}^* \cdots b_{i_k}^* EO^{i_1} 1 \cdots 0^{i_k} 1 \right|$$
$$k \geq 1;\ 1 \leq i_\nu \leq n \text{ for } \nu = 1, \ldots, k;\ a_{i_1} \cdots a_{i_k} = b_{i_1} \cdots b_{i_k}$$

(since $b_{i_k}^* \cdots b_{i_1}^* = (b_{i_1} \cdots b_{i_k})^*$). Hence, by Def. 6.1, $\mathcal{P}(a, b) = 0$ implies $L(a, b) \cap L_s = \emptyset$. If $\mathcal{P}(a, b) = 1$ then $L(a, b) \cap L_s$ is infinite, since

$$a_{i_1} \cdots a_{i_k} = b_{i_1} \cdots b_{i_k}$$

implies $(a_{i_1} \cdots a_{i_k})^m = (b_{i_1} \cdots b_{i_k})^m$ for $m = 1, 2, 3, \ldots$. Moreover, we shall show that when $\mathcal{P}(a, b) = 1$, $L(a, b) \cap L_s$ is not an SPL. But in order to prove our theorems also for SPGs with *two* terminal symbols only, we shall first appropriately recode the vocabulary of $L(a, b)$ and L_s.

Definition 6.5: Let φ: $W_{\{0,1,E\}} \to W_{\{0,1\}}$ be the following mapping:

$$\varphi(0) = 101,\ \varphi(1) = 1001,\ \varphi(E) = 10001,$$

and for any string $X_1 \cdots X_m$ over $\{0, 1, E\}$:

$$\varphi(X_1 \cdots X_m) = \varphi(X_1) \cdots \varphi(X_m) \text{ (and } \varphi(\Lambda) = \Lambda).$$

Notations: $\overline{0} = \varphi(0)$, $\overline{1} = \varphi(1)$, $\overline{E} = \varphi(E)$, and for any string x over $\{0, 1, E\}$: $\overline{x} = \varphi(x)$. $\overline{L_s} = \varphi(L_s)$, $\overline{L}(a, b) = \varphi(L(a, b))$, $\overline{W} = \varphi(W_{\{0,1,E\}})$, $L' = W_{\{0,1\}} - \overline{W}$.

It can easily be verified that the mapping φ is one-to-one. Therefore we have, for any languages L, L_1, L_2 over $\{0, 1, E\}$: $\varphi(L_1 \cap L_2) = \varphi(L_1) \cap \varphi(L_2)$; $\varphi(W_{\{0,1,E\}} - L) = \varphi(W_{\{0,1,E\}}) - \varphi(L) = \overline{W} - \varphi(L) = W_{\{0,1\}} - (L' - \varphi(L)) = W_{\{0,1\}} - (L' \cup \varphi(L))$.

Lemma 6.5: L' is an SPL, represented by the SPG:

$$\mathfrak{G}' = (V_1, P', T', S),$$

where

$$V_1 = \{S, S_1, 0, 1\}, \quad T' = \{0, 1\},$$

$$P' = \left\{ \begin{array}{l} S \to 0S_1,\ S \to S_1 0,\ S \to 11\,S_1,\ S \to S_1 11, \\ S \to S_1 010\,S_1,\ S \to S_1 111 S_1, \\ S \to S_1 0000\,S_1, \\ S_1 \to 0\,S_1,\ S_1 \to 1\,S_1,\ S_1 \to \Lambda \end{array} \right\}.$$

Proof: L' consists of all strings over $\{0, 1\}$ which start or terminate with 0 or with 11, or which contain a substring of the form 010 or 111 or 0000. These are exactly the strings over $\{0, 1\}$ generated by $\mathfrak{G}'$.

Lemma 6.6: Let (a, b) be the same as in Def. 6.1, and let $\mathcal{P}(a, b) = 1$. Then $\overline{L}(a, b) \cap \overline{L}_s$ is not an SPL.

Proof: The mapping φ is one-to-one. Therefore

$$\overline{L}(a, b) \cap \overline{L}_s = \varphi(L(a, b)) \cap \varphi(L_s) = \varphi(L(a, b) \cap L_s).$$

Since $\mathcal{P}(a, b) = 1$, $L(a, b) \cap L_s$ is infinite (see Remark preceding Def. 6.5), and therefore $\overline{L}(a, b) \cap \overline{L}_s$ is infinite too, and the lengths of its sentences are not bounded.

Every sentence $\bar{z} \in \overline{L}(a, b) \cap \overline{L}_s$ admits a unique decomposition of the form $\bar{z} = c\bar{E}d\bar{E}\bar{d}^*\bar{E}\bar{c}^*$, where

$$\bar{c} = \overline{10}^{i_k} \cdots \overline{10}^{i_1} \ (k \geq 1, \quad 1 \leq i_\nu \leq n \text{ for } \nu = 1, \ldots, k),$$

and $\bar{d} = \bar{a}_{i_1} \cdots \bar{a}_{i_k} = \bar{b}_{i_1} \cdots \bar{b}_{i_k}$. The strings $\bar{c}$ and $\bar{d}$ do not contain $\bar{E}$ as a substring ($\bar{E} = 10001$). Clearly, the substring $\bar{c}$ uniquely determines the whole string $\bar{z}$.

If the sentence $\bar{z}$ is long, then the number k must be large, since the lengths of the strings $\bar{a}_i$, $\bar{b}_i$, $\overline{10}^i$ $(1 \leq i \leq n)$ are bounded, and then $\bar{d}$ must be long too, because the strings $\bar{a}_i$, $\bar{b}_i$ $(1 \leq i \leq n)$ are non-empty.

Assume that $\overline{L}(a, b) \cap \overline{L}_s$ is an SPL. Then we have, by Theorem 4.1, two numbers p and q, such that every sentence $\bar{z} \in \overline{L}(a, b) \cap \overline{L}_s$ with $l(\bar{z}) \geq p$ admits a decomposition $\bar{z} = xuwvy$ with $l(u) + l(v) \geq 1$, $l(uwv) \leq q$, such that $xu^m wv^m y \in \overline{L}(a, b) \cap \overline{L}_s$ for $m = 1, 2, 3, \ldots$.

Take a sentence $\bar{z} = \bar{c}\bar{E}\bar{d}\bar{E}\bar{d}^*\bar{E}\bar{c}^* \in \overline{L}(a, b) \cap \overline{L}_s$, such that

$$l(\bar{z}) \geq p, \ l(\bar{d}) \geq q.$$

(This is possible by the remark on the lengths of $\bar{z}$ and $\bar{d}$.) Compare the two decompositions:

$$\bar{z} = \bar{c}\bar{E}\bar{d}\bar{E}\bar{d}^*\bar{E}\bar{c}^* = xuwvy. \tag{2}$$

$$l(\bar{E}) = l(10001) = 5, \quad l(\bar{c}^*) = l(\bar{c}), \quad l(\bar{d}^*) = l(\bar{d}) \geq q, \quad l(uwv) \leq q.$$

Therefore:

$$l(\bar{z}) = 2 \cdot l(\bar{c}) + 2 \cdot l(\bar{d}) + 15 = l(x) + l(uwv) + l(y),$$

$$l(x) + l(y) = 2 \cdot l(\bar{c}) + 2 \cdot l(\bar{d}) + 15 - l(uwv) \geqq 2 \cdot l(\bar{c}) + 15.$$

Hence

$$l(x) \geqq l(\bar{c}) + 5 \text{ or } l(y) \geqq l(\bar{c}) + 5.$$

Let

$$xuuwvvy = \bar{z}_2 = \bar{c}_2 \overline{E} \bar{d}_2 \overline{E} \bar{d}_2^* \overline{E} \bar{c}_2^* \in \overline{L}(a, b) \cap L_s.$$

If $l(x) \geqq l(\bar{c}) + 5$, then $\bar{c}\overline{E}$ is an initial segment of x (see (2)), and therefore $\bar{c}_2 = \bar{c}$ ($\bar{c}_2$ is the substring of $\bar{z}_2$ preceding the first occurrence of a substring $\overline{E}$ in $\bar{z}_2$), which implies $\bar{z}_2 = \bar{z}$, since the first quarter $\bar{c}$ uniquely determines the whole string $\bar{z}$. Similarly, if $l(y) \geqq l(\bar{c}) + 5$, then $\overline{E}\bar{c}^*$ is a confinal segment of y (see (2)), and therefore $\bar{c}_2^* = \bar{c}^*$ ($\bar{c}_2^*$ is the substring of $\bar{z}_2$ following the last occurrence of a substring $\overline{E}$ in $\bar{z}_2$), which again implies $\bar{c}_2 = \bar{c}$ and $\bar{z}_2 = \bar{z}$.

But $\bar{z}_2$ cannot equal $\bar{z}$, since

$$l(\bar{z}_2) = l(\bar{z}) + l(u) + l(v) > l(\bar{z}) \quad (l(u) + l(v) \geqq 1).$$

Thus the assumption that $\overline{L}(a, b) \cap \overline{L}_s$ is an SPL leads to a contradiction. Hence, if $\mathcal{P}(a, b) = 1$, then $\overline{L}(a, b) \cap \overline{L}_s$ is not an SPL.

Remarks: a) The proof of Lemma 6.6 shows that in case $\mathcal{P}(a, b) = 1$, $\overline{L}(a, b) \cap \overline{L}_s$ is not only different from any SPL, but does not even *contain* an infinite SPL.

b) If we were contented to prove Theorems 6.1, 6.2 and 6.3 only for terminal vocabularies T containing more than two symbols, then the "translation" φ would be superfluous and it would suffice to prove that $L(a, b) \cap L_s$ is not an SPL (for $\mathcal{P}(a, b) = 1$), which is somewhat easier than Lemma 6.6.

Proof of Theorem 6.1: Given a pair (a, b) of n-tuples of non-empty strings over $\{0, 1\}$, construct $\mathfrak{G}(a, b)$ as in Lemma 6.1, and $\mathfrak{G}_s$ as in Lemma 6.3. $L(\mathfrak{G}(a, b)) = L(a, b)$, $L(\mathfrak{G}_s) = L_s$. By Corollary 3.1 to Theorem 3.3 construct an SPG $\overline{\mathfrak{G}}(a, b)$, such that $L(\overline{\mathfrak{G}}(a, b)) = \varphi(L(a, b)) = \overline{L}(a, b)$, and an SPG $\overline{\mathfrak{G}}_s$, such that $L(\overline{\mathfrak{G}}_s) = \varphi(L_s) = \overline{L}_s$.

If $\mathcal{P}(a, b) = 1$, then, by Lemma 6.5, $\overline{L}(a, b) \cap \overline{L}_s = L(\overline{\mathfrak{G}}(a, b)) \cap L(\overline{\mathfrak{G}}_s)$ is not an SPL, and therefore, by Theorem 2.1, is not an FAL, and is certainly neither empty nor finite. But if $\mathcal{P}(a, b) = 0$, then, by the Remark preceding Def. 6.5, $\overline{L}(a, b) \cap \overline{L}_s = L(\overline{\mathfrak{G}}(a, b)) \cap L(\overline{\mathfrak{G}}_s)$ is empty, and therefore certainly finite, and is representable by a FA and by an SPG.

Thus, if any one of the four problems a, b, c, d of Theorem 6.1 were effectively solvable, then we would have a decision procedure for Post's correspondence problem, which is impossible, by [P.].

Proof of Theorem 6.2: Given a pair (a, b) of n-tuples of non-empty strings over $\{0, 1\}$, construct $\mathfrak{G}'(a, b)$, as in Lemma 6.2, and $\mathfrak{G}'_s$ as in Lemma 6.4. $L(\mathfrak{G}'(a, b)) = W_{\{0,1,E\}} - L(a, b)$, $L(\mathfrak{G}'_s) = W_{\{0,1,E\}} - L_s$. By Corollary 3.1 to Theorem 3.3 construct an SPG $\overline{\mathfrak{G}}'(a, b)$, such that

$$L(\overline{\mathfrak{G}}'(a, b)) = \varphi(W_{\{0,1,E\}} - L(a, b)) = \overline{W} - \overline{L}(a, b)$$

($\overline{W} = \varphi(W_{\{0,1,E\}})$ and φ is one-to-one), and an SPG $\overline{\mathfrak{G}}'_s$, such that

$$L(\overline{\mathfrak{G}}'_s) = \varphi(W_{\{0,1,E\}} - L_s) = \overline{W} - \overline{L}_s.$$

Since

$$\bar{L}(a, b) \cap \bar{L}_s \subseteq \bar{W} \subseteq W_{\{0,1\}}$$

we have:

$$W_{\{0,1\}} - \bar{L}(a, b) \cap \bar{L}_s = (W_{\{0,1\}} - \bar{W}) \cup (\bar{W} - \bar{L}(a, b)) \cup (\bar{W} - \bar{L}_s).$$

By Lemma 6.5, construct an SPG $\mathfrak{G}'$, such that $L(\mathfrak{G}') = L' = W_{\{0,1\}} - \bar{W}$. Finally construct from $\mathfrak{G}'$, $\bar{\mathfrak{G}}'(a, b)$ and $\bar{\mathfrak{G}}'_s$ an SPG $\bar{\mathfrak{G}}''(a, b)$, such that

$$L(\bar{\mathfrak{G}}''(a, b)) = L(\mathfrak{G}') \cup L(\bar{\mathfrak{G}}'(a, b)) \cup L(\bar{\mathfrak{G}}'_s) =$$

$$= (W_{\{0,} - \bar{W}) \cup (\bar{W} - \bar{L}(a, b)) \cup (\bar{W} - \bar{L}_s) = W_{\{0,1\}} - \bar{L}(a, b) \cap \bar{L}_s$$

(by a double application of Theorem 3.1. e).

Now,

$$W_T - L(\bar{\mathfrak{G}}''(a, b)) = W_{\{0,1\}} - (W_{\{0,1\}} - \bar{L}(a, b) \cap \bar{L}_s) = \bar{L}(a, b) \cap \bar{L}_s.$$

Therefore, as in Theorem 6.1, the answers to the four questions a, b, c, d of Theorem 6.2 are positive for $\mathfrak{G} = \bar{\mathfrak{G}}''(a, b)$ if $\mathcal{P}(a, b) = 0$, and are negative if $\mathcal{P}(a, b) = 1$. Thus a decision procedure for any one of the four problems a, b, c, d of Theorem 6.2 would yield a decision procedure for Post's correspondence problem. Hence, no such decision procedure exists.

Proof of Theorem 6. 3: a. 1) Let $\mathfrak{G}_W = (V_W, P_W, T, S)$, where

$$V_W = \{0, 1, S\}, \quad T = \{0, 1\}, \quad P_W = \{S \to S0, S \to S1, S \to \Lambda\}.$$

$L(\mathfrak{G}_W) = W_{\{0,1\}}$. Now, for any SPG $\mathfrak{G}_2$ with a terminal vocabulary $T = \{0,1\}$, $W_{\{0,1\}} = L(\mathfrak{G}_W) \subseteq L(\mathfrak{G}_2)$ iff $L(\mathfrak{G}_2) = W_{\{0,1\}}$, i. e., iff $W_T - L(\mathfrak{G}_2) = \emptyset$, and the problem whether $W_T - L(\mathfrak{G}_2)$ is empty is effectively unsolvable for $\mathfrak{G}_2$, by Theorem 6.2. a.

a. 2) Let $\bar{\mathfrak{G}}'_s$ be the SPG constructed in the proof of Theorem 6.2, such that $L(\bar{\mathfrak{G}}'_s) = \bar{W} - \bar{L}_s$. Let $\bar{\mathfrak{G}}$ (a, b) be the SPG constructed in the proof of Theorem 6.1, such that

$$\bar{L}(\bar{\mathfrak{G}}(a, b)) = \bar{L}(a, b). \quad \bar{L}(a, b) = \varphi(L(a, b)) \subseteq \bar{W} = \varphi(W_{\{0,1,E\}}).$$

If $\mathcal{P}(a, b) = 0$, then $\bar{L}(a, b) \cap \bar{L}_s = \emptyset$, so that

$$L(\bar{\mathfrak{G}}(a, b)) = \bar{L}(a, b) \subseteq \bar{W} - \bar{L}_s = L(\bar{\mathfrak{G}}'_s).$$

If $\mathcal{P}(a, b) = 1$, then $\bar{L}(a, b) \cap \bar{L}_s$ is non-empty, so that

$$L(\bar{\mathfrak{G}}(a, b)) = \bar{L}(a, b) \not\subseteq \bar{W} - \bar{L}_s = L(\bar{\mathfrak{G}}'_s).$$

The effective unsolvability of Post's correspondence problem thus implies the effective unsolvability of the problem whether, for any given pair (a, b) of n-tuples of non-empty strings over $\{0, 1\}$, $L(\bar{\mathfrak{G}}(a, b)) \subseteq L(\bar{\mathfrak{G}}'_s)$, and a fortiori the effective unsolvability of the problem whether, for any given SPG $\mathfrak{G}_1$ with a terminal vocabulary $T = \{0, 1\}$, $L(\mathfrak{G}_1) \subseteq L(\bar{\mathfrak{G}}'_s)$.

b) Let $\mathfrak{G}_W$ be the SPG constructed in a. 1), such that $L(\mathfrak{G}_W) = W_{\{0,1\}}$. Now, for any SPG $\mathfrak{G}_1$ with a terminal vocabulary $T = \{0, 1\}$, $L(\mathfrak{G}_1) = L(\mathfrak{G}_W) = W_{\{0,1\}}$ iff $W_T - L(\mathfrak{G}_1) = \emptyset$, and the problem whether $W_T - L(\mathfrak{G}_1)$ is empty is effectively unsolvable for $\mathfrak{G}_1$, by Theorem 6.2 a.

c) The class of FALs over a vocabulary T is closed under complementation with respect to W_T. Therefore $L(\mathfrak{G})$ is an FAL iff $W_T - L(\mathfrak{G})$ is an FAL,

where T is the terminal vocabulary of the SPG $\mathfrak{G}$. The problem whether $W_T - L(\mathfrak{G})$ is an FAL is effectively unsolvable, by Theorem 6.2 c.

d) Let $\mathfrak{A}_W = (T, \Sigma, M, \sigma_0, \Phi)$, where

$T = \{0, 1\}$, $\Sigma = \Phi = \{\sigma_0\}$, $M(\sigma_0, 0) = M(\sigma_0, 1) = \sigma_0$. $L(\mathfrak{A}_W) = W_{\{0,1\}}$.

For any SPG $\mathfrak{G}$ with a terminal vocabulary $T = \{0, 1\}$, $L(\mathfrak{G}) = L(\mathfrak{A}_W) = W_{\{0,1\}}$ iff $W_T - L(\mathfrak{G}) = \emptyset$, and the problem whether $W_T - L(\mathfrak{G})$ is empty is unsolvable for $\mathfrak{G}$, by Theorem 6.2 a.

Remarks: a) Theorems 6.1, 6.2 and 6.3 may be proved also for SPGs with terminal vocabularies T containing n symbols, for any $n \geq 3$. This may be done, for example, by letting the terminal vocabulary T of the SPGs $\mathfrak{G}(a, b)$, $\mathfrak{G}'(a, b)$, $\mathfrak{G}_s$ and $\mathfrak{G}'_s$ (Lemmata 6.1, 6.2, 6.3 and 6.4, respectively) be $T = \{E, 0, 1, 2, \ldots, n-2\}$, and changing $\mathfrak{G}'(a, b)$ and $\mathfrak{G}'_s$ such that they represent $W_T - L(a, b)$ and $W_T - L_s$ respectively, with the new vocabulary T. Then the "translation" φ (Def. 6.5) becomes superfluous, and the proof proceeds as in Lemma 6.6, except that $L(a, b) \cap L_s$ (instead of $\bar{L}(a, b) \cap \bar{L}_s$) is shown not to be an SPL, if $\mathcal{P}(a, b) = 1$.

b) There is a simpler proof of Theorem 6.1 a, b, c, and Theorem 6.2 a, b, c. Given a pair (a, b) of n-tuples of non-empty strings over $\{0, 1\}$, construct SPGs $\mathfrak{G}(a)$ and $\mathfrak{G}(b)$ representing the languages $L(a)$ and $L(b)$ (see Def. 6.2). If $\mathcal{P}(a, b) = 0$, then $L(a) \cap L(b) = \emptyset$. If $\mathcal{P}(a, b) = 1$, then $L(a) \cap L(b)$ is infinite and is not representable by a FA, as is easily proved by [R. S., Lemma 8].

This simplified proof of Theorem 6.1 a, b, c via $L(a)$ and $L(b)$ is almost identical with the proof of the effective unsolvability of the empty-intersection problem for TTAa, given in [R. S., Theorem 18]. This proof, however, would not have yielded the effective unsolvability for $\mathfrak{G}_2$ of problems a, b, c of Theorem 6.1 with $\mathfrak{G}_1$ fixed, which we obtained here by using $L(a, b)$ and L_s. Theorem 6.1 d cannot be proved in this way, since $L(a) \cap L(b)$ may be an SPL even when $\mathcal{P}(a, b) = 1$.

In order to prove Theorem 6.2 a, b, c, construct SPGs $\mathfrak{G}'(a)$ and $\mathfrak{G}'(b)$ representing $W_T - L(a)$ and $W_T - L(b)$. ($T = \{0, 1, E\}$), and an SPG $\mathfrak{G}'(a)(b)$ representing $(W_T - L(a)) \cup (W_T - L(b)) = W_T - L(a) \cap L(b)$. Now,

$$W_T - L(\mathfrak{G}'(a)(b)) = W_T - (W_T - L(a) \cap L(b)) = L(a) \cap L(b),$$

which language is empty if $\mathcal{P}(a, b) = 0$, and not representable by a FA if $\mathcal{P}(a, b) = 1$.

In order to get the results for SPGs with a terminal vocabulary containing two symbols only, use the mapping $\varphi: W_{\{0,1,E\}} \to W_{\{0,1\}}$ given in Def. 6.5. Note that

$$\begin{aligned} W_{\{0,1\}} - \varphi(L(a) \cap L(b)) &= (W_{\{0,1\}} - \bar{W}) \cup (\bar{W} - \varphi(L(a) \cap L(b))) \\ &= L' \cup \varphi(W_{\{0,1,E\}} - L(a) \cap L(b)). \end{aligned}$$

L' is represented by the SPG $\mathfrak{G}'$ constructed in Lemma 6.5, and an SPG representing $W_{\{0,1\}} - \varphi(L(a) \cap L(b))$ can be effectively constructed from $\mathfrak{G}'$ and $\mathfrak{G}'(a)(b)$.

Similary, we could also have proved Theorem 6.3 by the method mentioned here, except for that part of Theorem 6.3 a in which $\mathfrak{G}_2$ is assumed to be fixed, which part is based on Theorem 6.1 a with fixed $\mathfrak{G}_2$.

Section 7: Some relations between SPGs and finite automata

Lemma 7.1: Let $\mathfrak{G} = (V, P, T, S)$ be a 1-SPG. If all the productions of P have the form $A \to tB$ or $A \to t$, where $\Lambda \neq t \in W_T$, then $L(\mathfrak{G})$ is an FAL. The same result holds if all the productions of P have the dual form $A \to Bt$ or $A \to t$.

Proof: The case $l(t) = 1$ in every production was already dealt with (Remark following Th. 2.1). The general case is easily reduced to it: replace each production $A \to T_1 \cdots T_k B$ by

$$A \to T_1 B_1,\ B_1 \to T_2 B_2,\ \ldots,\ B_{k-1} \to T_k B,$$

where all B_j are distinct and new.

Definition 7.1: A symbol A of an SPG is called *self-embedding* (s. e.) if the condition:

$$(\exists u)(\exists v)(u \neq \Lambda \,\&\, v \neq \Lambda \,\&\, A \overset{*}{\Rightarrow} uAv)$$

holds for A. An SPG is called *self-embedding* (s. e.) if it contains a s. e. symbol. (Compare Def. 5.2.)

By Th. 5.1 d, the property of being s. e., for a symbol as well as for an SPG, is effectively decidable.

Theorem 7.1: A non-s. e. SPG is equivalent to a FA.

Remarks: a) The theorem is due to Chomsky [C. 1], [C. 2]. It provides an effective sufficient condition for an SPG to be equivalent to a FA. However, we know that an effective *criterion* (necessary and sufficient condition) does not exist (Th. 6.3 c).

b) For an actual examination of SPGs, the theorem can be stated in a more useful form: a 1-SPG, whose reduced form is non-s. e., is equivalent to a FA. Indeed, an SPG may contain superfluous s. e. symbols which are eliminated in its reduced form.

c) The proof given below is Chomsky's second proof [c. 2] which, except for slight modifications, was also obtained independently by us.

Proof: By Lemmata 4.1, 4.2 and 5.1, we pass to a reduced 2-SPG $\mathfrak{G} = (V, P, T, S)$ which is equivalent to the original SPG except for strings of length ≤ 1. $\mathfrak{G}$ is also non-s. e., and it suffices to prove that $L(\mathfrak{G})$ is an FAL.

Let $A_1 = S, A_2, \ldots, A_k$ be the auxiliary symbols of $\mathfrak{G}$, i. e. the symbols of V-T. $\mathfrak{G}$ being reduced, we have

$$A_1 \overset{*}{\Rightarrow} u_i A_i v_i \text{ with } l(u_i v_i) \geq 1 \quad (i = 2, 3, \ldots, k). \tag{1}$$

We first prove the theorem in a special case:

Case I: Suppose that we have:

$$A_i \Rightarrow w_i A_1 z_i, \text{ with } l(w_i z_i) \geq 1 \quad (i = 1, 2, \ldots, k). \tag{2}$$

In this case, all the A_i are embedding symbols. Since A_1 is non-s. e., the embeddings of A_1 are either all of them left: $A_1 \overset{*}{\Rightarrow} A_1 a$ $(a \neq \Lambda)$, or all of them right: $A_1 \overset{*}{\Rightarrow} a A_1$. We claim that if A_1 admits left embeddings, then all the

productions of P have the form $A_i \to tA_j$ or $A_i \to t$, $t \in W_T$. Indeed suppose that $A_i \to xA_jy$, $y \neq \Lambda$, belongs to P. Then by (1) and (2) we obtain:

$$A_i \to xA_jy \overset{*}{\Rightarrow} xw_jA_1z_jy \overset{*}{\Rightarrow} xw_jaA_1z_jy \overset{*}{\Rightarrow} xw_jau_iA_iv_iz_jy.$$

Since $a \neq \Lambda$, $y \neq \Lambda$, this contradicts the assumption that A_i is non-s. e. .

In the same way, if A_1 admits right embeddings only, then the productions are all of the form $A_i \to A_jt$ or $A_i \to t$. In both cases we obtain by Lemma 1 that $L(\mathfrak{G})$ is an FAL.

In the general case, we shall proceed by induction on k. If $k = 1$, then S is the only auxiliary symbol and all the productions have the form $S \to tS$, $S \to t$ or the dual form $S \to St$, $S \to t$, $t \in W_T$. Again by Lemma 1, $L(\mathfrak{G})$ is an FAL.

We assume now that the theorem holds true for SPGs with less than k auxiliary symbols. We may also assume that we do not have Case I, i. e., that the negation of (2) holds:

(3) $$\text{For some } j, \quad 1 \leqq j \leqq k, \sim (\exists w)(\exists z)(A_j \overset{*}{\Rightarrow} wA_1z).$$

Suppose first that $j > 1$ in (3). Let $\mathfrak{G}'$ be the SPG which differs from $\mathfrak{G}$ only in that A_j and each production $A_j \to a$ are deleted and that in the other productions A_j is everywhere replaced by a new terminal symbol D. This $\mathfrak{G}'$ is a non-s. e. SPG which contains k-1 auxiliaries, – hence by the induction hypothesis $L(\mathfrak{G}')$ is an FAL. The same is true for the language

$$K = \{x \mid x \in W_T \,\&\, A_j \overset{*}{\Rightarrow} x \text{ in } L(\mathfrak{G})\},$$

since K is determined by the SPG $(V - \{S\}, P_1, T, A_j)$ where P_1 is obtained from P by deleting all the productions containing S. (Here $S = A_1$ is superfluous because of (3).) Now a string z belongs to $L(\mathfrak{G})$ just in case it is obtained from some string x of $L(\mathfrak{G}')$ by a substitution of arbitrary strings of K in all the occurrences of D in x. Hence the fact that $L(\mathfrak{G})$ is an FAL follows from the Remark following Th. 3.3.

Finally we take the case $j = 1$ in (3). Let $a_1, \ldots, a_r$ be all the strings such that $A_1 \to a_i$. Let $H_i = \{x \in W_T \mid a_i \overset{*}{\Rightarrow} x\}$. Clearly $L(\mathfrak{G}) = \bigcup_{i=1}^{r} H_i$. Let $a_1 = B_1 \cdots B_n$ and let $K_m = \{x \in W_T \mid B_m \overset{*}{\Rightarrow} x\}$, then H_1 equals the product $K_1 \cdot K_2 \cdot \ldots \cdot K_n$. The other H_i are given by similar products. K_m is determined by the SPG $(V - \{S\}, P_1, T, B_m)$ which contains $k - 1$ auxiliaries, hence each K_m is an FAL. Thus $L(\mathfrak{G})$ is obtained from FALs by products and unions and is therefore an FAL itself. (See Section 2, page 145).

This concludes the proof of the theorem.

Theorem 7.2: Let L be an SPL over the vocabulary $T \cup \{H\}$ $(H \notin T)$, fulfilling the conditions:

(i) Every sentence of L has the form xHy, where $x, y \in W_T$ and $l(xy) \geqq 1$;

(ii) For a given x, the number of y such that $xHy \in L$ is finite, and for a given y the number of x such that $xHy \in L$ is finite.

Then the sets $M = \{x \mid (\exists y)(xHy \in L)\}$ and $N = \{y \mid (\exists x)(xHy \in L)\}$ are FALs.

Proof: Suppose L is represented by the reduced 2-SPG $\mathfrak{G} = (V, P, T \cup \{H\}, S)$. Let

$$R = \{A \in V \mid (\exists u)(\exists v)(A \overset{*}{\Rightarrow} uHv)\},$$

and let $Q = V - R$. The set R contains H and every symbol from which an "H-string" can be generated. Clearly $S \in R$.

Consider any "maximal" generation chain (a chain which cannot be continued) starting from S: $S \overset{*}{\Rightarrow} a_1 \overset{*}{\Rightarrow} a_2 \overset{*}{\Rightarrow} \cdots \overset{*}{\Rightarrow} a_k$. Since $\mathfrak{G}$ is reduced, a_k is a sentence of L. Therefore, each string a_i in such a chain contains exactly one *occurrence* of a symbol of R. For if $a_i = \cdots A \cdots B \ldots$, where $A, B \in R$, ($A = B$ not excluded), we would obtain

$$S \overset{*}{\Rightarrow} a_i \overset{*}{\Rightarrow} \cdots A \cdots B \cdots \overset{*}{\Rightarrow} \cdots H \cdots H \cdots \in L,$$

which is impossible. (It should be constantly remembered that in a reduced SPG any non-terminal string generates a terminal string.)

It follows that in our reduced SPG all the productions of P are of the form:

(1) $A \to uBv$, where $A, B \in R$ and $u, v \in W_Q$,
or $C \to d$, where $C \in Q$ and $d \in W_Q$.

Now, by using condition (ii), we obtain that no symbol of Q is embedding. For if $C \in Q$ were embedding, i.e., if there were d and e such that $C \overset{*}{\Rightarrow} dCe$, with $l(de) \geqq 1$, we would have:

$$S \overset{*}{\Rightarrow} \cdots C \cdots Ht \quad \text{or} \quad S \overset{*}{\Rightarrow} tH \cdots C \ldots, \text{ with } t \in W_T.$$

Since C is embedding, we would obtain an infinite number of strings $x \in W_T$ with $xHt \in L$, or an infinite number of $y \in W_T$ with $tHy \in L$, contradicting condition (ii).

Therefore each string over Q generates only a finite number of terminal strings, and we do not change the language if we replace the productions $A \to uBv$, $C \to d$ (see (1)) by the productions $A \to u_i Bv_j$, $C \to d_k$, where u_i, v_j and d_k run over all the terminal strings generated by u, v and d, respectively.

Consider now the SPG $(V, P_1, T \cup \{H\}, S)$ whose productions are $A \to u_i B$ and $C \to d_k$ (i. e. we replace each of the productions $A \to u_i Bv_j$ by $A \to u_i B$). This SPG fulfils the requirement of Lemma 1, hence it determines an FAL. It is immediately verified that this FAL consists of all the strings xH for which there exists a y with $xHy \in L$. Omitting the symbol H, we obtain that $M = \{x \mid (\exists y)(xHy \in L)\}$ is an FAL. In a similar manner we obtain that N is an FAL.

Example: Let L be a *mirror language*, i. e., all its strings have the form xHx^*. Our theorem yields: If L is an SPL, then $M = \{x \mid xHx^* \in L\}$ is an FAL. The converse is easy and was proved (Th. 2.1.b). We could use this fact, in place of Theorem 4.1, in order to show that various languages are not SPLs. For instance, $\{0^n10^n H 0^n10^n\}$ $(n = 1, 2, 3 \ldots)$ is not an SPL, though it can easily be shown to be an intersection of two SPLs.

Section 8: The intersection of SPGs and FAa

In this last section we shall prove the following

Theorem 8.1: Let $\mathfrak{G}$ be an SPG, $\mathfrak{A}$ a FA. It is possible to construct an SPG $\overline{\mathfrak{G}}$ such that $L(\overline{\mathfrak{G}}) = L(\mathfrak{G}) \cap L(\mathfrak{A})$.

Proof: First note that it suffices to deal with 1-SPGs. Secondly, if $\mathfrak{A}$ is a FA with r final states, then $L(\mathfrak{A})$ is a union $\bigcup_{i=1}^{r} L(\mathfrak{A}_i)$, where each $\mathfrak{A}_i$ has a single final state. Thus

$$L(\mathfrak{G}) \cap L(\mathfrak{A}) = \bigcup_{i=1}^{r} [L(\mathfrak{G}) \cap L(\mathfrak{A}_i)].$$

Hence it suffices to prove the theorem for FAa with a single final state.

Let therefore $\mathfrak{G} = (V, P, T, S)$ be a 1-SPG, and let

$$\mathfrak{A} = (T, \Sigma, M, \sigma_0, \{\sigma_f\})$$

be a FA with σ_f as a single final state.

Construction: We construct the 1-SPG $\overline{\mathfrak{G}} = (\overline{V}, \overline{P}, T, \overline{S})$ where:

1. $\overline{V} = (\Sigma \times V \times \Sigma) \cup T$, (i. e., $\overline{V}$ contains T and all the triplets (σ, A, τ) where $\sigma \in \Sigma$, $A \in V$, $\tau \in \Sigma$),
2. $\overline{S} = (\sigma_0, S, \sigma_f)$,
3. $\overline{P}$ contains the following productions:

(i) if $X \to A_1 \cdots A_k \in P$, then
$(\sigma, X, \tau) \to (\sigma, A_1, \tau_1)(\tau_1, A_2, \tau_2) \cdots (\tau_{k-2}, A_{k-1}, \tau_{k-1})(\tau_{k-1}, A_k, \tau) \in P$, for every $\sigma, \tau, \tau_1, \ldots, \tau_{k-1} \in \Sigma$;

ii) if $X \in T$ and $M(\sigma, X) = \tau$, then $(\sigma, X, \tau) \to X$.

Lemma 8.1: If $x \in L(\mathfrak{G}) \cap L(\mathfrak{A})$, then $x \in L(\overline{\mathfrak{G}})$.

Proof: Let $x = A_1 \cdots A_k$. $x \in L(\mathfrak{A})$ implies the existence of a sequence $\sigma_0, \ldots, \sigma_k = \sigma_f$ such that $\sigma_i = M(\sigma_{i-1}, A_i)$, $i = 1, \ldots, k$.
$x \in L(\mathfrak{G})$ implies $S \overset{*}{\Rightarrow} A_1 \cdots A_k$ in $\mathfrak{G}$. From the production of 3 (i) it follows that

$$\overline{S} = (\sigma_0, S, \sigma_f) \overset{*}{\Rightarrow} (\sigma_0, A_1, \sigma_1)(\sigma_1, A_2, \sigma_2) \cdots (\sigma_{k-1}, A_k, \sigma_f)$$

in $\overline{\mathfrak{G}}$. (The last assertion, for *any* sequence $\sigma_0, \sigma_1, \ldots, \sigma_{k-1}, \sigma_f$ is easily verified.) Since $\sigma_i = M(\sigma_{i-1}, A_i)$, we obtain from 3 (ii) that $(\sigma_{i-1}, A_i, \sigma_i) \to A_i$, for $i = 1, \ldots, k$. Hence we finally obtain $\overline{S} \overset{*}{\Rightarrow} A_1 \cdots A_k$ in $\overline{\mathfrak{G}}$.

Lemma 8.2: If $x \in L(\overline{\mathfrak{G}})$, then $x \in L(\mathfrak{G}) \cap L(\mathfrak{A})$.

Proof: We have $\overline{S} = (\sigma_0, S, \sigma_f) \overset{*}{\Rightarrow} x = A_1 \cdots A_k$ in $\overline{\mathfrak{G}}$. Clearly each $A_i \in T$ is generated from some (σ, A_i, τ) by one of the productions of 3 (ii). Before using these productions, i. e., before the passage to the terminals, we necessarily had

(I) $\quad \overline{S} = (\sigma_0, S, \sigma_f) \overset{*}{\Rightarrow} (\sigma_0, A_1, \sigma_1)(\sigma_1, A_2, \sigma_2) \cdots (\sigma_{k-1}, A_k, \sigma_f)$.

(Every string generated from (σ_0, S, σ_f), which does not contain terminal symbols, is composed of triplets, such that the first triplet begins with σ_0,

the last triplet ends with σ_f, and the last element of each triplet equals the first one of its successor – this is easily proved by induction.)

A consideration of the productions of $\bar{\mathfrak{G}}$ shows that relation (I) holds in $\bar{\mathfrak{G}}$ only if $S \overset{*}{\Rightarrow} A_1 \cdots A_k$ in $\mathfrak{G}$. Thus $A_1 \cdots A_k = x \in L(\mathfrak{G})$. Since the A_i were generated from $(\sigma_{i-1}, A_i, \sigma_i)$ by application of 3 (ii), we have also

$$\sigma_i = M(\sigma_{i-1}, A_i),$$

which proves that $A_1 \cdots A_k \in L(\mathfrak{A})$. Lemmata 8.1 and 8.2 together yield $L(\bar{\mathfrak{G}}) = L(\mathfrak{G}) \cap L(\mathfrak{A})$. This concludes the proof of Th. 8.1.

As a simple application we note the following: If we delete a finite set of strings from an SPL, the resulting language is still an SPL. Indeed, it is obtained as the intersection of the original SPL with the complement of the finite set.

References

1. [B. G. S.] Bar-Hillel, Y., Gaifman, C., and E. Shamir, On categorial and phrase-structure grammars, *Bulletin of the Research Council of Israel*, 9 F (1960) 1–16.
2. [B-H. S.] Bar-Hillel, Y. and E. Shamir, Finite-state languages: Formal representation and adequacy problems, *Bulletin of the Research Council of Israel* 8 F (1960) 155–166.
3. [C. 1.] Chomsky, N., On certain formal properties of grammars, *Inform. and Control* 2 (1959) 137–167.
4. [C. 2.] Chomsky, N., A note on phrase structure grammars, *Inform. and Control* 2 (1959) 393–395.
5. [C. M.] Chomsky, N., and G. A. Miller, Finite state languages, *Inform. and Control* 1 (1958) 91–112.
6. [D.] Davis, M., *Computability and unsolvability*, McGraw-Hill, New-York 1958.
7. [P.] Post, E., A variant of a recursively unsolvable problem, *Bull. Amer. Math. Soc.* 52 (1946) 264–268.
8. [R. S.] Rabin, M. O. and D. Scott, Finite automata and their decision problems, *IBM Journal* 3 (1959) 115–125.