Grammatical constructions in selective substitution grammars

By J. GONCZAROWSKI*, H. C. M. KLEIJN**, G. ROZENBERG**

0. Introduction

Selective substitution grammars were proposed as a unifying framework for "grammatically oriented" formal language theory (see [R2]). Informally speaking, a selective substitution grammar consists of a *base* (this is the underlying grammar providing productions) and of a *selector* (which prescribes the use of productions for the rewriting of strings). If one allows the use of arbitrary productions of the form $b \rightarrow w$, where b is a symbol and w is a word, then one deals with the so-called EOS bases. A selector for such a base is a language over a set of symbols consisting of letters and their barred *(activated)* versions.

An element y of the selector (called a selector word) prescribes the rewriting mode of a word x as follows.

If y results from x by barring some occurrences in x, then y gives concession for x to be rewritten; then the rewriting consists of applying productions from the base to *all and only* those occurrences in x that appear barred in y. Thus, given a word x, a direct rewriting of x consists of two steps: (i) matching a selector word y which gives concession to x and (ii) applying to x productions from the base in the fashion prescribed by y.

This defines the direct derivation relation in a selective substitution grammar and (through its transitive and reflexive closure) the derivation relation. A somewhat informal but useful way of thinking about selective substitution grammars is to think of productions as instructions and of the selector as the program in the word processing system that a given selective substitution grammar defines.

Typical research projects concerning the theory of selective substitution grammars are the following.

(1) Specifying (language theoretical) properties of selectors which guarantee that selective substitution grammars using them represent rewriting of a context-free nature. The main theme here is to detect those properties of selectors that allow the transmitting of context during the rewriting process (see, e.g., [KR]).

(2) Discussing the "standard" issue of the difference between sequential and parallel rewriting in a uniform framework. This research sheds some additional light on the difference between those two classical modes of rewriting as well as it

leads to the investigation of new (and natural) classes of rewriting systems (see [EMR] and [KR2]).

(3) The influence of the choice of either various classes of allowable productions (under a fixed class of selectors) or various classes of selectors (under a fixed class of allowable productions) on the language-generating power of the resulting classes of selective substitution grammars (see, e.g., [KR] and [RW]).

In this paper we consider the influence of the properties of selectors on the possibilities of performing several standard grammatical transformations.

A transformation of a grammar to another one (preserving the generated language) is a step done very often in grammatically oriented language theory. Such transformations should lead to grammars which are in a convenient form either from the "user point of view" (e.g., for parsing) or from the analytical point of view (e.g., for proving properties of the generated languages). Once we allow the use of all context-free productions in selective substitution grammars, the fact whether or not a given grammatical transformation can be performed within a given class of selective substitution grammars must depend on (the form of) the selectors available. This dependence is the topic of this paper. In particular we investigate a number of standard grammatical constructions, such as removing λ-productions, removing right recursion, removing chain productions, restricting the right-hand sides of productions to the length 2 and synchronization.

We assume the reader to be familiar with the basic formal language theory (see, e.g., [S]); as far as the theory of selective substitution grammars is concerned, the paper is self-contained.

1. Basic concepts and definitions

We assume the reader to be familiar with formal language theory as, e.g., in the scope of [S] and [RS]. Some notations need, perhaps, an additional explanation. For a word w, $|w|$ denotes its length. λ denotes the empty word. For a finite set X, $\#X$ denotes the cardinality of X. We shall usually identify a singleton set with its element. Alphabets are finite nonempty sets of symbols. For a word w, $\mathrm{alph}(w)$ denotes the set of symbols in w. For a language L, $\mathrm{alph}(L) = \bigcup_{w \in L} \mathrm{alph}(w)$.

Let L_1 and L_2 be languages. Then L_1 and L_2 are considered *equal* if $L_1 \cup \{\lambda\} = L_2 \cup \{\lambda\}$.

Let G be a rewriting system. Then $\mathbf{L}(G)$ denotes the language of G. Two rewriting systems are *equivalent* if the languages they generate are equal.

Let Σ and Φ be alphabets. We denote the family of total homomorphisms from Σ^* into Φ^* by $HOM(\Sigma, \Phi)$ and the family of total finite substitutions from Σ^* into (subsets of) Φ^* by $FSUB(\Sigma, \Phi)$. A homomorphism $h \in HOM(\Sigma, \Phi)$ is a *coding* if $h(a) \in \Phi$ for all $a \in \Sigma$. A homomorphism $h \in HOM(\Sigma, \Phi)$ is a *weak identity* if $h(a) \in \{a, \lambda\}$ for all $a \in \Sigma$. A finite substitution $\varphi \in FSUB(\Sigma, \Phi)$ is a *letter-to-letters substitution* if $\varphi(a) \subset \Phi$ for all $a \in \Sigma$.

A *letter monoid* (*monoid*, for short) is a language of the form Θ^* where Θ is an alphabet. A *word monoid* is a language of the form L^* where L is a finite set of words.

In context-free grammars only non-terminal symbols can be rewritten. Very often it is convenient to permit the rewriting of terminal symbols as well. Thus we arrive at EOS systems (see, e.g., [KR]).

Definition 1.1. An EOS *system* G is a quadruple $\langle \Sigma, h, S, \Delta \rangle$, where Σ is the *alphabet* of G, h is a total finite substitution from Σ^* into (nonempty subsets of) Σ^* called the *substitution* of G, $S \in \Sigma - \Delta$ is the *start symbol* of G and $\Delta \subset \Sigma$ is the set of *terminal symbols* of G. □

As customary, if $a \in \Sigma$ and $w \in h(a)$ then (a, w) is called a *production* in G.

Prod (G) denotes the set of all productions in G and Maxr $(G) = \max \{|w|: (a, w) \in \text{Prod } (G)\}$.

The direct derivation relation in $G(\underset{G}{\Rightarrow})$ and the derivation relation in $G(\overset{*}{\underset{G}{\Rightarrow}})$ are straightforward generalizations of the analogous relations for context free grammars. It is easily seen that EOS systems generate precisely the class of context-free languages.

Whenever an EOS system does not contain productions of the form (a, λ) (called *erasing productions* or λ-productions) we call it *propagating*.

REMARK. (1) Throughout this paper we will assume that the start symbol of an EOS system does not occur in any right hand side of a production rule.

(2) Note that, unlike in context-free grammars, it is required that the substitution of an EOS system is a total mapping. However, a finite substitution h' on Σ^* that is not total, can be "completed" to a total finite substitution h as follows. Let F be a "new" non-terminal symbol, called the *failure symbol*, for which $h(F) = F$. Then, we let $h(a) = F$ for all those symbols a for which h' is not defined. We shall, in fact, use this as a convention throughout this paper: whenever there is no production specified for a symbol, say a, we imply the existence of the production (a, F). The symbol F will be used for this purpose only. □

The mode of rewriting in a selective substitution grammar is given by means of selectors, see, e.g., [RW] and [KR].

Definition 1.2. A *selector* K is a 3-tuple $\langle \Sigma, L, \Delta \rangle$, where Σ is the *alphabet* of K, denoted by Al (K), $\bar{\Sigma} = \{\bar{a}: a \in \Sigma\}$ is the set of *activated symbols* of K (we assume that $\bar{\Sigma} \cap \Sigma = \emptyset$). $L \subset (\Sigma \cup \bar{\Sigma})^*$ is the *language* of K, denoted by La (K) and $\Delta \subset \Sigma$ is the set of *terminal symbols* of K, denoted by Term (K). □

REMARK. An activated symbol is thus denoted by barring the corresponding symbol from the alphabet of the selector. The "bar notation" is used for no other purpose throughout this paper; thus for an alphabet Θ, $\bar{\Theta} = \{\bar{a}: a \in \Theta\}$ (it is assumed that, for all alphabets Σ and Θ of non-activated symbols, $\bar{\Theta} \cap \Sigma = \emptyset$). $\tilde{\Theta}$ will denote $\Theta \cup \bar{\Theta}$. □

A selector added to an EOS system will form a "rewriting system" (where the EOS system provides productions and the selector specifies the mode of rewriting). Certain "consistency conditions" are needed to put together a selector and an EOS system.

Definition 1.3. Let $G = \langle \Sigma, h, S, \Delta \rangle$ be an EOS system and let $K = \langle \Sigma', L, \Delta' \rangle$ be a selector. G and K *fit* if $\Delta \cap (\Sigma' - \Delta') = \emptyset$ and $\Delta' \cap (\Sigma - \Delta) = \emptyset$. □

Definition 1.4. An EOS-*based s-grammar* H is a pair $\langle G, K\rangle$, where

$\mathrm{Base}(H) = G$ is an EOS system and

$\mathrm{Sel}(H) = K$ is a selector that fits G. □

Let $H=\langle G, K\rangle$ be an EOS-based s-grammar where $G=\langle \Sigma, h, S, \Delta\rangle$. Then we specify H also in the form $H=\langle \Sigma, h, S, \Delta, K\rangle$.

To simplify the notation, we will write $\mathrm{Maxr}(G)$ and $\mathrm{Prod}(G)$ to denote $\mathrm{Maxr}(\mathrm{Base}(G))$ and $\mathrm{Prod}(\mathrm{Base}(G))$, respectively.

We will denote the *total alphabet of* an s-grammar G (i.e. the union of the alphabets of $\mathrm{Base}(G)$ and $\mathrm{Sel}(G)$) by $\mathrm{Total}(G)$ and the *total terminal alphabet of* G by $\mathrm{Teral}(G)$.

REMARK. In [RW] and [KR] a selector is just a language and it appears as one component in the specification of a selective substitution grammar. For the purpose of this paper it is necessary to include more structure in the notion of a selector, because we want to be able to treat selectors as separate entities independent of a base. By requiring additionally to the "fit condition" that the alphabets of the base and of the selector are the same (which is a mere technicality) one arrives at the EOS based s-grammars from [KR].

Since in EOS systems productions are available for all symbols (i.e. all symbols are *active*) we allow every symbol in a selector alphabet to be activated. If one considers selective substitution grammars with other kinds of bases (e.g. context-free) one can impose on a selector $K=\langle \Sigma, L, \Delta\rangle$ the condition that $L\subseteq(\Sigma\cup\bar{A})$ should hold, where $A\subset\Sigma$ is the set of active symbols (e.g. $A=\Sigma\setminus\Delta$), or, equivalently, add A as a fourth component to the specification of K. In our study we shall be concerned with EOS-based s-grammars only. We will thus write "s-grammar" rather than "EOS-based s-grammar".

We distinguish between non-terminal and terminal symbols in a selector because in the sequel various constructions of selectors depend on this distinction. The central component of a selector is its language.

When considering equality of selector languages we will assume that selector languages differing by λ only are *still* different. This different treatment of selector languages allows us to look more carefully into their structure and in particular into their role in controlling rewriting in s-grammars. For example, if we would not have this special treatment of selector languages, the language obtained from an arbitrary selector language $\mathrm{La}(K)$ by an inverse weak identity φ that intersperses symbols from an alphabet Θ would always include Θ^*. This may drastically change (as compared to the obvious intention) the structure of rewriting in an s-grammar where the selector with the language $\varphi(\mathrm{La}(K))$ would be used. □

Several kinds of homomorphic mappings will be particularly useful throughout the paper. They are defined now.

(1) Let Σ and Θ be alphabets. Then

— iden_Σ is the coding in $HOM(\tilde{\Sigma}, \Sigma)$ defined by
$\mathrm{iden}_\Sigma(\bar{a})=\mathrm{iden}_\Sigma(a)=a$ for all $a\in\Sigma$.

— $\text{pres}_{\Sigma,\Theta}$ is the weak identity in $HOM(\Sigma, \Theta)$ defined by

$$\text{pres}_{\Sigma,\Theta}(a) = \begin{cases} a & \text{if } a \in \Theta \\ \lambda & \text{otherwise} \end{cases}$$

— $\text{erase}_{\Sigma,\Theta}$ is the weak identity in $HOM(\Sigma, \Sigma - \Theta)$ defined by
$\text{erase}_{\Sigma,\Theta}(a) = \text{pres}_{\Sigma,\Sigma-\Theta}(a)$ for all $a \in \Sigma$.

Whenever Σ is clear from the context we will write iden, pres_Θ and erase_Θ rather than iden_Σ, $\text{pres}_{\Sigma,\Theta}$ and $\text{erase}_{\Sigma,\Theta}$, respectively.

(2) Let $G = \langle \Sigma, h, S, \Delta, K \rangle$ be an s-grammar. Let $\mathbf{I}_\Sigma$ be the set $\{\iota_a : a \in \Sigma\}$ such that $\mathbf{I}_\Sigma \cap \text{Prod}(G) = \emptyset$. Then

— lhbar is the homomorphism in $HOM(\mathbf{I}_\Sigma \cup \text{Prod}(G), \bar{\Sigma})$ defined by
$\text{lhbar}((a, w)) = \bar{a}$ for all $(a, w) \in \text{Prod}(G)$ and
$\text{lhbar}(\iota_a) = a$ for all $\iota_a \in \mathbf{I}_\Sigma$.

— lhs and rhs are the homomorphisms in $HOM(\mathbf{I}_\Sigma \cup \text{Prod}(G), \Sigma)$ defined by
$\text{lhs}(\pi) = \text{iden}(\text{lhbar}(\pi))$ for all $\pi \in \mathbf{I}_\Sigma \cup \text{Prod}(G)$,
$\text{rhs}((a, w)) = w$ for all $(a, w) \in \text{Prod}(G)$ and
$\text{rhs}(\iota_a) = a$ for all $\iota_a \in \mathbf{I}_\Sigma$.

Definition 1.5. Let $G = \langle \Sigma, h, S, \Delta, K \rangle$ be an s-grammar.

— A *derivation of length* 1 *(in G)* is a word $w \in (\mathbf{I}_\Sigma \cup \text{Prod}(G))^*$ with $\text{lhbar}(w) \in \text{La}(K)$, and such that $\text{lhs}(w) \neq \text{lhbar}(w)$.

— For $x, y \in \Sigma^*$ we say that x *directly derives* y *(in G)* if there exists a derivation w of length 1 with $\text{lhs}(w) = x$ and $\text{rhs}(w) = y$; we write then $x \underset{G}{\Rightarrow} y$.

— A *derivation of length* $i > 1$ *(in G)* is a sequence $(w_1, \ldots, w_i)$ of words from $(\mathbf{I}_\Sigma \cup \text{Prod}(G))^*$ such that $(w_1, \ldots, w_{i-1})$ is a derivation of length $i-1$, w_i is a derivation of length 1 and $\text{rhs}(w_{i-1}) = \text{lhs}(w_i)$. For $x, y \in \Sigma^*$ and $i \geqq 1$ we say that x *derives* y *in* i *steps (in G)* if there is a derivation of length i, $(w_1, \ldots, w_i)$, where $\text{lhs}(w_1) = x$ and $\text{rhs}(w_i) = y$; we write then $x \overset{i}{\underset{G}{\Rightarrow}} y$.

— A *derivation (in G)* is a derivation of length i for some $i \geqq 1$; the length of a derivation D is denoted by $|D|$.
If $D = (w_1, \ldots, w_i)$, $i \geqq 1$, is a derivation then D is a *derivation of* $\text{rhs}(w_i)$ *from* $\text{lhs}(w_1)$ *(in G)*.

For $x, y \in \Sigma^*$ we say that x *properly derives* y *(in G)* if there is a derivation of y from x; we write then $x \overset{+}{\underset{G}{\Rightarrow}} y$.

Let $\overset{*}{\underset{G}{\Rightarrow}}$ be the reflexive closure of $\overset{+}{\underset{G}{\Rightarrow}}$. We say that x *derives* y *(in G)* for $x, y \in \Sigma^*$ if $x \overset{*}{\underset{G}{\Rightarrow}} y$.

We write $x \underset{G}{\Rightarrow} y$ whenever $x = y$.

— Let $D = (w_1, \ldots, w_i)$ be a derivation. The *barred trace of* D $(\text{Btrace}(D))$ is the sequence of words $(\text{lhbar}(w_1), \ldots, \text{lhbar}(w_i))$. The *trace of* D $(\text{Trace}(D))$ is the sequence of words $(\text{lhs}(w_1), \ldots, \text{lhs}(w_i), \text{rhs}(w_i))$. If $\text{lhs}(w_1) = S$ then the elements of $\text{Trace}(D)$ are called *sentential forms (of G)*.

— The *language of* G is the set $\mathbf{L}(G) = \{w \in \Delta^* : S \overset{*}{\underset{G}{\Rightarrow}} w\}$. □

The following example will illustrate the above notions.

Example 1.1. Let $G=\langle\{A, B, C, a, b, c, S\}, h, S, \{a, b, c\}, K\rangle$, where

$$K=\langle\{A, B, C, a, b, c, S\},\ \bar{S}\cup\bar{A}a^*\bar{B}b^*\bar{C}c^*,\ \{a, b, c\}\rangle$$

and h is defined by

$$h(A)=\{Aa, a\},\ h(B)=\{Bb, b\},\ h(C)=\{Cc, c\} \text{ and } h(S)=ABC.$$

$$h(d)=d, \text{ for all } d\in\{a, b, c\}.$$

Table 1.1 shows a derivation, its barred trace and its trace.

Table 1.1

Derivation	Btrace	Trace
(S, ABC)	$\bar{S}$	S
$(A, Aa)(B, Bb)(C, Cc)$	$\bar{A}\bar{B}\bar{C}$	ABC
$(A, Aa)\iota_a(B, Bb)\iota_b(C, Cc)\iota_c$	$\bar{A}a\bar{B}b\bar{C}c$	$AaBbCc$
$(A, Aa)\iota_a\iota_a(B, Bb)\iota_b\iota_b(C, Cc)\iota_c\iota_c$	$\bar{A}aa\bar{B}bb\bar{C}cc$	$AaaBbbCcc$
		$AaaaBbbbCccc$

Obviously $\mathbf{L}(G)=\{a^nb^nc^n: n>0\}$. □

Following [KR] various features common to different types of "context-independent" rewriting are formalized and imposed as restrictions on selectors.

Definition 1.6. Let $K=\langle\Sigma, L, \Delta\rangle$ be a selector.

K is *active bar-free* (abf) if, for all $v, w\in\bar{\Sigma}^*$ and for all $a\in\Sigma$, whenever $v\bar{a}w\in L$ then $vaw\in L$.

K is *context bar-free* (cbf) if, for all $v, w\in\bar{\Sigma}^*$ and for all $a\in\Sigma$, whenever $vaw\in L$ then $v\bar{a}w\in L$.

K is *bar-free* (bf) if K is abf and cbf. □

Definition 1.7. Let $K=\langle\Sigma, L, \Delta\rangle$ be a selector.

K is *active symbol-free* (asf) if, for all $v, w\in\bar{\Sigma}^*$ and for all $a\in\Sigma$, whenever $v\bar{a}w\in L$ then $v\bar{\Sigma}w\subset L$.

K is *context symbol-free* (csf) if, for all $v, w\in\bar{\Sigma}^*$ and for all $a\in\Sigma$, whenever $vaw\in L$ then $v\Sigma w\subset L$.

K is *symbol-free* (sf) if K is asf and csf. □

Definition 1.8. Let $K=\langle\Sigma, L, \Delta\rangle$ be a selector and $\Theta\subset\bar{\Sigma}$.

K is *active Θ-interspersed* (Θ-ai) if, for all $v, w\in\bar{\Sigma}^*$ and for all $a\in\Sigma$, whenever $v\bar{a}w\in L$ then $v\Theta^*\bar{a}\Theta^*w\subset L$.

K is *context Θ-interspersed* (Θ-ci) if, for all $v, w\in\bar{\Sigma}^*$ and for all $a\in\Sigma$, whenever $vaw\in L$ then $v\Theta^*a\Theta^*w\subset L$.

K is *Θ-interspersed* (Θ-i) if K is Θ-ai and Θ-ci. □

Definition 1.9. Let $K=\langle\Sigma, L, \Delta\rangle$ be a selector and $\Theta\subset\Sigma$.

K is *Θ-universal* (Θ-u) if, for all $w\in\Theta^*$, there is a $v\in L, v\neq w$, such that iden $(v)=w$.

K is *Θ-occurrence-universal* (Θ-ou) if, for all $w_1, w_2 \in \Theta^*$ and for all $a \in \Theta$, there exist $v_1, v_2 \in \bar{\Theta}^*$ such that $v_1 \bar{a} v_2 \in L$ where iden $(v_1) = w_1$ and iden $(v_2) = w_2$. □

Definition 1.10. Let $K = \langle \Sigma, L, \Delta \rangle$ be a selector and $\Theta \subset \bar{\Sigma}$.

K is *Θ-erasing* (Θ-e) if, for all $w \in L$, $\text{erase}_\Theta(w) \in L$. □

If Θ is an alphabet and G is an s-grammar, then G is abf (cbf, bf, asf, csf, sf, Θ-ai, Θ-ci, Θ-i, Θ-e) if Sel (G) is abf (cbf, bf, asf, csf, sf, Θ-ai, Θ-ci, Θ-i, Θ-e, respectively). If Θ is the alphabet of Base (G) we omit Θ as a prefix in the above acronyms. Moreover we say that G is *universal* (u), respectively *occurrence universal* (ou), whenever Sel (G) is Θ-u, respectively Θ-ou, where Θ is the alphabet of Base (G).

The definitions given above correspond to those given in [KR] (with the exception of Definition 1.10, which does not appear there), for the case that all symbols are active. However, one should take into consideration that in [KR] the above notions are defined directly for s-grammars and hence subject to the assumption that the alphabet of the base and the not specified alphabet of the selector are the same.

The traditionally considered grammar and language families, as seen from the point of view of the theory of s-grammars, are defined using a fixed selector (if the alphabet is fixed); grammars differ only by the set of productions they use. In this way one talks, e.g., about all context-free grammars or all EOS systems (where the selector language is of the form $\Sigma^*(\overline{\Sigma - \Delta})\Sigma^*$ or $\Sigma^*\bar{\Sigma}\Sigma^*$, respectively), or about all EOL systems (where the selector language is of the form $\bar{\Sigma}^+$). To define a family of selectors based on (the structure of) one selector only we proceed as follows.

Definition 1.11. A family of selectors $\mathscr{S}$ is a *selector scheme* if

(a) $\mathscr{S}$ contains a selector K_0 of the form $\langle \{a\}, L_0, \{a\} \rangle$.

(b) For all alphabets Σ and $\Delta \subset \Sigma$ there is exactly one selector K in $\mathscr{S}$ with Al $(K) = \Sigma$ and Term $(K) = \Delta$; it is also required that La $(K) = \varphi_\Sigma(L_0)$, where $\varphi_\Sigma \in FSUB(\bar{a}, \bar{\Sigma})$ is defined by

$$\varphi_\Sigma(a) = \Sigma \quad \text{and} \quad \varphi_\Sigma(\bar{a}) = \bar{\Sigma}. \quad \square$$

It is straightforward to see that if $\mathscr{S}$ is a selector scheme and $K \in \mathscr{S}$ then K is sf. Moreover, for every sf selector K there exists exactly one selector scheme $\mathscr{S}$ with $K \in \mathscr{S}$. As a matter of fact a selector scheme represents the selector of a pattern grammar (see [KR]).

Note that whenever a selector scheme $\mathscr{S}$ contains an abf (cbf, bf) selector then all the selectors in $\mathscr{S}$ are abf (cbf, bf, respectively). Hence we can speak of an abf (cbf, bf) *selector scheme.*

In the sequel we will attempt to investigate properties of selectors that allow us to perform various operations on s-grammars. We will consider two approaches in parallel (whenever possible): properties of general selector families on one hand and properties of selector schemes (or selectors) on the other hand. Although we distinguish between non-terminal and terminal symbols in an individual selector for the purpose of this paper it suffices to assume that every family $\mathscr{K}$ of selectors that we consider satisfies the following condition:

If $\langle \Sigma, L, \Delta \rangle \in \mathscr{K}$, then, for every $\theta \subset \Sigma$, $\langle \Sigma, L, \theta \rangle \in \mathscr{K}$.

The notion of closure for language families is extended to families of selectors in the following way.

Definition 1.12. Let $\mathscr{K}$ be a family of selectors and let τ be an n-ary mapping on languages, $n \geqq 1$. We say that $\mathscr{K}$ is *closed under* τ if, for every $K_1, \ldots, K_n \in \mathscr{K}$, there is a selector $K \in \mathscr{K}$ such that

$$\mathrm{Al}(K) = \mathrm{iden}\big(\mathrm{alph}\,(\tau((\mathrm{Al}(K_1))^*, \ldots, (\mathrm{Al}(K_n))^*))\big) \quad \text{and}$$

$$\mathrm{La}(K) = \tau\big(\mathrm{La}(K_1), \ldots, \mathrm{La}(K_n)\big). \quad \square$$

The word "universal" has been used in the theory of s-grammars to express different phenomena (see, e.g., [RW] and [KR]). To avoid confusion we use the following notion to describe "universality with respect to generative power".

Definition 1.13. Let $\mathscr{L}$ be a family of languages. An EOS system $G = \langle \Sigma, h, S, \Delta \rangle$ is an *s-generator of* $\mathscr{L}$ (*with respect to* Δ) if for all $L \in \mathscr{L}$ with $L \subset \Delta^*$ there exists a selector K that fits G such that $\mathbf{L}(\langle G, K \rangle) = L$. $\square$

2. The existence of normal forms

Let $\mathscr{G}$ be a family of grammars. If $\mathscr{C}$ is a set of conditions and if the subclass $\mathscr{G}_{\mathscr{C}}$ (consisting of all those grammars of $\mathscr{G}$ that satisfy $\mathscr{C}$) still generates all the languages generated by grammars from $\mathscr{G}$ then we say that $\mathscr{C}$ constitutes a *normal form* of $\mathscr{G}$.

Investigation of normal forms for various classes of grammars constitutes a major research topic in formal language theory. In this section we investigate the existence of various normal forms in the general framework of s-grammars.

The basic conditions (imposed on s-grammars) that we will consider in this paper are defined as follows.

Definition 2.1. Let $G = \langle \Sigma, h, S, \Delta, K \rangle$ be an s-grammar.

— A symbol $a \in \Sigma$ is *versatile* (in G) if there is a production $(a, w) \in \mathrm{Prod}\,(G)$ with $w \neq a$. Let $a \in \Sigma$. A rule (a, w) is a *chain* in G if w consists of a single versatile letter.

— G is *chain-free* if either there are no chains in G or every chain in G is of the form (S, a), where $a \in \Delta^*$ is such that, for all $w \in h(a)$, w contains only non-versatile symbols and $w \in \Delta^*$ implies $w = a$.

— G is *synchronized* if, for all $a \in \Delta$, $a \underset{\mathrm{Base}(G)}{\overset{+}{\Rightarrow}} w$ implies that w is not in Δ^*.

— G is *binary* if, for all $a \in \Sigma$, $w \in h(a)$ implies that $|w| \leqq 2$.

— G is *propagating* if for all $a \in \Sigma - S$, λ is not in $h(a)$.

— G is *right-recursive (left-recursive)* if there exists a versatile symbol a and a word $w \in \Sigma^*$, such that $a \underset{\mathrm{Base}(G)}{\overset{+}{\Rightarrow}} wa$ ($a \underset{\mathrm{Base}(G)}{\overset{+}{\Rightarrow}} aw$, respectively). $\square$

REMARK. (1) The above definition adopts the notions of chain-freeness, synchronization, etc. as used in the theory of context-free grammars and ETOL systems to the framework of s-grammars. For example the classical notion of chain-freeness is modified by the use of versatile symbols to account for the fact that terminal symbols can also be rewritten.

(2) We will use the above terminology (chain-freeness, synchronization, etc.) also for ETOL systems. Although ETOL systems are not *s*-grammars this should not lead to confusion. □

In the rest of this section we will demonstrate that the restrictions discussed above on the form of *s*-grammars, even when combined with additional restrictions on the properties of selectors used, do not affect the language-generating power (of the whole class of *s*-grammars).

The following results are generalized versions of theorems in [KR]. The proofs are similar. However, basic constructions in the proofs had to be modified. In the proofs below we provide such basic constructions, and leave to the reader the (not difficult) task of proving that these constructions yield *s*-grammars with properties as required in the statement of the theorems in question.

Theorem 2.1. *Every language L can be generated by a chain-free synchronized propagating non-left-recursive (or non-right-recursive) binary and* bf *s-grammar.*

Proof. Let $L \subseteq \Delta^*$. We define $G = \langle \Sigma, h, S, \Delta, K \rangle$ as follows:

$$\Sigma = \{S, T, F\} \cup \Delta \cup \Theta_2, \quad \text{where} \quad \Theta_2 = \{[a, b]\colon a, b \in \Delta\}$$

such that Δ, $\{S, T, F\}$ and Θ_2 are pairwise disjoint. h is defined by

$$h(S) = \{aT\colon a \in \Delta\} \cup \{w \in L\colon |w| \leqq 2\} \cup \{a[b, c]\colon a, b, c \in \Delta\}.$$

$$h(T) = \{aT\colon a \in \Delta\} \cup \{a[b, c]\colon a, b, c \in \Delta\}.$$

$$h([a, b]) = ab \quad \text{and}$$

$$h(a) = F \quad \text{for all} \quad a \in \Delta \cup \{F\}.$$

$$K = \langle \Sigma, \tilde{\Sigma}^*(\tilde{S} \cup \tilde{T}) \cup \bigcup_{a, b \in \Delta} L^*_{[a, b]}[\widetilde{a, b}]\Delta \rangle, \quad \text{where}$$

$$L_{[a, b]} = \{w \in \tilde{\Delta}^*\colon \operatorname{iden}(wab) \in L\} \quad \text{for all} \quad a, b \in \Delta.$$

It is easily seen that $L = \mathbf{L}(G)$ and, moreover, G is chain-free, synchronized, propagating, non-left-recursive, bf and binary. □

Theorem 2.1 yields immediately to the following result.

Corollary 2.1. *For every s-grammar there is an equivalent* bf *s-grammar that is chain-free, synchronized, propagating, non-left-recursive (or non-right-recursive) and binary.*

Theorem 2.2. *Every language L can be generated by a chain-free synchronized propagating non-left-recursive (or non-right-recursive) binary* abf, asf *and* u *s-grammar.*

Proof. Let $L \subseteq \Delta^*$. We define $G = \langle \Sigma, h, S, \Delta, K \rangle$ as follows.

$$\Sigma = \{S, T, F\} \cup \Delta \cup \Xi_4 \cup \Theta_4, \quad \text{where}$$

$$\Xi_4 = \{\langle a, b, c, d\rangle\colon a, b, c, d \in \Delta\},$$

$$\Theta_4 = \{[a, b, c, d]\colon a, b, c, d \in \Delta\}$$

and Δ, $\{S, T, F\}$, Ξ_4 and Θ_4 are pairwise disjoint. h is defined by

$$h(S) = \{w \in L: |w| \leqq 2\} \cup \{aT: a \in \Delta\} \cup$$
$$\{a[b, c, b, c]: abc \in L\} \cup \{\langle a, b, c, d\rangle [a, b, c, d]: a, b, c, d \in \Delta\},$$
$$h(T) = \{aT: a \in \Delta\} \cup \{\langle a, b, c, d\rangle [a, b, c, d]: a, b, c, d \in \Delta\}.$$

$h(\langle a, b, c, d\rangle) = ab$ and $h([a, b, c, d]) = cd$ for all $a, b, c, d \in \Delta$, and

$$h(a) = F \text{ for all } a \in \Delta \cup \{F\}$$

$$K = \langle \Sigma, \Sigma^* \tilde{\Sigma} \cup \{a_1 \ldots a_{k-4} b [a_{k-3}, a_{k-2}, a_{k-1}, a_k]: a_1 \ldots a_k \in L, \ k \geqq 4 \text{ and } b \in \tilde{\Sigma}\}, \Delta\rangle.$$

Clearly G is chain-free, synchronized, propagating, non-left-recursive, binary, abf, asf and u. It can easily be seen from the construction that $L = \mathbf{L}(G)$. □

Corollary 2.2. *For every s-grammar there is an equivalent* abf, asf, *and* u *s-grammar that is chain-free, synchronized, propagating, non-left-recursive (or non-right-recursive) and binary.*

Theorem 2.3. *Every language L can be generated by a chain-free synchronized propagating non-left-recursive (or non-right-recursive) binary* abf *and* ai *s-grammar.*

Proof. Let $L \subseteq \Delta^*$. We define $G = \langle \Sigma, h, S, \Delta, K\rangle$ as follows.

$$\Sigma = \{S, T, F\} \cup \Delta \cup \Theta_2 \cup \Theta_4, \text{ where}$$
$$\Theta_2 = \{[a, b]: a, b \in \Delta\},$$
$$\Theta_4 = \{[a, b, c, d]: a, b, c, d \in \Delta\}$$

and Δ, $\{S, T, F\}$, Θ_2 and Θ_4 are pairwise disjoint. h is defined by

$$h(S) = \{w \in L: |w| \leqq 2\} \cup \{aT: a \in \Delta\} \cup$$
$$\cup \{[a, b, a, b]c: abc \in L\} \cup \{[a, b, c, d][c, d]: a, b, c, d \in \Delta\},$$
$$h(T) = \{aT: a \in \Delta\} \cup \{[a, b, c, d][c, d]: a, b, c, d \in \Delta\},$$
$$h([a, b, c, d]) = h([a, b]) = ab \text{ for all } a, b, c, d \in \Delta \text{ and}$$
$$h(a) = F \text{ for all } a \in \Delta \cup \{F\}.$$

$$K = \langle \Sigma, \Sigma^* \{\widetilde{S, T}\} \Sigma^* \cup \bigcup_{a, b \in \Delta} L_{[a, b]} \Sigma^* [\widetilde{a, b}] \Sigma^* \cup \bigcup_{a, b, c, d \in \Delta} \Sigma^* \widetilde{[a, b, c, d]} \Sigma^*, \Delta\rangle, \text{ where}$$

$$L_{[a, b]} = \{a_1 \ldots a_{n-4} [a_{n-3}, a_{n-2}, a, b]: a_1 \ldots a_{n-2} ab \in L, n \geqq 4\}.$$

It is easily seen that $L = \mathbf{L}(G)$ and, moreover, G is chain-free, synchronized, propagating, non-left-recursive, binary, abf and ai. □

Corollary 2.3. *For every s-grammar there is an equivalent* abf *and* ai *s-grammar that is chain-free, synchronized, propagating, non-left-recursive (non-right-recursive) and binary.*

Theorem 2.4. *Every language L can be generated by a chain-free synchronized propagating non-left-recursive (non-right-recursive) binary* cbf, csf, ci *and* ou *s-grammar.*

Proof. Let $L \subseteq \Delta^*$. Let $G = \langle \Sigma, h, S, \Delta, K \rangle$, where

$$\Sigma = \{S, T, F\} \cup \Delta \cup \Theta_2 \cup \Theta_3 \cup \Theta_4 \cup \Xi_2, \quad \text{where}$$

$$\Theta_2 = \{[a, b]\colon a, b \in \Delta\},$$

$$\Theta_3 = \{[a, b, c]\colon a, b, c \in \Delta\},$$

$$\Theta_4 = \{[a, b, c, d]\colon a, b, c, d \in \Delta\},$$

$$\Xi_2 = \{\langle a, b \rangle\colon a, b \in \Delta\}$$

and Δ, $\{S, T, F\}$, Θ_2, Θ_3, Θ_4 and Ξ_2 are pairwise disjoint. h is defined by

$$h(S) = \{w \in L\colon |w| \leqq 2\} \cup \{[a, b]T\colon a, b \in \Delta\} \cup$$
$$\cup \{a\langle b, c\rangle\colon abc \in L\} \cup \{\langle a, b\rangle\langle c, d\rangle\colon abcd \in L\},$$

$$h(T) = \{[a, b]T\colon a, b \in \Delta\} \cup \{[a, b, c, d]d\colon a, b, c, d \in \Delta\} \cup$$
$$\cup \{[a, b, c]c\colon a, b, c \in \Delta\},$$

$$h([a, b, c, d]) = \langle a, b\rangle c \quad \text{for all} \quad a, b, c, d \in \Delta,$$

$$h([a, b, c]) = h(\langle a, b\rangle) = ab \quad \text{for all} \quad a, b, c \in \Delta,$$

$$h([a, b]) = ab \quad \text{for all} \quad a, b \in \Delta \quad \text{and}$$

$$h(a) = F \quad \text{for all} \quad a \in \Delta \cup \{F\}.$$

$$K = \langle \Sigma, \bar{\Sigma}^+ \cup \tilde{\Sigma}^* \bar{T} \cup \overline{L'} \tilde{\Sigma}^* \cup \tilde{\Sigma}^* \{\overline{\langle a, b \rangle}\colon a, b \in \Delta\} \tilde{\Sigma}^*, \Delta \rangle, \quad \text{where}$$

$$\overline{L'} = \{\overline{[a_1, a_2]} \ldots \overline{[a_{2k-5}, a_{2k-4}]} \overline{[a_{2k-3}, a_{2k-2}, a_{2k-1}, a_{2k}]}\colon a_1 \ldots a_{2k} \in L,\ k \geqq 3\} \cup$$
$$\cup \{\overline{[a_1, a_2]} \ldots \overline{[a_{2k-5}, a_{2k-4}]} \overline{[a_{2k-3}, a_{2k-2}, a_{2k-1}]}\colon a_1 \ldots a_{2k-1} \in L,\ k \geqq 3\}.$$

Clearly G is chain-free, synchronized, propagating, non-left-recursive, binary, cbf, csf, ci and ou. It can easily be seen from the construction that $L = \mathbf{L}(G)$. □

Corollary 2.4. *For every s-grammar there is an equivalent* cbf, csf, ci *and* ou *s-grammar that is chain-free, synchronized, propagating, non-left-recursive (non-right-recursive) and binary.*

Theorem 2.5. *Every language L can be generated by a chain-free synchronized propagating non-left-recursive (or non-right-recursive) binary* csf *and* i *grammar.*

Proof. Let $L \subseteq \Delta^*$. We define $G = \langle \Sigma, h, S, \Delta, K \rangle$ as follows.

$$\Sigma = \{S, T, F\} \cup \Delta \cup \Theta_2 \cup \Theta_3 \cup \Xi_2, \quad \text{where}$$

$$\Theta_2 = \{[a, b]\colon a, b \in \Delta\},$$

$$\Theta_3 = \{[a, b, c]\colon a, b, c \in \Delta\},$$

$$\Xi_2 = \{\langle a, b\rangle\colon a, b \in \Delta\}$$

and Δ, $\{S, T, F\}$, Θ_2, Θ_3 and Ξ_2 are pairwise disjoint. h is defined by

$$h(S) = \{w \in L: |w| \leqq 2\} \cup \{\langle a, b\rangle T: a, b \in \Delta\} \cup$$
$$\cup \{\langle a, b\rangle c: abc \in L\} \cup \{\langle a, b\rangle [c, d]: a, b, c, d \in \Delta\},$$
$$h(T) = \{[a, b]T: a, b \in \Delta\} \cup \{[a, b][c, d]: a, b, c, d \in \Delta\} \cup$$
$$\cup \{[a, b, c]c: a, b, c \in \Delta\},$$
$$h(\langle a, b\rangle) = h([a, b]) = h([a, b, c]) = ab \quad \text{for all} \quad a, b, c \in \Delta \quad \text{and}$$
$$h(a) = F \quad \text{for all} \quad a \in \Delta \cup \{F\}.$$

$$K = \langle \Sigma, \Sigma^* \{\overline{S, T}\} \Sigma^* \cup \psi(\overline{L'}), \Delta\rangle, \quad \text{where}$$

$$\overline{L'} = \{\overline{\langle a_1, a_2\rangle [a_3, a_4]} \ldots \overline{[a_{2n-1}, a_{2n}]}: a_1 \ldots a_{2n} \in L, n \geqq 2\} \cup$$
$$\cup \{\overline{\langle a_1, a_2\rangle [a_3, a_4]} \ldots \overline{[a_{2n-1}, a_{2n}, a_{2n+1}]}: a_1 \ldots a_{2n+1} \in L, n \geqq 2\} \cup$$
$$\cup \{\overline{\langle a, b\rangle}: abc \in L, \text{ for some } c \in \Delta\}$$

and ψ is the substitution on $\bar{\Sigma}^*$ defined by $\psi(\bar{a}) = \Sigma^* \bar{a} \Sigma^*$ for all $a \in \Sigma$. It is easily seen that $L = \mathbf{L}(G)$ and, moreover, G is chain-free, synchronized, propagating, non-left-recursive, binary, csf and i. □

Corollary 2.5. *For every s-grammar there is an equivalent* csf *and* i *s-grammar that is chain-free, synchronized, propagating, non-left-recursive (or non-right-recursive) and binary.*

Theorem 2.6. *Every language L can be generated by a chain-free synchronized propagating non-left-recursive (or non-right-recursive) binary* abf, ci *and* u *s-grammar.*

Proof. Let $L \subseteq \Delta^*$. We define $G = \langle \Sigma, h, S, \Delta, K\rangle$ as follows.

$$\Sigma = \{S, T, F\} \cup \Delta \cup \Theta_2 \cup \Theta_3 \cup \Xi_2, \quad \text{where}$$
$$\Theta_2 = \{[a, b]: a, b \in \Delta\},$$
$$\Theta_3 = \{[a, b, c]: a, b, c \in \Delta\},$$
$$\Xi_2 = \{\langle a, b\rangle: a, b \in \Delta\}$$

and Δ, $\{S, T, F\}$, Θ_2, Θ_3 and Ξ_2 are pairwise disjoint. h is defined by

$$h(S) = \{w \in L: |w| \leqq 2\} \cup \{[a, b]T: a, b \in \Delta\} \cup$$
$$\cup \{a[b, c]: abc \in L\} \cup \{[a, b][c, d]: a, b, c, d \in \Delta\} \cup$$
$$\cup \{[a, b][c, d, e]: a, b, c, d, e \in \Delta\},$$
$$h(T) = \{[a, b]T: a, b \in \Delta\} \cup$$
$$\cup \{[a, b][c, d]: a, b, c, d \in \Delta\} \cup \{[a, b][c, d, e]: a, b, c, d, e \in \Delta\},$$
$$h([a, b, c]) = a\langle b, c\rangle \quad \text{for all} \quad a, b, c \in \Delta,$$
$$h(\langle a, b\rangle) = h([a, b]) = ab \quad \text{for all} \quad a, b \in \Delta,$$
$$h(a) = F \quad \text{for all} \quad a \in \Delta \cup \{F\}.$$
$$K = \langle \Sigma, \Sigma^* \bar{\Sigma} \cup \psi(\overline{L'}), \Delta\rangle$$

where ψ is the substitution on $\bar{\Sigma}^*$ defined by $\psi(\bar{a})=\{\bar{a}\}\cup\Sigma^* a\Sigma^*$ for all $a\in\Sigma$ and

$$\overline{L'}=\{\overline{[a_1, a_2]}\ldots\overline{[a_{2k-1}, a_{2k}]}\colon a_1\ldots a_{2k}\in L\}\cup$$
$$\cup\{\overline{[a_1, a_2]}\ldots\overline{[a_{2k-1}, a_{2k}, a_{2k+1}]}\colon a_1\ldots a_{2k+1}\in L\}.$$

Clearly G is chain-free, synchronized, propagating, non-left-recursive, binary, abf, ci and u. It can easily be seen from the construction that $L=\mathbf{L}(G)$. □

Corollary 2.6. *For every s-grammar there is an equivalent* abf, ci *and* u *s-grammar that is chain-free, synchronized, non-left-recursive (or non-right-recursive) and binary.*

We observe that the construction used in the proof of Theorem 2.1 yields a rather strong normal form for *s*-generators.

Corollary 2.7. *Let $\mathscr{L}_0$ be the family of all languages containing no word of length less than* 3. *Then there is an* EOS *grammar that is chain-free, synchronized, propagating, non-left-recursive (or non-right-recursive) and binary, that is an s-generator of $\mathscr{L}_0$.*

REMARK. The common feature of all the constructions used in this section to obtain normal forms is "language-dependency" rather than "grammar-dependency". That is, to demonstrate that *s*-grammars satisfying a particular set of conditions can generate a language, we would use this language explicitly in constructing a selector; this is done without any knowledge whatsoever about the way that this language is grammatically generated. Thus, in general, our results are per se non-effective. From the grammatical point of view it is certainly more desirable to obtain normal forms starting with a language given through an *s*-grammar where, moreover, the resulting *s*-grammar has a selector of the same kind (belonging to the same selector family) as the selector of the originally given *s*-grammar. The rest of this paper will consider the latter approach. □

3. ETOL systems and CS grammars — the *s*-grammars point of view

In the sequel we will investigate properties of (families of) selectors which allow one to perform various operations on *s*-grammars using these selectors. In this section we look at ETOL systems and the family of context-sensitive grammars from the point of view of *s*-grammars. The results of this section allow us to provide some applications of the results obtained in the sequel; they also give us a sort of guideline as to which operations on families of selectors (not) to consider.

We shall specify an ETOL *system* as a quadruple $\langle\Sigma, \mathscr{H}, S, \Delta\rangle$, where Σ, S and Δ are like in EOS systems and $\mathscr{H}$ is a finite set $\{h_1, \ldots, h_{\#\mathscr{H}}\}\subseteq FSUB(\Sigma, \Sigma)$ (of *tables*) such that $h_i(a)\neq\emptyset$ and $h_i(a)\cap\Sigma^* S\Sigma^*=\emptyset$ for all $a\in\Sigma$ and $1\leqq i\leqq\#\mathscr{H}$. (This is a normal form for ETOL systems, as, e.g., shown in [RS].)

Whenever an ETOL system is *propagating* we call it an EPTOL *system*.

Whenever for an ETOL system $G=\langle\Sigma, \mathscr{H}, S, \Delta\rangle$ the set $\mathscr{H}$ is a singleton, we call G an EOL system.

Theorem 3.1. *Let $G=\langle\Sigma, h, S, \Delta, K\rangle$ be an s-grammar, where* La (K) *is a union of $n\geqq 1$ letter monoids. Then* $\mathbf{L}(G)$ *can be generated by an* ETOL *system with n tables.*

Proof. We shall construct an ETOL system H that is equivalent to G. Let $\mathrm{La}(K)$ be of the form $\bigcup_{i=1}^{n}(\Theta_i \cup \bar{\Phi}_i)^*$. Let H be the ETOL system $\langle \Sigma \cup \{F\}, \{h_1, \ldots, h_n\}, S, \Delta\rangle$, where F is the failure symbol and h_i is defined as follows, for $1 \leqq i \leqq n$:

$$h_i(a) = \begin{cases} h(a) & \text{if } a \in \Phi_i - \Theta_i \\ a & \text{if } a \in \Theta_i - \Phi_i \\ h(a) \cup a & \text{if } a \in \Theta_i \cap \Phi_i \\ F & \text{otherwise.} \end{cases}$$

It is easy to see that there is a derivation in G if and only if there is a corresponding derivation in H, because whenever a sentential form is rewritten in G using a word in $(\Theta_i \cup \bar{\Phi}_i)^*$ it can be rewritten in H in the same way using h_i and vice versa. □

Corollary 3.1. *A language L can be generated by an* ETOL *system with n tables if and only if it can be generated by an s-grammar with a selector that is the union of n letter monoids.*

Proof. The *if* direction follows from Theorem 3.1. The *only if* direction was shown in [EMR], where it was proved that every language that can be generated by an ETOL system with n tables can be generated by an n SC-grammar, i.e. an s-grammar with a selector the language of which is of the form $\bigcup_{i=1}^{n} \bar{\Phi}_i^*$. □

A *context-sensitive grammar* G will be specified in Penttonen Normal Form. Hence G is a quadruple $\langle \Sigma, P, S, \Delta\rangle$ where Σ, S and Δ are as in EOS systems and P is a set of productions of the form (b, a) or (b, cd) or (bc, bd), where $b, c, d \in \Sigma - \Delta$ and $a \in \Delta$.

That grammars in this form generate all (and only) context-sensitive languages was shown in [P].

Theorem 3.2. *For every context-sensitive language L there is a propagating s-grammar G such that $\mathbf{L}(G) = L$ and $\mathrm{La}(\mathrm{Sel}(G))$ is a word monoid.*

Proof. Let $H = \langle \Sigma', P, S, \Delta\rangle$ be a context-sensitive grammar for L. Let each production in P be numbered distinctly, by a number between 1 and $\#P$, in an arbitrary manner. Let $(A_i b_i, A_i w_i)$ be the i'th production in P, where $A_i \in (\Sigma' - \Delta) \cup \{\lambda\}$, $b_i \in \Sigma' - \Delta$ and $w_i \in \Sigma'^*$.

Let G be the s-grammar $\langle \Sigma, h, S, \Delta, K\rangle$ where

$$\Sigma = \Sigma' \cup \{a^{(j)} \colon a \in \Sigma' \text{ and } 1 \leqq j \leqq \#P\}$$

where all the $a^{(j)}$ are new symbols. h is defined by

$$h(a) = \{a^{(j)} \colon 1 \leqq j \leqq \#P\} \text{ for all } a \in \Sigma',$$

$$h(a^{(i)}) = \begin{cases} w_i & \text{if } b_i = a \\ a & \text{otherwise} \end{cases} \text{ for all } a \in \Sigma \text{ and } 1 \leqq i \leqq \#P.$$

$$K = \left\langle \Sigma, \left(\tilde{\Sigma}' \cup \bigcup_{i=1}^{\#P} A_i \overline{b_i^{(i)}} \right)^*, \Delta \right\rangle.$$

The equivalence of G and H is easily established. Hence the theorem holds. □

Corollary 3.2. *A language is context-sensitive if and only if it can be generated by a propagating s-grammar with a selector the language of which is a word monoid.*

Proof. The *if* direction can be shown by a standard automata-theoretic construction. The *only if* direction follows from Theorem 3.2. □

Corollary 3.3. *Every context-sensitive language can be generated by an s-grammar with a selector of the form* $\langle \Phi, h(\Theta^*), \Xi \rangle$, *where* Θ *and* Φ *are alphabets,* $h \in HOM(\Theta, \tilde{\Phi})$ *and* $\Xi \subseteq \Phi$.

Proof. Immediate from Corollary 3.2. □

This result implies that we should be careful when using homomorphisms or (finite) substitutions in *s*-grammar constructions because we are liable to arrive at very large language families. Corollaries 3.1 and 3.3 show that a homomorphism may take us from the family of EOL languages to the family of context-sensitive languages.

4. On shadows

The following grammatical construction will often be used in the sequel. It generalizes the classical construction used to obtain the synchronized version of an EOL system (see, e.g., [RS]) to the case of *s*-grammars. The main goal of this construction is to obtain an equivalent *s*-grammar where the "representational" and the "generative" role for terminal symbols are separated.

Definition 4.1. (1) Let Θ and Γ be alphabets such that $\Theta \cap \Gamma = \varnothing$ and $\#\Theta = \#\Gamma$. Let ϱ be a fixed injective coding in $HOM(\Theta, \Gamma)$.

For an alphabet Φ such that $\Theta \subseteq \Phi$ and $\Phi \cap \Gamma = \varnothing$ we define the finite substitutions $\mathrm{shad}_{\Phi,\varrho}$ and $\mathrm{fshad}_{\Phi,\varrho}$ in $FSUB(\tilde{\Phi}, \tilde{\Phi} \cup \tilde{\Gamma})$ as follows.

For all $a \in \tilde{\Phi} - \tilde{\Theta}$,

$$\mathrm{shad}_{\Phi,\varrho}(a) = \mathrm{fshad}_{\Phi,\varrho}(a) = a,$$

for all $a \in \Theta$,

$$\mathrm{shad}_{\Phi,\varrho}(a) = \mathrm{fshad}_{\Phi,\varrho}(a) = \{a, \varrho(a)\}$$

and, for all $\bar{a} \in \bar{\Theta}$,

$$\mathrm{shad}_{\Phi,\varrho}(\bar{a}) = \overline{\varrho(a)} \quad \text{and}$$

$$\mathrm{fshad}_{\Phi,\varrho}(\bar{a}) = \{\bar{a}, \overline{\varrho(a)}\}.$$

(2) Let $K = \langle \Phi, L, \Theta \rangle$ be a selector and let Γ be an alphabet such that Φ, Θ, Γ and ϱ are as in (1).

The *shadow of* K *with respect to* ϱ, denoted by $\mathrm{sh}_\varrho(K)$, is the selector $\langle \Phi \cup \Gamma, \mathrm{shad}_{\Phi,\varrho}(L), \Theta \rangle$ and the *full shadow of* K *with respect to* ϱ, denoted by $\mathrm{fsh}_\varrho(K)$, is the selector $\langle \Phi \cup \Gamma, \mathrm{fshad}_{\Phi,\varrho}(L), \Theta \rangle$.

(3) Let $G = \langle \Sigma, h, S, \Delta, K \rangle$ be an *s*-grammar and let $\Phi = \mathrm{Total}(G)$ and $\Theta = \mathrm{Teral}(G)$. Let Φ, Θ, Γ and ϱ be as in (1). The *shadow of* G *with respect to* ϱ is the *s*-grammar $\langle \Sigma \cup \varrho(\Delta) \cup \{F\}, h', S, \Delta, \mathrm{sh}_{\varrho'}(K) \rangle$ where ϱ' is the restriction

of ϱ to Term (K), F is a new symbol,

$$h'(\varrho(a)) = \mathrm{shad}_{\Phi,\varrho}(h(a)), \quad \text{for all} \quad a \in \Sigma \quad \text{and}$$

$$h'(a) = \{F\}, \quad \text{for all} \quad a \in \Delta \cup \{F\}.$$

The full shadow of G *with respect to* ϱ is the s-grammar $\langle \Sigma \cup \varrho(\Delta) \cup \{F\}, h'', S, \Delta, \mathrm{fsh}_{\varrho'}(K)\rangle$, where ϱ' is the restriction of ϱ to Term (K), F is a new symbol,

$$h'(\varrho(a)) = \mathrm{fshad}_{\Phi,\varrho}(h(a)), \quad \text{for all} \quad a \in \Sigma - \Delta \quad \text{and}$$

$$h'(a) = \{F\}, \quad \text{for all} \quad a \in \Delta \cup \{F\}.$$

(4) A *(full) shadow of* a selector K is the (full) shadow of K with respect to some injective coding ϱ.

A *(full) shadow of* an s-grammar G is the (full) shadow of G with respect to some injective coding ϱ. □

Theorem 4.1. *Let G be an s-grammar. Then, for all shadows and full shadows H of G, $\mathbf{L}(H) = \mathbf{L}(G)$.*

Proof. Immediate from the definition. □

Theorem 4.2. *Let G be an s-grammar and let H be a shadow of G. If G is* abf (ai, ci, csf, e) *then so is H.*

Proof. Immediate from the definition. □

As a matter of fact, a shadow of an s-grammar is not necessarily cbf or asf or u or ou, if the original s-grammar is cbf or asf or u or ou, respectively. This observation should be contrasted with the following result.

Theorem 4.3. *Let G be an s-grammar and H a full shadow of G. If G is* abf (cbf, ai, ci, asf, csf, ou, u, e) *then so is H.*

Proof. Immediate from the definition. □

Let Σ and Θ be alphabets and let φ be a mapping from $\tilde{\Sigma}^*$ into $\tilde{\Theta}^*$.

— φ is *bar-preserving* if $\varphi(a) \subseteq \Theta^*$ and $\varphi(\bar{a}) \subseteq \bar{\Theta}^*$ for all $a \in \Sigma$.

— φ is *bar-invariant* if it is bar-preserving and, furthermore, $\varphi(\bar{a}) = \overline{\varphi(a)}$ for all $a \in \Sigma$. □

Theorem 4.4. *Let $\mathscr{K}$ be a family of selectors that is closed under bar-preserving letter-to-letters substitution. Then for every $K \in \mathscr{K}$ every shadow of K is also in $\mathscr{K}$.*

Proof. Immediate by the definition of a shadow. □

Theorem 4.5. *Let $\mathscr{K}$ be a family of selectors that is closed under bar-invariant letter-to-letters substitution. Then for every $K \in \mathscr{K}$ every full shadow of K is also in $\mathscr{K}$.*

Proof. Immediate by the definition of a full shadow. □

Lemma 4.1. *Let $\mathscr{S}$ be a selector scheme. Then $\mathscr{S}$ is closed under bar-invariant letter-to-letters substitution.*

Proof. Immediate from the definition of a selector scheme. □

Corollary 4.1. *Let $\mathscr{S}$ be a selector scheme and let $K \in \mathscr{S}$. Then every full shadow of K is also in $\mathscr{S}$.*

Proof. Immediate from Lemma 4.1 and Theorem 4.5. □

5. Synchronization

In this section we will investigate the possibilities of obtaining synchronized normal forms for *s*-grammars. We start by using the operations of shadowing and full shadowing.

Theorem 5.1. *Let G be an s-grammar. Then every shadow and full shadow of G is synchronized and equivalent to G.*

Proof. Immediate from the definition of shadows and full shadows and Theorem 4.1. □

However the use of the full shadow construction (rather than the shadow construction) gives us additional advantages in the sense that we stay within a family of selectors satisfying some additional properties (see Theorem 4.3).

Theorem 5.2. *Let $\mathscr{K}$ be a family of selectors that is closed under bar-invariant letter-to-letters substitution. Then for every s-grammar G with* Sel $(G) \in \mathscr{K}$ *there is an equivalent synchronized s-grammar H with* Sel $(H) \in \mathscr{K}$.

Proof. This is immediate by Theorems 4.1, 4.5 and 5.1. □

REMARK. If one changes the statement of the above theorem by requiring "bar-preserving" rather than "bar-invariant", then the proof (of such a modified version) of the theorem can be obtained by using the shadow operation. □

Continuous grammars were introduced in [EMR] as a "missing link" between sequential and parallel rewriting as seen from the theory of *s*-grammars. In the sequel we will apply some of our results also to continuous grammars.

Definition 5.1. Let $n \geqq 1$. An *(n-)continuous selector* is a selector K, where La (K) is of the form $\bigcup_{i=1}^{n} \Theta_i^* \bar{\Phi}_i^* \Xi_i^*$ for alphabets $\Theta_1, \ldots, \Theta_n, \Phi_1, \ldots, \Phi_n$ and $\Xi_1, \ldots, \Xi_n$ of symbols without bars.

An *s*-grammar with an n-continuous selector is termed a *(n-)continuous grammar.* A *continuous language* is the language of a continuous grammar. □

Corollary 5.1. *Let $n \geqq 1$. For every n-continuous grammar there is an equivalent synchronized n-continuous grammar.*

Proof. This is immediate from Theorem 5.2 because bar-invariant letter-to-letters substitutions preserve the structure of selectors of n-continuous grammars. □

Corollary 5.2. *For every* ETOL *system there is an equivalent synchronized* ETOL *system.*

Proof. This is an immediate consequence of Corollary 3.1 and Theorem 5.2. □

Corollary 5.3. *For every* EOL *system there is an equivalent synchronized* EOL *system.*

Proof. Immediate from Corollary 3.1 and Theorem 5.2. □

Corollary 5.4. *Let* $\mathscr{S}$ *be a selector scheme and let* G *be an s-grammar with* $\mathrm{Sel}(G)\in\mathscr{S}$. *Then there is an equivalent synchronized s-grammar* H *with* $\mathrm{Sel}(H)\in\mathscr{S}$.

Proof. Immediate from Lemma 4.1 and Theorem 5.2. □

Grammars considered in classical formal language theory (for example context-free grammars) do not rewrite terminal symbols. A straightforward simulation of such grammars by *s*-grammars yields productions of the form (a, a) for each terminal symbol a. Productions of this form yield "total desynchronization". *s*-grammars containing such productions will be considered now.

Definition 5.2. An *s*-grammar G is *totally desynchronized* if for all terminal symbols of $\mathrm{Base}(G)$, $(a, a)\in\mathrm{Prod}(G)$ and $\bar{a}\in\mathrm{alph}(\mathrm{La}(\mathrm{Sel}(G)))$. □

Theorem 5.3. *Let* $\mathscr{K}$ *be a family of selectors that is closed under union with monoids and under bar-invariant letter-to-letters substitution. Then for every universal s-grammar* G *with* $\mathrm{Sel}(G)\in\mathscr{K}$ *there is an equivalent universal totally desynchronized s-grammar* H *with* $\mathrm{Sel}(H)\in\mathscr{K}$.

Proof. Let $G'=\langle\Sigma, h, S, \Delta, K\rangle$ be a full shadow of G. Let $\Phi=\mathrm{Al}(K)$ and $\Theta=\mathrm{Term}(K)$. ($\Sigma\subset\Phi$ because G is universal). Note that $K\in\mathscr{K}$. Let Γ be an alphabet disjoint from Σ with $\#\Gamma=\#\Delta$. Let φ in $HOM(\tilde{\Phi}, (\tilde{\Phi}-\tilde{\Delta})\cup\tilde{\Gamma})$ be a bar-invariant injective coding that is the identity on $\tilde{\Phi}-\tilde{\Delta}$.

Let H be the *s*-grammar $\langle\Sigma\cup\Gamma, h', S, \Delta, K'\rangle$, where

$$K'=\langle\varphi(\Phi), (\mathrm{La}(K))\cup\Gamma^*\cup(\Sigma\cup\Gamma\cup\bar{\Delta})^*, \mathrm{Term}(K)\rangle$$

and h' is defined by

$$h'(a)=h(\varphi(a)) \quad \text{for} \quad a\in\Sigma-\Delta \quad \text{and}$$

$$h'(\varphi(a))=h'(a)=a \quad \text{for} \quad a\in\Delta.$$

It is straightforward to see that $\mathbf{L}(H)=\mathbf{L}(G)$ and that H is totally desynchronized. Since $K\in\mathscr{K}$ it follows that $K'\in\mathscr{K}$. Moreover, K' is universal and hence the theorem follows. □

REMARK. It is well-known (see, e.g., [RS]) that a totally desynchronized normal form exists for the family of ETOL systems but it cannot exist for the family of EOL systems. Theorem 5.3 together with Corollary 3.1 allows one to see this well-known fact in a more general perspective. □

6. Chain-freeness

In this section we will investigate the possibilities of obtaining chain-free normal forms for *s*-grammars. Our first method for obtaining chain-free normal forms preserves also the propagating property.

Theorem 6.1. *Let* $\mathscr{S}$ *be a* bf *selector scheme and* G *an s-grammar with*

Sel $(G)\in\mathscr{S}$. *Then there is an equivalent chain-free s-grammar* H *with* Sel $(H)\in\mathscr{S}$. *Moreover, if* G *is propagating then so is* H.

Proof. Let $\hat{G}$ be a full shadow of G. By Theorem 4.1 and Corollary 4.1, Sel $(\hat{G})\in\mathscr{S}$ and $L(\hat{G})=L(G)$. First we consider the symbols of Total $(\hat{G})-\mathrm{Al}(\mathrm{Sel}(\hat{G}))$. Without loss of generality we may assume that for every non-terminal symbol A of $\hat{G}$ that is not in Al (Sel $(\hat{G})$), (A, F) is the only production in $\hat{G}$ in which A appears. Note that the failure symbol F is not versatile in $\hat{G}$ and thus productions of the form (A, F) are not chains. For every terminal symbol c in $\hat{G}$ that is not in Al (Sel $(\hat{G})$), we remove all productions with c as a lefthand side and add one production (c, c). Hence, in the resulting grammar G', c is not versatile anymore and thus productions of the form (b, c) are not chains. Clearly $\hat{G}$ and G' are equivalent. So $L(G')=L(G)$. Moreover, for every chain (b, c) in G', both b and c are in Al (Sel (G')).

The following algorithm yields a chain-free s-grammar H.

— Let P be an initially empty set of chains and let H' be G' initially.

— If $H'=\langle\Sigma, h, S, \Delta, K\rangle$ is chain-free then let $H=H'$.

— Otherwise let (b, c) be a chain in H', $b\neq S$, and let $h_1\in FSUB(\Sigma, \Sigma)$ be defined by

$$h_1(b) = h(b)-\{c\} \quad \text{and}$$

$$h_1(a) = h(a) \quad \text{for} \quad a\in\Sigma-\{b\}.$$

We add (b, c) to P. Let $h'\in FSUB(\Sigma, \Sigma)$ be defined by

$$h'(a) = (h_1(a)\cup\{F\}\cup$$

$$\cup\{u_0d_1\ldots u_{n-1}d_nu_n\colon d_1, \ldots, d_n\in\{b, c\},\ u_0, \ldots, u_n\in(\Sigma-\{b\})^*$$

$$\text{and} \quad u_0bu_1b\ldots u_{n-1}bu_n\in h(a)\})-\{d\colon (a, d)\in P\} \quad \text{for all} \quad a\in\Sigma.$$

We then iterate this step for $H'=\langle\Sigma, h', S, \Delta, K\rangle$.

Clearly, this procedure terminates and produces a chain-free grammar. Thus it suffices to show that each iteration in the above algorithm preserves the generated language.

Let $H'=\langle\Sigma, h, S, \Delta, K\rangle$ be an s-grammar as above and let H_1 be the s-grammar obtained from H' by eliminating the chain (b, c). If a derivation in H' does not rewrite b into c then this is obviously also a derivation in H_1. Let thus $(a, u_1bu_1')\in$ $\in\mathrm{Prod}(H')$ for some $a\in\Sigma$ and $u, u'\in\Sigma^*$. We look at the trace of a derivation in H' that contains this production and that rewrites the occurrence of b thus obtained into c at some further step,

$$(S, \ldots, w_1av_1, w_2u_1bu_1'v_2, \ldots, w_3u_2bu_2'v_3, w_4u_3cu_3'v_4).$$

K is csf and abf, and $c\in\mathrm{Al}(K)$. Thus for every $wbw'\in\mathrm{La}(K)$ and $w\bar{b}w'\in\mathrm{La}(K)$, $wcw'\in\mathrm{La}(K)$. It follows that there is a derivation in H_1 with a trace of the form $(S, \ldots, x_1ay_1, x_2z_1cz_1'y_2, \ldots, x_3z_2cz_2'y_3, x_4z_3cz_3'y_4)$, where every x_i, y_i, z_i and z_i' can be obtained from w_i, v_i, u_i and u_i', respectively, by replacing some occurrences of b by c.

Note that b is always a non-terminal symbol since we construct a full shadow

first. Hence, if $w_4u_3cu_3'v_4$ is a word over the terminal alphabet, it equals $x_4z_3cz_3'y_4$. Thus $\mathbf{L}(H')\subset\mathbf{L}(H_1)$.

The opposite inclusion can be shown similarly because K is sf and bf and $b\in\mathrm{Al}\,(K)$ (thus on one hand, $wcw'\in\mathrm{La}\,(K)$ implies that $wbw'\in\mathrm{La}\,(K)$, $w\bar{b}w'\in\mathrm{La}(K)$ and $\mathrm{iden}\,(w)\bar{b}\,\mathrm{iden}\,(w')\in\mathrm{La}\,(K)$, and on the other hand, $w\bar{c}w'\in\mathrm{La}\,(K)$ implies that $\mathrm{iden}\,(w)\bar{b}\,\mathrm{iden}\,(w')\in\mathrm{La}(K)$).

Therefore $\mathbf{L}(H_1)=\mathbf{L}(H')$ and hence $\mathbf{L}(H)=\mathbf{L}(G)$. □

REMARK. Note that the conditions of Theorem 6.1 do not lead to strong limits on the language-generating power. In [KR] it was shown that bf and sf s-grammars generate arbitrary length sets. □

The following example illustrates the method from the proof of the above theorem.

Example 6.1. Let G be the s-grammar $\langle\Sigma, h, S, \Delta, K\rangle$, where

$$\Sigma = \{S, A, B, C, a, b, c, F\},$$

$$\Delta = \{a, b, c\},$$

$$K = \langle\Sigma, L, \Delta\rangle, \quad \text{where} \quad L = \bar{\Sigma}\cup\bigcup_{n\geq 1}(\bar{\Sigma}^{13n}\cup\bar{\Sigma}^{13n+5}),$$

and h is defined by

$$h(S) = a^5Aa^6B,$$

$$h(A) = \{a^7A, b^5A, a^2\},$$

$$h(B) = \{a^6B, b^8B, C, a^2\},$$

$$h(C) = \{c^8b, c^8a^2\} \quad \text{and}$$

$$h(a) = h(b) = h(c) = h(F) = \{F\}.$$

Clearly G is bf and sf, but not chain-free.

Let H be the s-grammar $(\Sigma, h', S, \Delta, K)$, where h' is defined by

$$h'(d) = h(d)\cup\{F\}, \quad \text{for} \quad d\in\{A, a, b, c, F\},$$

$$h'(S) = \{a^5Aa^6B, a^5Aa^6c, F\},$$

$$h'(B) = \{a^6B, a^6C, b^8B, b^8C, a^2, F\} \quad \text{and}$$

$$h'(C) = \{c^8B, c^8C, c^8a^2, F\}.$$

H is chain-free, $\mathrm{Sel}\,(H)=\mathrm{Sel}\,(G)$ (because G was synchronized there was no need to change the selector) and

$$L(H) = L(G) = \{a^5\varphi(w)a^2a^6\psi(w)a^2\colon\ w\in\{0, 1\}^*\},$$

where $\varphi\in HOM\,(\{0, 1\}, \{a, b\})$ and $\psi\in FSUB\,(\{0, 1\}, \{a, b, c\})$ are defined by

$$\varphi(0) = a^7, \quad \varphi(1) = b^5,$$

$$\psi(0) = a^6, \quad \psi(1) = \{b^8, c^8\}. \quad \square$$

REMARK. There are ways to specify conditions on a general selector family $\mathscr{K}$ that allow the classical "context-free style" chain elimination for an s-grammar G with $\mathrm{Sel}(G)\in\mathscr{K}$. The conditions known to us do, however, involve intersection of the selector language with the set of sentential forms generated by G, in the following sense. Say that we are given a chain (b, c). We want to be able to introduce c in the new grammar, say H, wherever b could occur in a derivation of G. The problem arises from the fact that there may be a word in $\mathrm{Sel}(G)$ of the form wcw' that was not applicable in G (because no such sentential form existed there) but it becomes applicable in H (which derives iden (wcw') from its start symbol). To eliminate such a case we have to get rid of all the words w in $\mathrm{Sel}(G)$ for which iden (w) is not a sentential form of G. This does, however, complicate the structure of a selector to such an extent as to make the result "useless". For this reason we do not present an analog of Theorem 6.1 for the case when $\mathscr{S}$ is a selector family other than a selector scheme. □

In the rest of this section we consider methods for achieving chain-free normal forms that make use of erasing productions.

Theorem 6.2. *Let K be a selector, such that $T\in\mathrm{Al}(K)-\mathrm{Term}(K)$ and $\Delta\subset\subset\mathrm{Term}(K)$. If K is $\{T,\overline{T}\}$-e and $\{\overline{T}\}$-i and if $\Delta^*\subseteq\mathrm{La}(K)$, then for every s-grammar $G=\langle\Sigma, h, S, \Delta, K\rangle$, where $T\notin\Sigma$, there is an equivalent chain-free s-grammar H with $\mathrm{Sel}(H)=K$.*

Proof. Let $H=\langle\Sigma\cup T, h', S, \Delta, K\rangle$, where h' is defined by
$h'(a)=(h(a)-\Sigma)\cup\{bT\colon b\in h(a)\cap\Sigma\}$ for all $a\in\Sigma$ and
$h'(T)=\lambda$.
Since H is chain-free it remains to show that $\mathbf{L}(H)=\mathbf{L}(G)$.

Let $S\overset{i}{\underset{G}{\Rightarrow}}w$, $w\in\Delta^*$. Then $S\overset{i}{\underset{H}{\Rightarrow}}v$ for some v with $\mathrm{erase}_{\{T\}}(v)=w$ because K is $\{\overline{T}\}$-i. But $v\underset{H}{\Rightarrow}w$ because $(\Delta\cup\{\overline{T}\})^*\subseteq\mathrm{La}(K)$. Hence $\mathbf{L}(G)\subset\mathbf{L}(H)$.

Vice versa it can easily be seen by induction on i that if $S\overset{i}{\underset{H}{\Rightarrow}}x$ then $S\overset{*}{\underset{G}{\Rightarrow}}\mathrm{erase}_{\{T\}}(x)$ because K is $\{T,\overline{T}\}$-e. Therefore $\mathbf{L}(H)\subset\mathbf{L}(G)$ and hence the equality holds.

Note that T-e is necessary here, because there may be a word w in $\mathrm{La}(K)$, such that $v=\mathrm{erase}_T(w)\notin\mathrm{La}(K)$. v is blocked in G but w is unblocked in H and, hence, additional words may be generated in H. □

Theorem 6.3. *Let $\mathscr{K}$ be a family of selectors that is closed under union with monoids. Then for each s-grammar G with $\mathrm{Sel}(G)\in\mathscr{K}$ there is an equivalent chain-free s-grammar H with $\mathrm{Sel}(H)\in\mathscr{K}$.*

Proof. Let $G=\langle\Sigma, h, S, \Delta, K\rangle$ and let T be a new symbol. Let $\mathrm{Base}(H)$ be as in Theorem 6.2 and let

$$\mathrm{Sel}(H)=\langle\mathrm{Al}(K)\cup\{T\},\ \mathrm{La}(K)\cup(\Sigma\cup\overline{\{T\}})^*,\ \mathrm{Term}(K)\rangle.$$

The equivalence of G and H can easily be seen by observing that every derivation D of length 1 in G is simulated by a derivation D' of length 1 or 2 in H; the first step of D' rewrites like D, maybe introducing some occurrences of T. If any T's were introduced the second step of D' erases them. □

Corollary 6.1. For *every* ETOL *system there is an equivalent chain-free* ETOL *system.*

Proof. Immediate from Theorem 6.3 and Corollary 3.1. □

We define now an operation that is based on the well-known shuffle operation (see, e.g., [HU] and [RS]) that allows us to specify conditions for achieving chain-free normal forms (also applicable to the EOL case).

Definition 6.1. Let K and L be languages over Δ and Θ, respectively. The *full shuffle of* K *and* L, denoted by $K \between L$, is defined by

$$K \between L = K \cup L \cup$$

$$\{x_1 y_1 x_2 y_2 \ldots x_n y_n : n \geq 1,\ x_1, \ldots, x_n \in \Delta^*,$$

$$y_1, \ldots, y_n \in \Theta^*,\ x_1 \ldots x_n \in K \text{ and } y_1 \ldots y_n \in L\}. \quad \square$$

Theorem 6.4. *Let* $\mathscr{K}$ *be a family of selectors that is closed under bar-preserving letter-to-letters substitution and full shuffle with monoids. Then for each s-grammar* G *with* $\mathrm{Sel}(G) \in \mathscr{K}$ *there is an equivalent chain-free s-grammar* H *with* $\mathrm{Sel}(H) \in \mathscr{K}$.

Proof. Let $G' = (\Sigma, h, S, \Delta, K)$ be a shadow of G and let T be a new symbol. Let H be the s-grammar $\langle \Sigma \cup \{T\}, h', S, \Delta, K' \rangle$, where

$$K' = \langle \mathrm{Al}(K) \cup \{T\},\ \mathrm{La}(K) \between \overline{T}^*,\ \mathrm{Term}(K) \rangle$$

and h' is defined by

$$h'(a) = \big(h(a) - (\Sigma - \Delta)\big) \cup \{bT :\ b \in h(a) \cap (\Sigma - \Delta)\}, \quad \text{for } a \in \Sigma - \Delta,$$

$$h'(a) = a, \quad \text{for } a \in \Delta, \quad \text{and}$$

$$h'(T) = \lambda.$$

The equivalence of H and G', and hence of H and G, follows from the observation that for all $a \in \Delta$, $\bar{a} \notin \mathrm{alph}(\mathrm{La}(K))$ and hence replacing the productions for those a by identity does not change the generated language. (As a result the terminals are not versatile and productions of the form (A, a), $a \in \Delta$, are not chains. This implies the chain-freeness of H.)

Since the symbol T is only introduced together with a non-terminal symbol it follows that (in a successful derivation) it can be erased in a next derivation step. Moreover $K' \in \mathscr{K}$ and the theorem follows. □

Corollary 6.2. *Let* $\mathscr{K}$ *be a family of selectors that is closed under bar-preserving letter-to-letters substitution and inverse weak identity, such that for each* $K \in \mathscr{K}$, $\lambda \in \mathrm{La}(K)$. *Then for each s-grammar* G *with* $\mathrm{Sel}(G) \in \mathscr{K}$ *there is an equivalent chain-free s-grammar* H *with* $\mathrm{Sel}(H) \in \mathscr{K}$.

Proof, The corollary is immediate from Theorem 6.4 because for all languages L with $\lambda \in L$ and for all monoids Θ^* such that $\Theta \cap \mathrm{alph}(L) = \varnothing$

$$L \between \Theta^* = \mathrm{erase}_{\Theta}^{-1}(L). \quad \square$$

Corollary 6.3. *For every* EOL *system there is an equivalent chain-free* EOL *system.*

Proof. This follows immediately from Theorem 6.4 and Corollary 3.1. □

7. Removing λ-productions

In this section we will investigate the possibilities of removing erasing productions. As a first step towards this goal we introduce and investigate a normal form for s-grammars in which the "erasing", and "terminal-generating" roles of symbols are separated; i.e. if a symbol derives λ (in the base of the grammar) then it cannot derive (in the base) a word containing any terminal symbols.

Definition 7.1. Let $G=\langle \Sigma, h, S, \Delta, K\rangle$ be an s-grammar. G is in λ *normal form* (λNF) if for all $a\in\Sigma-\{S\}$ with $a \overset{+}{\underset{\text{Base}(G)}{\Longrightarrow}} \lambda$, $a \overset{+}{\underset{\text{Base}(G)}{\Longrightarrow}} w$ implies that $w\in \in(\Sigma-\Delta)^*$.

The set $\{a\in\Sigma\colon a \overset{*}{\underset{\text{Base}(G)}{\Longrightarrow}} \lambda\}$ is then called the λ-*alphabet* of G and is denoted by Lamal (G). □

REMARK. From the constructions in Theorems 2.1 through 2.6 it can be seen that λNF is indeed a normal form for s-grammars. □

Theorem 7.1. *Let $\mathscr{K}$ be a family of selectors that is closed under bar-invariant letter-to-letters substitution. Then for every s-grammar G with* Sel $(G)\in\mathscr{K}$ *there is an equivalent s-grammar H in* λNF *with* Sel $(H)\in\mathscr{K}$.

Proof. Let $G=\langle \Sigma, h, S, \Delta, K\rangle$.

Let $\Sigma'=\{a\in\Sigma-\{S\}\colon a \overset{*}{\underset{\text{Base}(G)}{\Longrightarrow}} \lambda\}$ and $\Sigma_\lambda=\{a^\lambda\colon a\in\Sigma'\}$, such that Σ_λ is an alphabet of new symbols.

Let φ be the injective coding in $HOM(\Sigma', \Sigma_\lambda)$ defined by $\varphi(a)=a^\lambda$.

Let $\psi\in FSUB(\widetilde{\text{Total}(G)}, \widetilde{\text{Total}(G)}\cup\tilde{\Sigma}_\lambda)$ be defined by

$$\psi(a) = \{a, a^\lambda\} \quad \text{for} \quad a\in\Sigma'.$$

$$\psi(a) = a \quad \text{for} \quad a\in(\Sigma-\Sigma')\cup(\text{Al}\,(K)-\Sigma) \quad \text{and}$$

$$\psi(\bar{a}) = \overline{\psi(a)} \quad \text{for} \quad a\in\text{Total}\,(G).$$

Let H be the s-grammar $\langle \Sigma\cup\Sigma_\lambda, h', S, \Delta, K'\rangle$, where

$$K' = \langle \psi(\text{Al}\,(K)),\ \psi(\text{La}\,(K)),\ \text{Term}\,(K)\rangle$$

and h' is defined by

$$h'(a^\lambda) = \varphi(h(a)\cap\Sigma'^*)\cup\{F\} \quad \text{for all} \quad a\in\Sigma'.$$

$$h'(a) = (\psi(h(a))\cap(\Sigma\cup\Sigma_\lambda)^*\Sigma(\Sigma\cup\Sigma_\lambda)^*)\cup\{F\} \quad \text{for all} \quad a\in\Sigma-\{S\} \quad \text{and}$$

$$h'(S) = \psi(h(S)).$$

It is easy to see that $\mathbf{L}(H)=\mathbf{L}(G)$ and that H is in λNF. Moreover, K' is in $\mathscr{K}$ and hence the theorem holds. □

Corollary 7.1. *Let $n\geqq 1$. For every n-continuous grammar there is an equivalent n-continuous grammar in* λNF.

Proof. Immediate from the definition of continuous grammars and Theorem 7.1. □

Corollary 7.2. *For every* ETOL *system there is an equivalent* ETOL *system in* λNF.

Proof. Immediate from Corollary 3.1 and Theorem 7.1. □

Corollary 7.3. *Let* $\mathscr{S}$ *be a selector scheme. Then for every s-grammar* G *with* $\mathrm{Sel}(G)\in\mathscr{S}$ *there is an equivalent s-grammar* H *in* λNF *with* $\mathrm{Sel}(H)\in\mathscr{S}$.

Proof. Immediate by Theorem 7.1 and Lemma 4.1. □

Corollary 7.4. *For every* EOL *system there is an equivalent* EOL *system in* λNF.

Proof. Immediate from Corollaries 7.3 and 3.1. □

We will now investigate the possibilities of obtaining propagating s-grammars from s-grammars in λNF.

Theorem 7.2. *Let* G *be an s-grammar in* λNF *that is* $\overline{\mathrm{Lamal}(G)}$-i *and* $\overline{\mathrm{Lamal}(G)}$-e. *Then there is an equivalent propagating s-grammar* H *with* $\mathrm{Sel}(H)=\mathrm{Sel}(G)$.

Proof. Let $G=\langle\Sigma, h, S, \Delta, K\rangle$ and let $\Sigma_\lambda=\mathrm{Lamal}(G)$.
Let H be the s-grammar $\langle\Sigma-\Sigma_\lambda, h', S, \Delta, K\rangle$ where h' is defined by

$$h'(a)=\mathrm{erase}_{\Sigma_\lambda}(h(a)) \quad \text{for all} \quad a\in\Sigma-\Sigma_\lambda.$$

Obviously, H is propagating. It remains thus to show that G and H are equivalent. We shall show by induction on i that there is a monotonously increasing sequence of integers $j_0, j_1, \ldots$ such that if $S\overset{i}{\underset{G}{\Rightarrow}}u$ for some word $u\in\Sigma^*$ then $S\overset{j_i}{\underset{H}{\Rightarrow}}\mathrm{erase}_{\Sigma_\lambda}(u)$.

BASIS. Let $i=0$. Then $j_i=0$.

INDUCTION. Let the induction hypothesis hold for i.

Let $S\overset{i}{\underset{G}{\Rightarrow}}x\underset{G}{\Rightarrow}y$. By induction $S\overset{j_i}{\underset{H}{\Rightarrow}}\mathrm{erase}_{\Sigma_\lambda}(x)$. If $\mathrm{erase}_{\Sigma_\lambda}(y)=\mathrm{erase}_{\Sigma_\lambda}(x)$ then the hypothesis is also true for $i+1$ by letting $j_{i+1}=j_i$.

Otherwise there must be a word $z\in\mathrm{La}(K)$ with $\mathrm{iden}(z)=x$, such that (z) is the barred trace of a derivation of length 1 of y from x. Since K is $\bar{\Sigma}_\lambda$-e, it follows that $\mathrm{erase}_{\Sigma_\lambda}(z)\in\mathrm{La}(K)$. Therefore $\mathrm{erase}_{\Sigma_\lambda}(x)\underset{H}{\Rightarrow}\mathrm{erase}_{\Sigma_\lambda}(y)$, which proves the induction hypothesis for $i+1$, letting $j_{i+1}=j_i+1$.

Thus $\mathbf{L}(G)\subset\mathbf{L}(H)$.

The converse inclusion can easily be proved in a similar manner, showing that there is a monotonously increasing sequence of integers $j_0, j_1, \ldots$ such that if $S\overset{i}{\underset{H}{\Rightarrow}}x$ for some word $x\in\Sigma^*$ then there is a word $z\in\mathrm{erase}^{-1}_{\Sigma_\lambda}(x)$ such that $S\overset{j_i}{\underset{G}{\Rightarrow}}z$.

Hence $\mathbf{L}(H)=\mathbf{L}(G)$ and the theorem follows. □

8. Productions in binary form

In this section we will investigate the possibilities for s-grammars of obtaining equivalent binary s-grammars.

Let Σ and Θ be alphabets. A mapping from $\bar{\Sigma}^*$ into $\bar{\Theta}^*$ is called a *barring* mapping.

Theorem 8.1. *Let $\mathscr{K}$ be a family that is closed under inverse weak identity and barring letter-to-letters substitution. Then for every s-grammar G with* Sel $(G)\in\mathscr{K}$ *there is an equivalent binary s-grammar H with* Sel $(H)\in\mathscr{K}$.

Proof. If G is already binary, then the statement of the theorem follows for $H=G$. Let thus Maxr $(G)\geqq 3$. Let $G=\langle\Sigma, h, S, \Delta, K\rangle$ and let $l=\#\,\text{Prod}\,(G)$. Let $(a_j, b_{j,n_j} \ldots b_{j,1})$, $1\leqq j\leqq l$, be the productions in Prod (G), in some arbitrary order, where $a_j, b_{j,1}, \ldots, b_{j,n_j}\in\Sigma$ for $1\leqq j\leqq l$.

A derivation D of length 1 in G will be simulated by a derivation D' in H of length Maxr $(G)-1$ as follows:

— Every symbol a that is not rewritten in D occurs as $\langle a, 2\rangle$ and every symbol a that is rewritten in D occurs as itself in the (left hand side of the) first word of D'.

— Every symbol in the i'th word of Trace (D'), $1\leqq i<$Maxr $(G)-2$, is either a tuple of the form $\langle a,$ Maxr $(G)-i\rangle$ or $[j,$ Maxr $(G)-i]$ or $(a,$ Maxr $(G)-i)$. The tuples within angled brackets represent symbols that are not rewritten in D. If a symbol a is rewritten using the j'th production in Prod (G) (i.e. $(a, b_{j,n}\ldots b_{j,1})$), it is first rewritten in H into $[j,$ Maxr $(G)]$ (or into $(b_{j,n_j},$ Maxr $(G))[j,$ Maxr $(G)]$ if $n_j=$Maxr (G)). The tuples $[j, m]$ within square brackets keep track of the j'th original production by deriving a pair of tuples $(b_{j,m-1}, m-1)[j, m-1]$ for $m\leqq$ $\leqq n_j+1$ and by "counting down" to $[j, m-1]$ if $m>n_j+1$. The tuples within parentheses keep on "counting".

— Finally, every tuple of the form $\langle a, 3\rangle$ or $(a, 3)$ is rewritten into a or $\langle a, 2\rangle$, and every tuple of the form $[j, 3]$ is rewritten into $c_{j,2}c_{j,1}$, where $c_{j,i}$ is either $b_{j,i}$ or $\langle b_{j,i}, 2\rangle$.

Formally, let

$$\Phi = \{(a, i)\colon\ a\in\Sigma \quad\text{and}\quad 3\leqq i\leqq \text{Maxr}\,(G)\} \quad\text{and}$$

$$\Theta = \{\langle a, i\rangle\colon\ a\in\Sigma \quad\text{and}\quad 2\leqq i\leqq \text{Maxr}\,(G)\}\cup$$

$$\cup\,\{[k, i]\colon\ 1\leqq k\leqq l \quad\text{and}\quad 3\leqq i\leqq \text{Maxr}\,(G)\}$$

such that Φ, Θ and Total (G) are pairwise disjoint.

Let φ be the barring letter-to-letters substitution in $FSUB\,(\overline{\text{Total}\,(K)}, \overline{\text{Total}\,(K)}\cup\bar{\Theta})$ defined by

$$\varphi(a) = \{\overline{\langle a, j\rangle}\colon\ 2\leqq j\leqq \text{Maxr}\,(G)\} \quad\text{for all}\quad a\in\Sigma,$$

$$\varphi(\bar{a}) = \bar{a}\cup\{\overline{[j, i]}\colon\ a_j = a \quad\text{and}\quad 3\leqq i\leqq \text{Maxr}\,(G)\} \quad\text{for all}\quad a\in\Sigma,$$

$$\varphi(a) = \varphi(\bar{a}) = \bar{a} \quad\text{for all}\quad a\in\text{Al}\,(K)-\Sigma.$$

Let H be the s-grammar $\langle \Sigma \cup \Phi \cup \Theta, h', S, \Delta, K' \rangle$, where

$$K' = \langle \Phi \cup \mathrm{iden}(\varphi(\mathrm{Al}(K))), \mathrm{erase}_{\Phi}^{-1}(\varphi(\mathrm{La}(K))), \mathrm{Term}(K) \rangle$$

and h' is defined by

$$h'(a) = \{[j, \mathrm{Maxr}(G)] : a_j = a \text{ and } n_j < \mathrm{Maxr}(G)\} \cup$$
$$\cup \{(b_{j,n_j}, \mathrm{Maxr}(G))[j, \mathrm{Maxr}(G)] : a_j = a \text{ and } n_j = \mathrm{Maxr}(G)\}$$
$$\text{for all } a \in \Sigma,$$

$$h'([j, m+1]) = \begin{cases} (b_{j,m}, m)[j, m] & \text{if } m \leqq n_j \\ [j, m] & \text{otherwise} \end{cases}$$
$$\text{for all } 3 \leqq m \leqq \mathrm{Maxr}(G)-1 \text{ and } 1 \leqq j \leqq l,$$

$$h'([j, 3]) = \begin{cases} \{c_2 c_1 : c_i \in \{b_{j,i}, \langle b_{j,i}, 2 \rangle\} \text{ for } i \in \{1, 2\}\} & \text{if } n_j \geqq 2 \\ \{b_{j,1}, \langle b_{j,1}, 2 \rangle\} & \text{if } n_j = 1. \\ \lambda & \text{otherwise} \end{cases}$$
$$\text{for all } 1 \leqq j \leqq l,$$

$$h'((a, i+1)) = (a, i) \text{ for all } a \in \Sigma \text{ and } 3 \leqq i \leqq \mathrm{Maxr}(G)-1,$$

$$h'(\langle a, i+1 \rangle) = \langle a, i \rangle \text{ for all } a \in \Sigma \text{ and } 3 \leqq i \leqq \mathrm{Maxr}(G)-1,$$

$$h'(\langle a, 3 \rangle) = h'((a, 3)) = \{a, \langle a, 2 \rangle\} \text{ for all } a \in \Sigma \text{ and}$$

$$h'(\langle a, 2 \rangle) = \langle a, \mathrm{Maxr}(G) \rangle \text{ for all } a \in \Sigma.$$

Since H is binary and $K' \in \mathscr{K}$ whenever $K \in \mathscr{K}$, it remains to show that $\mathbf{L}(H) = \mathbf{L}(G)$.

Let $\psi \in FSUB(\Sigma, \Sigma \cup \Theta)$ be defined by

$$\psi(a) = \{a, \langle a, 2 \rangle\} \text{ for all } a \in \Sigma$$

and let χ be the injective coding in $HOM(\bar{\Sigma}, \Sigma \cup \Theta)$ defined by

$$\chi(a) = \langle a, 2 \rangle \text{ and}$$
$$\chi(\bar{a}) = a.$$

It can now easily be seen from the construction of H that every derivation D of length 1 in G can be simulated in H. Let $\mathrm{Trace}(D) = (x, y)$ and $\mathrm{Btrace}(D) = (z)$. Then $\chi(z) \overset{\mathrm{Maxr}(G)-1}{\underset{H}{\Longrightarrow}} u$ for all $u \in \psi(y)$.

Thus, $\mathbf{L}(G) \subset \mathbf{L}(H)$.

The converse inclusion follows from the fact that tuples (i.e. symbols from $\Theta \cup \Phi$) can only occur together in a sentential form of H if they have identical second components. Moreover, if the second component of the tuples is not 2, then this sentential form consists exclusively of tuples; otherwise, symbols from Σ can occur together with symbols of the form $\langle a, 2 \rangle$ where $a \in \Sigma$. Hence, successful derivations in H simulate successful derivations in G as described above. Therefore, $\mathbf{L}(H) \subset \mathbf{L}(G)$, and thus we have that $\mathbf{L}(H) = \mathbf{L}(G)$. This completes the proof of the theorem. □

REMARK. A similar construction can be performed in which every derivation of length 1 in G is simulated by a derivation of length $\lceil \log_2 \mathrm{Maxr}(G) \rceil$ in H using a "balanced" decomposition of the original productions. □

Corollary 8.1. *For every* EOL *system there is an equivalent binary* EOL *system.*

Proof. Immediate from Theorem 8.1 and Corollary 3.1. □

Theorem 8.2. *Let* $\mathscr{K}$ *be a family of selectors that is closed under union with monoids. Then for every s-grammar* G *with* $\mathrm{Sel}(G) \in \mathscr{K}$ *there is an equivalent binary s-grammar* H *with* $\mathrm{Sel}(H) \in \mathscr{K}$.

Proof. If G is already binary, then the statement of the theorem follows for $H = G$. Let thus $\mathrm{Maxr}(G) \geqq 3$. Let $G = \langle \Sigma, h, S, \Delta, K \rangle$ and let $l = \# \mathrm{Prod}(G)$. Let $(a_j, b_{j,n_j}, \ldots, b_{j,1})$ $1 \leqq j \leqq l$, be the elements of $\mathrm{Prod}(G)$ in some arbitrary enumeration, such that $a_j, b_{j,1}, \ldots, b_{j,n_j} \in \Sigma$ for $1 \leqq j \leqq l$.

Let $\Theta = \{[j, i]: 1 \leqq j \leqq l \text{ and } 3 \leqq i \leqq n_j\}$ such that Θ and $\mathrm{Total}(G)$ are pairwise disjoint.

Let H be the s-grammar $\langle \Sigma \cup \Theta, h', S, \Delta, K' \rangle$, where

$$K' = \langle \mathrm{Al}(K) \cup \Sigma \cup \Theta,\ \mathrm{La}(K) \cup (\bar{\Theta} \cup \Sigma)^*,\ \mathrm{Term}(K) \rangle$$

and h' is defined by

$$h'(a) = \big(h(a) - \Sigma\Sigma\Sigma^+\big) \cup \{b_{j,n_j}[j, n_j]: a = a_j \text{ and } n_j \geqq 3\} \text{ for all } a \in \Sigma,$$

$$h'([j, m+1]) = b_{j,m}[j, m] \text{ for all } 1 \leqq j \leqq l \text{ and } 3 \leqq m \leqq n_j \text{ and}$$

$$h([j, 3]) = b_{j,2} b_{j,1} \text{ for all } 1 \leqq j \leqq l \text{ with } n_j \geqq 3.$$

Obviously, $K \in \mathscr{K}$. It can now easily be verified that, for all $x, y \in \Sigma^*$, $x \underset{G}{\Rightarrow} y$ if and only if $x \overset{i}{\underset{H}{\Rightarrow}} y$ for some $1 \leqq i \leqq \mathrm{Maxr}(G) - 1$. Hence, $\mathbf{L}(H) = \mathbf{L}(G)$. Moreover, H is binary and thus the theorem holds. □

Corollary 8.2. *For every* ETOL *system there is an equivalent binary* ETOL *system.*

Proof. Immediate from Theorem 8.2 and Corollary 3.1. □

Note that the constructions presented in Theorems 8.1 and 8.2 are of a basically different nature. The former is of a "parallel" nature, while the second one is of a "sequential" nature.

REMARK. To apply the construction of Theorem 8.1 to the case of a selector scheme $\mathscr{S}$, we note that the construction requires all symbols in the resulting selector to be barred. Thus, for all $K \in \mathscr{S}$, $\mathrm{La}(K) \subset \overline{\mathrm{Al}(K)}^*$.

The construction requires that every $K \in \mathscr{S}$ must be $\overline{\mathrm{Al}(K)}$-interspersed and, in order not to add new words to the language by unblocking, $\overline{\mathrm{Al}(K)}$-erasing. These conditions imply, however, that $K = \overline{\mathrm{Al}(K)}^*$, which restricts the family of generated languages to EOL languages (see Corollary 3.1).

A similar argument can be used for the construction of Theorem 8.2. □

9. Removing right recursion

In this section we will investigate the possibilities of obtaining non-right-recursive normal forms for s-grammars. First we will consider the introduction of erasing productions as a method to eliminate right recursion.

Theorem 9.1. *Let $\mathscr{K}$ be a family of selectors that is closed under inverse weak identity, bar-preserving letter-to-letters substitution and union with monoids. Then for each s-grammar G with $\mathrm{Sel}(G)\in\mathscr{K}$ there is an equivalent s-grammar H with $\mathrm{Sel}(H)\in\mathscr{K}$ that is not right-recursive.*

Proof. Let $G'=\langle\Sigma, h, S, \Delta, K\rangle$ be a shadow of G and let T be a new symbol. K is in $\mathscr{K}$ by Theorem 4.4.

Let $\varphi\in HOM(\Sigma, \Sigma\cup T)$ be defined by

$$\varphi(a) = a \quad \text{for} \quad a\in\Delta\cup\{S\},$$

$$\varphi(a) = aT \quad \text{for} \quad a\in\Sigma-(\Delta\cup\{S\}).$$

Let H be the s-grammar $\langle\Sigma\cup\{T\}, \varphi\circ h, S, \Delta, K'\rangle$, where

$$K' = \langle \mathrm{Al}(K)\cup\{T\}, \mathrm{erase}_T^{-1}(\mathrm{La}(K))\cup(\Delta\cup\overline{T})^*, \mathrm{Term}(K)\rangle.$$

It is easy to see that H is equivalent to G and that H is not right-recursive. Moreover, $K'\in\mathscr{K}$. Hence the theorem holds. □

REMARK. The conditions that are necessary to apply the construction of Theorem 9.1 to selector schemes yield a trivial result; each selector language would be either of the form $\overline{\Sigma}^*$ or of the form $\overline{\Sigma}^*$ for some alphabet Σ. (For an analogous argument see section 8.) □

The following example illustrates the method from the proof of Theorem 9.1.

Example 9.1. Let G be the right-recursive s-grammar $\langle\Sigma, h, S, \Delta, K\rangle$, where

$$\Sigma = \{S, S', Z, A_1, A_2, B_1, B_2, B_3, F, a, b\},$$

$$\Delta = \{a, b\},$$

$$K = \langle\Sigma, (\overline{\Sigma}-\overline{\Delta})^*, \Delta\rangle$$

and h is defined by

$$h(S) = S',\ h(S') = A_1Z,\ h(Z) = \{B_1S', B_3\},$$

$$h(A_1) = \{A_1A_2, a\},\ h(B_1) = \{B_1B_2, b\},$$

$$h(A_2) = \{A_2, a\},\ h(B_2) = \{B_2, b\},\ h(B_3) = b,$$

$$h(a) = h(b) = F.$$

Let H be the s-grammar $\langle\Sigma\cup\{T\}, h', S, \Delta, K'\rangle$, where

$$K' = \langle\Sigma\cup\{T\}, ((\overline{\Sigma}-\overline{\Delta})\cup T)^*\cup(\Delta\cup\overline{T})^*, \Delta\rangle$$

and h' is defined by

$$h'(S) = S',$$
$$h'(C) = h(C)T \quad \text{for} \quad C \in \{S', Z, A_1, A_2, B_1, B_2, B_3\},$$
$$h'(a) = h'(b) = F \quad \text{and}$$
$$h(T) = \lambda.$$

Clearly $\mathbf{L}(G) = \mathbf{L}(H) = \{a^{2n}b^{2n-1}\ldots a^2b : n \geqq 1\}$. Moreover, H is not right-recursive. □

REMARK. Another method to eliminate right recursion is the classical Greibach Normal Form construction (see, e.g., [S]). We conjecture that there is no non-trivial condition that allows us to apply this construction to s-grammars, because it changes the structure of the grammar in a very severe way; for instance, symbols are shifted from one end of a production to the opposite end, other symbols are eliminated and yield thus changes in the length of derivations.

Formally we base our conjecture on the result that there is no Greibach Normal Form for EPTOL systems, as shown in the next theorem. □

Definition 9.1. (see, e.g. [R]). Let $G = \langle \Sigma, \mathscr{H}, S, \Delta \rangle$ be an ETOL system.

— Let $n \geqq 1$. Let $\mathscr{H}_n = \{h_{i_1} \circ \ldots \circ h_{i_n} : h_i \in \mathscr{H}\}$.

For every $\varphi \in \mathscr{H}_n$, let φ' be the finite substitution in $FSUB(\Sigma, \Sigma)$ defined by

$$\varphi'(S) = \{w \in \Sigma^* : S \overset{j}{\underset{G}{\Rightarrow}} w \quad \text{for} \quad 0 \leqq j \leqq n\} \quad \text{and}$$
$$\varphi'(a) = \varphi(a) \quad \text{for} \quad a \in \Sigma - \{S\}.$$

The *speedup of* G *by* n, denoted by $\text{speed}_n(G)$, is the ETOL system $\langle \Sigma, \{\varphi' : \varphi \in \mathscr{H}_n\}, S, \Delta \rangle$.

— An ETOL system H is a *speedup* of G if there is an integer $n \geqq 1$ such that $H = \text{speed}_n(G)$. □

Note that, for every ETOL system G and for every integer $n \geqq 1$, $\mathbf{L}(G) = \mathbf{L}(\text{speed}_n(G))$.

The following lemma can easily be established using standard techniques from the theory of ETOL systems.

Lemma 9.1. *Let* $G \in \langle \Sigma, \mathscr{H}, S, \Delta \rangle$ *be an* ETOL *system. Then there is an* ETOL *system* H *that is a speedup of* G *such that, for all symbols* $b \in \Sigma - \{S\}$ *which derive a word in* Δ^* *and for all* $j \geqq 1$, b *derives a word in* Δ^* *in* j *steps.*

Theorem 9.2. *Every* EPTOL *system that generates the* EOL *language* $L = \{a^{2n}b^{2n-1}a^{2n-2}\ldots b^3a^2b : n \geqq 1\}$ *is right-recursive.*

Proof. It follows from Example 9.1 and Corollary 3.1 that L is an EOL language. Let us assume that there is a non-right-recursive EPTOL system $G_1 = \langle \Sigma, \mathscr{H}, S, \{a, b\} \rangle$ with $\mathbf{L}(G_1) = L$.

For the rest of this proof we will make use of derivation trees as customary in L system theory (see, e.g., [RS]).

The symbol b is the rightmost symbol of every word in L. Thus, the rightmost path in every derivation tree of a word in L (up to the first occurrence of b) cannot be longer than $\# \Sigma$ because, otherwise, G_1 would be right-recursive. Since

L is an infinite language, it follows that b must derive itself (because, in a given ETOL derivation tree, all paths that lead from the root to a leaf have the same length). If there was a production (b, w) in G_1 with $w \neq b$ then b would be versatile and, hence, G_1 would be right-recursive. Therefore $h(b)=b$ for all $h \in \mathscr{H}$. The same argument can be used to show that $h(a)=a$, for all $h \in \mathscr{H}$, with an analogous reasoning for the second path from the right in derivation trees.

Note that, for every symbol $c \in \Sigma$ and for every word $x \in \Sigma^*$, if $c \overset{i}{\underset{G_1}{\Rightarrow}} x$ and $i \geqq \# \Sigma$, then either $x \in \Sigma^*\{a, b\}$ or $x \in \Sigma^*\{d\}$ for some non-versatile non-terminal symbol d (i.e. $h(d)=d$ for all $h \in \mathscr{H}$). In the latter case, x can never be rewritten into a string in Δ^*.

Let $G_2=\text{speed}_{\#\Sigma}(G_1)$. Note that $\mathbf{L}(G_2)=L$ and that G_2 is not right-recursive. Moreover, every production in G_2 that can occur in the derivation of a word $u \in L$ must be of the form (c, wa) or (c, wb), where $c \in \Sigma$ and $w \in \Sigma^*$.

Let us consider derivation trees of G_2. Since a and b derive only themselves, we will "prune" every path in such a tree at the uppermost occurrence (i.e. the one closest to the root) of a or b. We call the tree thus obtained a *pine tree* (see Fig. 9.1). Note that in a pine tree every rightmost path of every (non-trivial) subtree is of length 1.

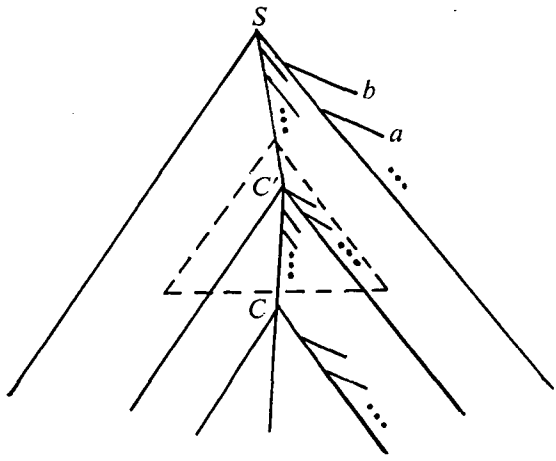

Fig. 9.1

Let us look at the pine tree of an arbitrarily large word $w \in L$. Let π be some path in that tree that contains more than one occurrence of a symbol. Let C' be the uppermost occurrence of some symbol c that occurs more than once on π and let C be the lowermost occurrence of c on π. Let v denote the subword of w derived from C' and let n be the number of symbols in w from v on to the right end of w. We will first show by contradiction that v cannot contain a subword of the from $a^ib^ja^k$ or $b^ia^jb^k$, for some $i, j, k \geqq 1$. We can intercalate the subtree with root C' (not including the subtree with root C) n times. We "fill" the pine tree arbitrarily to the left and to the right such that it yields a word w' in the language (this is possible by Lemma 9.1). Note that w' still contains v as a subword. It

contains, however, at least n symbols to the right of v, because each intercalated subtree derives at least one symbol to the right of C (we recall that every production with left hand side c must be of the form (c, wa) or (c, wb) if it is to occur in the derivation of a word in $\{a, b\}^*$). This is a contradiction. Hence, v must either be of the form a^ib^j or of the form b^ia^j for some i, j with $i+j \geqq 1$.

We can thus decompose every path π (from the root to a leaf) in the pine tree into two parts; π_1 is the subpath of π from the root to the first occurrence, say C', of the first symbol occurring more than once on π, and π_2 is the subpath of π from C' to the appropriate leaf. Note that every such path π_1 is of length $\leqq \# \Sigma$. Thus, there are at most $k = \text{Maxr}\,(G_2)^{\# \Sigma}$ distinct nodes in all these paths π_1 together.

For short, let us call a maximal adjacent sequence of a-s (b-s) in w a *block*. Note that every node in each path π_1 as above can derive at most 2 blocks in w. Thus, w can contain at most $2k$ blocks. This is a contradiction.

Thus every EPTOL system G with $\mathbf{L}(G) = L$ is right-recursive. □

Abstract

This paper investigates the possibilities of performing grammatical transformations on selective substitution grammars. The influence of the form of the selectors available on the possibilities of performing various grammatical constructions is considered. The grammatical transformations under investigation include standard ones, such as:

removing chain productions, removing λ-productions, restricting the right hand sides of productions to length 2 and synchronization.

*INSTITUTE OF MATHEMATICS
AND COMPUTER SCIENCE
THE HEBREW UNIVERSITY OF JERUSALEM
JERUSALEM, ISRAEL

**INSTITUTE OF APPLIED MATHEMATICS
AND COMPUTER SCIENCE
UNIVERSITY OF LEIDEN
P. O. BOX 9512
2300 RA LEIDEN
THE NETHERLANDS

References

[EMR] EHRENFEUCHT, A., H. MAURER and G. ROZENBERG, Continuous Grammars, *Inform. and Control.*, v. 46, 1980, pp. 71—91.

[HU] HOPCROFT, J. E and J. D. ULLMAN, *Introduction to automata theory, languages, and computation*, Addison-Wesley, Reading, Massachusetts, 1979.

[KR] KLEIJN, H. C. M. and G. ROZENBERG, Context-free like restrictions on selective rewriting, *Theoret. Comput. Sci.*, v. 16, 1981, pp. 237—269.

[KR2] KLEIJN, H. C. M. and G. ROZENBERG, Sequential, continuous and parallel grammars, *Inform. and Control*, v. 48, 1981, pp. 221—260.

[P] PENTTONEN, M., One-sided and two-sided context in formal grammars, *Inform. and Control*, v. 25, 1974, pp. 371—392.

[R] ROZENBERG, G., On slicing of K-iteration grammars, *Inform. Process. Lett.*, v. 4, 1976, pp. 127—131.

[R2] ROZENBERG, G., Selective substitution grammars (towards a framework for rewriting systems). Part I: Definitions and Examples. *Elektron. Informationsverarb. Kybernet*, v. 13, No. 9, 1977, pp. 455—463.

[RS] ROZENBERG, G. and A. SALOMAA, *The mathematical theory of L systems*, Academic Press, New York, 1980.

[RW] ROZENBERG, G. and D. WOOD, Context-free grammars with selective rewriting, *Acta Inform.*, v. 13, 1980, pp. 257—268.

[S] SALOMAA, A., *Formal languages*, Academic Press, New York, 1973.

(Received Jan. 13, 1983)