Tomus 6. Fasciculus 3.

ACTA CYBERNETICA

FORUM CENTRALE PUBLICATIONUM
CYBERNETICARUM HUNGARICUM

Szeged, 1983
Curat: Universitas Szegediensis de Attila József nominata

6. kötet 3. füzet

ACTA CYBERNETICA

A HAZAI KIBERNETIKAI KUTATÁSOK
KÖZPONTI PUBLIKÁCIÓS FÓRUMA

Szeged, 1983. december
A Szegedi József Attila Tudományegyetem gondozásában

All minimal clones on the three-element set

By B. CSÁKÁNY

1. Preliminaries

A *clone* on a set M is a set of finitary operations on M which is closed under composition and contains all projections. The clones on M form an algebraic lattice; the atoms and the dual atoms of this lattice are called *minimal clones* and *maximal clones* on M, respectively. A full description of all clones, hence of all minimal and maximal clones for $|M|=2$ was given by Post; a complete list of all maximal clones was found by Jablonskiĭ for $|M|=3$ and by Rosenberg for any finite M (see [15], [10], and [17]). Until now, only special examples of minimal clones were known for the case $|M|>2$. In this paper we determine all minimal clones on a three-element M.

We use the standard universal algebraic terminology [9] except that *function* stands for operation and *term function* for polynomial. All functions (and hence all clones) are defined on the base set $\mathbf{3}=\{0, 1, 2\}$. If f is a function, $[f]$ is the clone generated by f i.e. the clone of all term functions of the algebra $\langle \mathbf{3}; f\rangle$. Projections will also be called *trivial functions*. We use the notation σ for the set of triplets consisting of distinct entries from $\mathbf{3}$ and ι for $\mathbf{3}^3\setminus\sigma$.

In what follows we often make use of functions of the following types 1)—4).

1) *Unary functions.* Such a function f is denoted by u_n, where $n=9.f(0)+3.f(1)+f(2)$.

2) *Binary idempotent functions.* Such a function with the Cayley table

	0	1	2
0	0	n_5	n_4
1	n_3	1	n_2
2	n_1	n_0	2

will be denoted by b_n, where $n=\Sigma_{i=0}^{5}3^i n_i$.

3) *Majority functions.* A ternary function m satisfying $m(x, x, y)=m(x, y, x)=m(y, x, x)=x$ for any $x, y\in\mathbf{3}$ is called a majority function.

4) *Semiprojections.* A ternary function s is called a semiprojection if there exists a $k\in\mathbf{3}$ such that $s(x_0, x_1, x_2)=x_k$ for arbitrary $\langle x_0, x_1, x_2\rangle\in\iota$.

A function belonging to one of the above four classes will be referred to as a *special function*. For $n>1$ we call an n-ary nontrivial function f *sharp* if $f(x_0, \ldots, x_{j-1}, x_i, x_{j+1}, \ldots, x_{n-1})=x_{m_{ij}}$ where $0\leqq m_{ij}<n$ for all $i\neq j$, $0\leqq i, j<n$. A binary function is sharp iff it is idempotent. Majority functions and nontrivial semiprojections are sharp ternary. As a trivial consequence of the definition, an n-ary function f is sharp iff all k-ary functions in $[f]$ are trivial provided $k<n$.

A sharp ternary function f is uniquely determined by the values $f(0, 0, 1)$, $f(0, 1, 0)$, $f(0, 1, 1)$, and all $f(\varphi)$ with $\varphi\in\sigma$. Call the numbers

$$\chi(f) = 4.f(0, 0, 1)+2.f(0, 1, 0)+f(0, 1, 1)$$

and

$$\mu(f) = 3^5.f(0, 1, 2)+3^4.f(0, 2, 1)+3^3.f(1, 0, 2)+3^2.f(1, 2, 0)+$$
$$+3.f(2, 0, 1)+f(2, 1, 0)$$

the *characteristic* and *mantissa* of f, and let the pair $\chi(f), \mu(f)$ stand for f. For example 4,44 is Pixley's ternary discriminator function ([20], p. 8) and 1,624 is the dual discriminator [8] on 3. Observe that $1, t$ $(t=0, \ldots, 728)$ are the majority functions, and $0, t$ are the semiprojections with $i=0$. As these ternary functions will play an important role, we also use an alternative notation m_t for $1, t$ and s_t for $0, t$; e.g. m_{728} is the majority function on 3 whose value is 2 on each $\varphi\in\sigma$. Clearly, every majority function or semiprojection f is uniquely determined by the sequence of its values on σ, called the *range* of f; further, the number $v(f)$ of distinct entries in the range of f is called the *variance* of f.

Let φ be a permutation of 3. To each n-ary function f we assign f^φ, called a *conjugate* of f, defined by $f^\varphi(x_0, \ldots, x_{n-1})=(f(x_0\varphi^{-1}, \ldots, x_{n-1}\varphi^{-1}))\varphi$. The map $f\to f^\varphi$ carries each clone $\mathscr{C}$ onto the clone $\mathscr{C}^\varphi$; in particular $[f]^\varphi=[f^\varphi]$, and

(*) $$g\in[f] \quad \text{implies} \quad g^\varphi\in[f^\varphi].$$

We can permute the variables of f as well: for a permutation ψ of $\mathbf{n}$ $(=\{0, \ldots, n-1\})$ put $f_\psi(x_0, \ldots, x_{n-1})=f(x_{0\psi}, \ldots, x_{(n-1)\psi})$. Remark that always $(f^\varphi)_\psi=(f_\psi)^\varphi$; hence we can write simply f^φ_ψ. Note also that $[f_\psi]=[f]$ for any ψ.

The conjugations and permutations of variables generate a permutation group T_n of order $3!n!$ on the set of all n-ary functions on 3. The classes 1)—4) are closed with respect to T_1, T_2, T_3, and T_3 respectively. Two functions are said to be *essentially distinct* if they have different arities, or belong to distinct orbits of T_n.

We conclude the introduction with the following immediate observation: a nontrivial clone $\mathscr{C}$ is minimal iff $\mathscr{C}=[f]$ for each nontrivial $f\in\mathscr{C}$.

2. The list of minimal clones

First we approximately locate the functions generating minimal clones.

Proposition. *Let $\mathscr{C}$ be a minimal clone and m the minimum arity of nontrivial functions from $\mathscr{C}$. Then $1\leqq m\leqq 3$. If $f\in\mathscr{C}$ is m-ary then f is special; moreover if $f, g\in\mathscr{C}$ are m-ary then both f and g are of the same type i $(1\leqq i\leqq 4)$.*

Proof. The statement is clear if $m=1$. If $m=2$, then each nontrivial binary function $g\in\mathscr{C}$ is sharp and hence idempotent.

Let $m\geqq 3$. First we show that $\mathscr{C}$ contains a sharp ternary function. Indeed, to each sharp at least quaternary f on **3** there exists an i such that $f(x_0, \ldots, x_{n-1})=x_i$ $(i\in\mathbf{n})$ whenever $x_0, \ldots, x_{n-1}$ are not all distinct ([18]; see also the proof of Thm. 1, § 33 in [9]). This proves that on **3** each sharp function is at most ternary.

Let t be a sharp ternary function from $\mathscr{C}$. We show that $[t]$ contains a nontrivial semiprojection or a majority function. Indeed, if $\chi(t)=0$, 3 or 5 then t itself is a semiprojection, and if $\chi(t)=1$ then t is a majority function. Thus let $\chi(t)\notin\{0, 1, 3, 5\}$. We show that $\chi(g)=6$ for some $g\in[t]$. First if $\chi(t)=2$ $(\chi(t)=7)$ then exchanging the last (first) two variables we obtain r with $\chi(r)=4$. Thus let $\chi(t)=4$. Then the characteristic of

$$t(t(x, y, z),\ t(x, y, t(x, z, y)),\ t(x, z, y))$$

equals 6 (as direct verification or Lemma 1.10 in [20] would show). Let $\chi(t)=6$. Write $t'(x, y, z)$ for $t(t(x, y, z), y, z)$. Then $\chi(t')=0$ and if $t'\neq e_1^3$ $(=0{,}44)$ we are done. Now if $\mu(t)\neq 44$ then $t(a, b, c)=d\neq a$ for some $\langle a, b, c\rangle\in\sigma$ and therefore $t'(a, b, c)=t(d, b, c)=d\neq a$. It remains to consider $\mu(t)=44$. In this case $t(y, t(z, y, x), z)=0{,}424$ is the required semiprojection.

We thus have that there is a majority function or a nontrivial semiprojection in $\mathscr{C}$. Taking into account that for g a semiprojection each ternary $f\in[g]$ is a semiprojection and that a similar assertion is valid for majority functions, this proves the first part of the proposition. The second part is implied by the following simple observation: if $i\prec j$ and the functions f and g are of type i) and j) then $f\notin[g]$.

In virtue of the Proposition our task is to find the different minimal clones generated by functions of the four types above.

1) *Unary functions.* A nontrivial unary function f generates a minimal clone iff either f is a retraction of M (i.e. $f\circ f=f$) or a permutation of prime order ([16], Theorem 4.4.1). The functions u_0 and u_2 are representatives of the two orbits of retractions while u_7 and u_{15} are representatives of the two orbits of prime order permutations. The table below shows the minimal clones generated by unary functions on **3**. The clone standing at the meet of the row starting with $[u]$ and column marked by the permutation φ is $[u]^\varphi$. The place of $[u]^\varphi$ is empty if $[u]^\varphi$ is equal to some $[u]^\psi$ which appeared earlier. One may check directly that the clones in the table are pairwise distinct.

Table 1

	(01)	(02)	(12)	(012)	(021)
$[u_0]$	$[u_{13}]$	$[u_{26}]$			
$[u_2]$	$[u_{14}]$	$[u_8]$	$[u_3]$	$[u_4]$	$[u_{23}]$
$[u_7]$	$[u_{21}]$	$[u_{11}]$			
$[u_{15}]$					

2) *Binary functions.* We proved in [6]: every minimal clone on **3** containing an essentially binary operation is a conjugate of exactly one of the following twelve clones: $[b_i]$ with $i\in\{0, 8, 10, 11, 16, 17, 26, 33, 35, 68, 178, 624\}$.

Table 2 (which is constructed on the same principle as Table 1) displays the minimal clones generated by binary functions on **3**. The clones are pairwise distinct

because each $[b]$ in the table contains no nontrivial binary function other than $b(x, y)$ and its dual $b(y, x)$. In other words, the free algebra with two free generators in the variety generated by $\langle \mathbf{3}; b\rangle$ consists of no more than four elements; in this form, our observation may be found in Berman's paper on three element algebras [2]. Thus, it remains to check that Table 2 has no pair of dual functions.

Table 2

	(01)	(02)	(12)	(012)	(021)
$[b_0]$	$[b_{324}]$	$[b_{728}]$			
$[b_8]$	$[b_{368}]$	$[b_{80}]$	$[b_{36}]$	$[b_{40}]$	$[b_{692}]$
$[b_{10}]$	$[b_{280}]$	$[b_{458}]$	$[b_{20}]$	$[b_{448}]$	$[b_{188}]$
$[b_{11}]$	$[b_{286}]$	$[b_{215}]$			
$[b_{16}]$	$[b_{281}]$	$[b_{296}]$	$[b_{47}]$	$[b_{205}]$	$[b_{179}]$
$[b_{17}]$	$[b_{287}]$	$[b_{53}]$	$[b_{38}]$	$[b_{43}]$	$[b_{206}]$
$[b_{26}]$	$[b_{449}]$		$[b_{37}]$		
$[b_{33}]$	$[b_{122}]$	$[b_{557}]$			
$[b_{35}]$	$[b_{125}]$	$[b_{71}]$	$[b_{42}]$	$[b_{41}]$	$[b_{530}]$
$[b_{68}]$	$[b_{528}]$	$[b_{116}]$			
$[b_{178}]$	$[b_{290}]$				
$[b_{624}]$					

The next lemma serves as a tool for handling the remaining two cases. It is a direct consequence of the definition of a minimal clone.

Lemma 1. *Let G, H be subsets of the set F of special functions such that*

I. $H \subseteq G$;

II. $[g] \cap F \subseteq G$ *for every* $g \in G$;

III. $[g] \cap H \neq \emptyset$ *for every* $g \in G$;

IV. *If* $h_1, h_2 \in H$ *and* $h_1 \neq h_2$ *then* $h_1 \notin [h_2]$.

Then $\{[h]: h \in H\}$ is the set of pairwise distinct minimal clones generated by $g \in G$.

3. *Majority functions.* For a majority function f generating a minimal clone we have two possibilities:

a) $v(f)=3$. We prove that up to permutation of variables f is the dual discriminator d (i.e. the majority function with $d(a, b, c)=c$ if $\langle a, b, c\rangle \in \sigma$). Following [8], a non-empty binary relation C on $\mathbf{3}$ is called *p-rectangular* if for every pair $\langle i, j\rangle \notin C$ there are no more than two elements in C which have the form $\langle i, x\rangle$ or $\langle y, j\rangle$. First we show that the subalgebras of $\langle \mathbf{3}; f\rangle^2$ are p-rectangular relations on $\mathbf{3}$. In the opposite case, we may suppose without loss of generality (renaming the elements and taking C^{-1} if necessary) that there is a subalgebra C of $\langle \mathbf{3}; f\rangle^2$ such that $\langle 0, 0\rangle \notin C$ but $\langle 0, 1\rangle, \langle 0, 2\rangle, \langle 1, 0\rangle \in C$. By assumption $v(f)=3$, the range of f contains 0, and we may permute the variables of f so that $f(2, 0, 1)=0$. Then $\langle 0, 0\rangle = \langle f(0, 1, 0), f(2, 0, 1)\rangle = f(\langle 0, 2\rangle, \langle 1, 0\rangle, \langle 0, 1\rangle) \in C$, a contradiction. However, d preserves the p-rectangular relations on $\mathbf{3}$ (see [8]), hence it preserves all subalgebras of $\langle \mathbf{3}; f\rangle^2$. Now we can apply the following theorem of Baker and Pixley ([1], see also [20], Theorem 1.2): if a finite algebra $\mathbf{A}$ has a term function which is a majority function, then every function preserving all subalgebras of $\mathbf{A}^2$ is a term function of $\mathbf{A}$. We obtain that d is a term function of $\langle \mathbf{3}; f\rangle$, hence $d \in [f]$. On the other hand, $[d]$ is a minimal clone ([7], Theorem 1), hence $f \in [d]$.

As d is a homogeneous function (i.e. a function preserving all permutations), f must be a homogeneous majority function. Thus f coincides with d up to ordering of variables.

b) $v(f)<3$. There exists $3\cdot 2^6-3=189$ majority functions with this property, and they belong to 10 distinct orbits of T_3. Here we list the representatives having the least index for these orbits (the number in brackets indicates the number of functions constituting the represented orbit):

Table 3

$m_0(3)$	$m_{13}(18)$	$m_{109}(6)$
$m_1(36)$	$m_{28}(36)$	$m_{120}(18)$
$m_4(18)$	$m_{39}(18)$	
$m_{10}(18)$	$m_{85}(18)$	

We prove that the minimal clones generated by majority functions having variance less than 3 are exactly $[m_0]$, $[m_{324}]$ $(m_{324}=(m_0)^{(01)})$, $[m_{728}]$ $(m_{728}=(m_0)^{(02)})$, $[m_{109}]$, $[m_{473}]$ $(m_{473}=(m_{109})^{(02)})$, and $[m_{510}]$ $(m_{510}=(m_{109})^{(12)})$; note that the remaining functions in the orbit of m_{109} may be obtained from the listed ones by permutations of variables and hence do not generate further clones.

Apply Lemma 1. Let F be the set of special functions, G the set of majority functions with variance <3, and $H=\{m_i\colon i=0, 364, 728, 109, 473, 510\}$. The requirement I is fulfilled by definition. As for II, it is a consequence of the following lemma which will be used once more later.

Lemma 2. *Let f be a majority function whose range does not contain the element a ($\in\mathbf{3}$). Then the same holds for every non-trivial ternary function in $[f]$.*

Proof. Clearly, a non-trivial ternary function in $[f]$ is a majority function. Assume that $[f]$ contains a non-trivial ternary function whose range includes a, and let g be such a function with a shortest f-term: $g(x, y, z)=f(g_0(x, y, z), g_1(x, y, z), g_2(x, y, z))$. For a suitable $\varphi\in\sigma$ we have $g(\varphi)=f(g_0(\varphi), g_1(\varphi), g_2(\varphi))=a$. Thus $\langle g_0(\varphi), g_1(\varphi), g_2(\varphi)\rangle\in\iota$, and hence $g_i(\varphi)=a$ for at least two distinct $i\in\mathbf{3}$. By the minimality of g, these g_i must be trivial and hence both of them equal the same projection. But then g also equals that projection, i.e. g is trivial, a contradiction.

Next we prove III. In virtue of (*) it is enough to show that for each function m_i in Table 3 there is an $m_j\in H$ such that $m_j\in[m_i]$. Write $\hat{f}=\hat{f}(x, y, z)$ for $f(f, f_{(012)}, f_{(021)})=f(f(x, y, z), f(y, z, x), f(z, x, y))$. Then

$$m_0=\hat{m}_1=\hat{m}_4=\hat{m}_{10}=\hat{m}_{120}$$

and

$$m_{109}=(\hat{m}_{13})_{(01)}=\hat{m}_{28}=(\hat{m}_{39})_{(01)}=\hat{m}_{85},$$

i.e. $m_0\in[m_1]$, $[m_4]$, $[m_{10}]$, $[m_{120}]$ and $m_{109}\in[m_{13}]$, $[m_{28}]$, $[m_{39}]$, $[m_{85}]$, as required.

Finally, we have to prove that IV is fulfilled, i.e. that none of the functions in H is contained in the clone generated by another one. The unique non-trivial permutation of $\mathbf{3}$ preserved by m_{105} and m_{728} is (01), that preserved by m_{324} and m_{510} is (02), and that preserved by m_0 and m_{473} is (12). Hence no function in one of these three pairs is included in the clone generated by a function appearing

in another pair. Furthermore, the ranges of m_{105} and m_{728} have no common entry; thus, by Lemma 2, $m_{109} \notin [m_{728}]$ and $m_{728} \notin [m_{109}]$. Concerning the remaining pairs we can argue in the same way.

Now we see that the conclusion of Lemma 1 also holds. This combined with the result in the case $v(f)=3$ allows us to summarize the minimal clones generated by majority functions as follows:

Table 4

	(01)	(02)	(12)
$[m_0]$	$[m_{324}]$	$[m_{728}]$	
$[m_{109}]$		$[m_{473}]$	$[m_{510}]$
$[m_{624}]$			

4) *Semi-projections.* There are $3 \cdot 3^6 = 2187$ semiprojections, and they belong to 74 distinct orbits of T_3. Table 5 shows representatives of these orbits (the number of functions in the orbit is added in brackets if it differs from 36). Every orbit is represented by its member of characteristic 0 having the least mantissa. For the sake of brevity we write down the mantissa only.

Table 5

0(9)	21(18)	86	108	150(12)
1	22	87	109(18)	153
2	23	88	110	154
4(18)	25	90	111(18)	156
5	26(18)	91	113	157
8(18)	44(3)	92	126	324(18)
10	49	96	127	325(18)
11(18)	50(18)	99	128	342
12	52(18)	100	135	343
13	76(9)	101	136	345
14	81	102	138	346(18)
15(18)	82	103	139(18)	396(6)
16	83	104(18)	140	424(6)
17	84	105	141	426(18)
19(18)	85	106	144	

We prove that the semi-projections generating minimal clones are exactly $s_0, s_8, s_{26}, s_{76}, s_{424}$, and the functions in their orbits. This means that the minimal clones generated by semi-projections are exactly those in the following table (which is constructed in the usual manner).

Table 6

	(01)	(02)	(12)	(012)	(021)
$[s_0]$	$[s_{364}]$	$[s_{728}]$			
$[s_8]$	$[s_{368}]$	$[s_{80}]$	$[s_{36}]$	$[s_{40}]$	$[s_{692}]$
$[s_{26}]$	$[s_{449}]$		$[s_{97}]$		
$[s_{76}]$	$[s_{684}]$	$[s_{332}]$			
$[s_{424}]$					

Denote by S the set of functions appearing in Table 6, and let S_i stand for the set of conjugates of s_i.

For the proof, we again apply Lemma 1. Let F be the set of special functions, G the set of semi-projections, and $H=S$. Clearly, they fulfil I and II. However, the verification of III and IV demands tiresome computations.

Consider two semi-projections, s_i, s_j, which are representatives of orbits of T_3 (i.e. whose indices appear in Table 5). Draw an arrow from s_i to s_j if there exists an s_i-term function which is conjugate to s_j (i.e. $(s_j)^\varphi \in [s_i]$ for some permutation of **3**). To prove III, in view of (*) it is sufficient to produce a set of arrows such that in the resulting oriented graph, for each non-trivial representative s there exists a path which starts from s and ends in one of s_0, s_8, s_{26}, s_{76}, and s_{424}. Such a set of 68 arrows is in the appendix at the end of the paper.

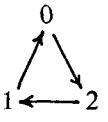
Our concluding task is to prove that for any different functions $s_i, s_j \in S$ $s_j \notin [s_i]$ is valid. We start with a trivial lemma.

Lemma 3. *Let f, g be functions. If there exist a subset K and an element k of some direct power $\mathbf{3}^n$ such that k belongs to the subalgebra of $\langle \mathbf{3}; g\rangle^n$ generated by K but does not belong to the subalgebra of $\langle \mathbf{3}; f\rangle^n$ generated by K, then $g \notin [f]$.*

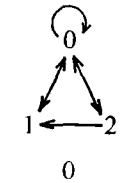
We also need a special case of this lemma.

Lemma 3*. *Let f, g be as above. If there exists a permutation of M which is an automorphism of the algebra $\langle M; f\rangle$ but not of the algebra $\langle M; g\rangle$ then $g \notin [f]$* (cf. the proof of IV for majority functions with variance <3).

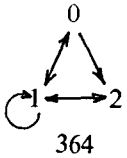
Apply Lemma 3 for the case $M=\mathbf{3}$, $n=2$, $K=\{\langle 0, 2\rangle, \langle 1, 0\rangle, \langle 2, 1\rangle\}$. The set K may be visualized by means of the figure



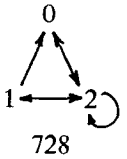
Then the subalgebras of $\langle \mathbf{3}; s_i\rangle^2$ generated by K appear in the following figures (i is indicated below the given figure):



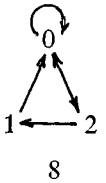
0



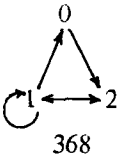
364



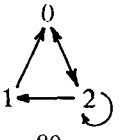
728



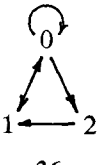
8



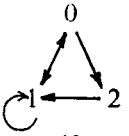
368



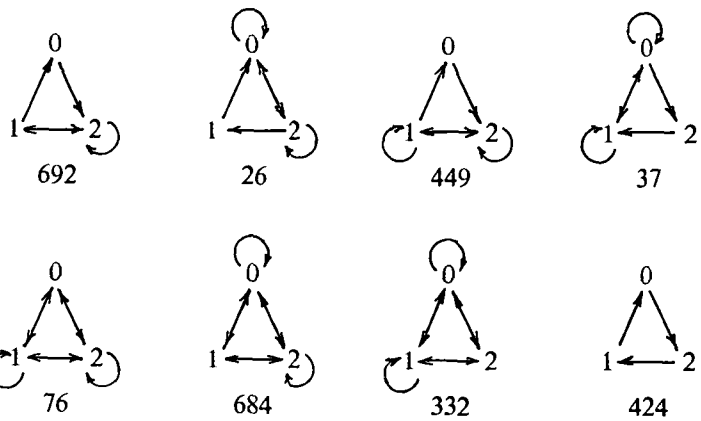
80



36



40



It can be read:

α) $[s_{424}] \cap S = \{s_{424}\}$,

β) if $s \in S_8$, $[s] \cap S = \{s\}$,

γ) if $s \in S_0 \cup S_{26}$, then $[s] \cap (S_0 \cup S_{26} \cup S_{76}) = \{s\}$,

δ) if $s \in S_{76}$, then $[s] \cap S_{76} = \{s\}$,

ε) $s_0 \notin [s_{76}]$, $s_{364} \notin [s_{684}]$, $s_{728} \notin [s_{332}]$.

Observe further that for $s_i \in S_0 \cup S_{26} \cup S_{76}$ the algebra $\langle \mathbf{3}; s_i \rangle$ has a non-trivial automorphism, while for $s_j \in S_8$ the algebra $\langle \mathbf{3}; s_j \rangle$ does not. Hence, by Lemma 3*, we have

ζ) if $s_i \in S_0 \cup S_{26} \cup S_{76}$ and $s_j \in S_8$, then $s_j \notin [s_i]$.

The transposition (01) is an automorphism of $\langle \mathbf{3}; s_{332} \rangle$ and not an automorphism of $\langle \mathbf{3}; s_i \rangle$ for $i = 0, 364, 26, 449$. Hence

η) $s_i \notin [s_{332}]$ for $i = 0, 364, 26, 49$.

Similarly, with the aid of (02) we obtain

θ) $s_i \notin [s_{684}]$ for $i = 0, 728, 449, 37$,

and using (12) there follows

ι) $s_i \notin [s_{76}]$ for $i = 364, 728, 26, 37$.

Lemma 3 can be used also to prove

$\varkappa$) for $s_i \in S$, $i \neq 424$, always $s_{424} \notin [s_i]$.

Indeed, subalgebras of $\langle \mathbf{3}; s_{424} \rangle^2$ must be p-rectangular, since for a subalgebra C and distinct elements $x_1, x_2 \in \mathbf{3}$ it is impossible to have $\langle i, j \rangle \notin C$ and $\langle i, x_1 \rangle$, $\langle i, x_2 \rangle$, $\langle y, j \rangle \in C$ because of $s_{424}(\langle i, x_1 \rangle, \langle y, j \rangle, \langle i, x_2 \rangle) = \langle i, j \rangle$. Now, for $s_i \in S$, $i \neq 424$, consider the subset of $\mathbf{3}^2$ displayed in the above figure with subscript i, and call it K_i. Take this subset for K, s_i for f, and s_{424} for g. Then, by its definition, K_i is a subalgebra of $\langle \mathbf{3}; s_i \rangle^2$ but it is not p-rectangular (see the figure) and thus the subalgebra of $\langle \mathbf{3}; s_{424} \rangle$ generated by K_i contains at least one element $k \notin K_i$. Lemma 3 then applies and gives $\varkappa$).

The last step is:

λ) $s_{449} \notin [s_{76}]$, $s_{26} \notin [s_{684}]$, $s_{37} \notin [s_{332}]$.

We prove the first assertion, the proof of the others being similar. As the range of s_{449} does not contain 0, we are done if we show that the range of any function in $[s_{76}]$ contains 0. In the contrary case, take the shortest s_{76}-term function t whose

range does not contain 0. Let $t(x, y, z)=s_{76}(t_0(x, y, z), t_1(x, y, z), t_2(x, y, z))$. Then $t_0(\varphi)=0$ for a suitable $\varphi\in\sigma$, and hence $t(\varphi)=0$, a contradiction.

α)—λ) together mean that for any $s_i, s_j\in S$, $s_j\notin[s_i]$ whenever $i\neq j$. We have verified IV of Lemma 1, hence Table 6 really is the list of all minimal clones on **3** generated by semi-projections.

3. Summary

In the preceding part we have proved the following result:

Theorem. *There exist* 84 *minimal clones on* **3** *and each of them is a conjugate of exactly one of the following* 24 *clones:*
$[u_0]$, $[u_2]$, $[u_7]$, $[u_{15}]$;
$[b_0]$, $[b_8]$, $[b_{10}]$, $[b_{11}]$, $[b_{16}]$, $[b_{17}]$, $[b_{26}]$, $[b_{33}]$, $[b_{35}]$, $[b_{68}]$, $[b_{178}]$, $[b_{624}]$;
$[m_0]$, $[m_{109}]$, $[m_{624}]$;
$[s_0]$, $[s_8]$, $[s_{26}]$, $[s_{76}]$, $[s_{424}]$.

The other minimal clones may be read from Tables 1, 2, 4, and 6. The Cayley tables of the binary functions in the Theorem may be found in [6]. The ranges of the ternary functions appearing in the Theorem may be seen in the following:

Table 7

f	$f(0,1,2)$	$f(0,2,1)$	$f(1,0,2)$	$f(1,2,0)$	$f(2,0,1)$	$f(2,1,0)$
m_0	0	0	0	0	0	0
m_{109}	0	1	1	0	0	1
m_{624}	2	1	2	0	1	0
s_0	0	0	0	0	0	0
s_8	0	0	0	0	2	2
s_{26}	0	0	0	2	2	2
s_{76}	0	0	2	2	1	1
s_{424}	1	2	0	2	0	1

Several functions occurring here are familiar from earlier research. $\langle\mathbf{3}; b_0\rangle$ and $\langle\mathbf{3}; b_{10}\rangle$ are the two three-element semilattices. They and $\langle\mathbf{3}; b_{11}\rangle$, $\langle\mathbf{3}; b_{26}\rangle$ (left zero semigroups with an outer zero, resp. unit element) are bands satisfying $xyx=xy$ identically. Hence the minimality of clones they generate is involved by Theorem 4.4.4 in the book of Pöschel and Kalužnin [16]. The minimality of $[b_{178}]$ was proved in [12] by Marčenkov, Demetrovics, and Hannák ($\langle\mathbf{3}; b_{178}\rangle$ is a tournament; it is known as "the paper-stone-scissors algebra"). The algebra $\langle\mathbf{3}; b_{624}\rangle$ is essentially the affine space [4] over GF(3).

The functions b_8, b_{11}, b_{35}, and b_{68} appeared in Płonka's paper [13]; the last one goes back to Takasaki [19]. Kepka deals with b_{16} and b_{17} in [11].

As for the ternary functions, m_0 is the median function on the chain $1<0<2$, and the minimality of $[m_0]$ is a special case of 4.4.5 in [16]. Finally, the minimality of $[m_{624}]$ and $[s_{424}]$ was established in [7].

Acknowledgements. Thanks are due to Joel Berman, Károly Dévényi, Isidore Fleischer, Péter P. Pálfy, László Szabó and Ágnes Szendrei for their interest and com-

ments. I am especially indebted to Ivo G. Rosenberg for his helpful criticism. This research was partly supported by NSERC Canada grant A-5407. The Centre de recherche de mathématiques appliquées, Université de Montréal, provided excellent conditions for the completion of this work. The computing was done at the Kalmár Laboratory of Cybernetics of the József Attila University, Szeged.

Apology. I am aware that this research may be regarded as something in computer-assisted quasi-mathematics (cf. [3], p. 14). However, I am convinced that the determination of minimal clones is important; hence I feel that the result should be published in spite of the fact that my way is tedious and several steps were obtained through a computer search. Let us consider this paper as a modest step towards the description of minimal clones.

Appendix

$s_1 \to s_0 = s_1(s_1, (s_1)_{(12)}, x) = s_1(s_1(x, y, z), s_1(x, z, y), x)$
$s_2 \to s_0 = s_2(s_2, (s_2)_{(12)}, y)$
$s_4 \to s_0 = s_4(s_4, (s_4)_{(02)}, x)$
$s_5 \to s_1 = s_5(s_5, (s_5)_{(02)}, (s_5)_{(12)})$
$s_{10} \to s_1 = s_{10}(s_{10}, (s_{10})_{(12)}, y)$
$s_{11} \to s_0 = s_{11}(s_{11}, (s_{11})_{(12)}, (s_{11})_{(01)})$
$s_{12} \to s_1 = s_{12}((s_{12})_{(12)}, s_{12}, z)$
$s_{13} \to s_4 = s_{13}(s_{13}, (s_{13})_{(12)}, y)$
$s_{14} \to s_4 = s_{14}(s_{14}, (s_{14})_{(02)}, (s_{14})_{(01)})$
$s_{15} \to s_0 = s_{15}(s_{15}, (s_{15})_{(021)}, (s_{15})_{(012)})$
$s_{16} \to s_4 = s_{16}(s_{16}, (s_{16})_{(12)}, (s_{16})_{(012)})$
$s_{17} \to s_8 = s_{17}(s_{17}, (s_{17})_{(12)}, (s_{17})_{(01)})$
$s_{19} \to s_0 = s_{19}(s_{19}, (s_{19})_{(12)}, (s_{19})_{(01)})$
$s_{21} \to s_1 = s_{21}((s_{21})_{(12)}, (s_{21})_{(02)}, s_{21})$
$s_{22} \to s_{13} = s_{22}(s_{22}, (s_{22})_{(12)}, (s_{22})_{(01)})$
$s_{23} \to s_{11} = s_{23}(s_{23}, (s_{23})_{(02)}, x)$
$s_{25} \to s_{23} = s_{25}(s_{25}, (s_{25})_{(12)}, (s_{25})_{(02)})$
$s_{49} \to s_8 = (s_{49}(s_{49}, (s_{49})_{(12)}, (s_{49})_{(02)}))^{(021)}$
$s_{50} \to s_{49} = s_{50}(s_{50}, (s_{50})_{(02)}, y)$
$s_{52} \to s_{76} = s_{52}(s_{52}, (s_{52})_{(12)}, (s_{52})_{(01)})$
$s_{81} \to s_0 = s_{81}(s_{81}, (s_{81})_{(12)}, y)$
$s_{82} \to s_1 = s_{82}(s_{82}, (s_{82})_{(12)}, y)$
$s_{83} \to s_0 = s_{83}(s_{83}, (s_{83})_{(12)}, y)$
$s_{84} \to s_1 = s_{84}(s_{84}, x, y)$
$s_{85} \to s_1 = s_{85}(s_{85}, x, y)$
$s_{86} \to s_2 = s_{86}(s_{86}, (s_{86})_{(12)}, (s_{86})_{(012)})$
$s_{87} \to s_2 = s_{87}((s_{87})_{(12)}, s_{87}, (s_{87})_{(021)})$
$s_{88} \to s_4 = s_{88}(s_{88}, (s_{88})_{(12)}, y)$
$s_{90} \to s_0 = s_{90}(s_{90}, (s_{90})_{(12)}, y)$
$s_{91} \to s_1 = s_{91}(s_{91}, (s_{91})_{(12)}, y)$
$s_{92} \to s_0 = s_{92}(s_{92}, (s_{92})_{(12)}, y)$
$s_{96} \to s_5 = s_{96}((s_{96})_{(12)}, s_{96}, z)$
$s_{99} \to s_0 = s_{99}(s_{99}, (s_{99})_{(12)}, (s_{99})_{(02)})$

$s_{100} \to s_1 = s_{100}(s_{100}, (s_{100})_{(12)}, (s_{100})_{(02)})$
$s_{101} \to s_2 = s_{101}(s_{101}, (s_{101})_{(12)}, (s_{101})_{(02)})$
$s_{102} \to s_{15} = s_{102}(s_{102}, (s_{102})_{(021)}, (s_{102})_{(02)})$
$s_{103} \to s_{16} = s_{103}(s_{103}, (s_{103})_{(021)}, (s_{103})_{(02)})$
$s_{104} \to s_{26} = s_{104}(s_{104}, (s_{104})_{(12)}, y)$
$s_{105} \to s_2 = s_{105}((s_{105})_{(12)}, x, (s_{105})_{(021)})$
$s_{106} \to s_{25} = s_{106}(s_{106}, (s_{106})_{(12)}, (s_{106})_{(01)})$
$s_{108} \to s_{17} = s_{108}(x, z, (s_{108})_{(02)})$
$s_{109} \to s_{26} = (s_{109}(s_{109}, y, (s_{109})_{(12)}))^{(12)}$
$s_{110} \to s_{15} = s_{110}((s_{110})_{(12)}, (s_{110})_{(02)}, s_{110})$
$s_{111} \to s_{109} = s_{111}(s_{111}, x, (s_{111})_{(12)})$
$s_{113} \to s_{16} = s_{113}((s_{113})_{(12)}, (s_{113})_{(02)}, s_{113})$
$s_{126} \to s_{111} = s_{126}(s_{126}, (s_{126})_{(021)}, (s_{126})_{(02)})$
$s_{127} \to s_{109} = s_{127}(s_{127}, (s_{127})_{(012)}, x)$
$s_{128} \to s_{126} = s_{128}(s_{128}, (s_{128})_{(12)}, (s_{128})_{(012)})$
$s_{135} \to s_0 = s_{135}(s_{135}, (s_{135})_{(021)}, (s_{135})_{(012)})$
$s_{136} \to s_{135} = s_{136}(s_{136}, (s_{136})_{(012)}, z)$
$s_{138} \to s_{135} = s_{138}(s_{138}, (s_{138})_{(02)}, (s_{138})_{(012)})$
$s_{139} \to s_{26} = s_{139}(x, s_{139}, (s_{139})_{(12)})$
$s_{140} \to s_{26} = s_{140}((s_{140})_{(12)}, (s_{140})_{(02)}, s_{140})$
$s_{141} \to s_{26} = s_{141}(x, s_{141}, (s_{141})_{(12)})$
$s_{144} \to s_{90} = s_{144}(s_{144}, (s_{144})_{(12)}, (s_{144})_{(02)})$
$s_{150} \to s_{396} = s_{150}(s_{150}, (s_{150})_{(01)}, (s_{150})_{(12)})$
$s_{153} \to s_{135} = s_{153}(s_{153}, (s_{153})_{(02)}, (s_{153})_{(021)})$
$s_{154} \to s_{150} = s_{154}(x, (s_{154})_{(02)}, s_{154})$
$s_{156} \to s_{105} = s_{156}(s_{156}, (s_{156})_{(02)}, y)$
$s_{157} \to s_{17} = (s_{157}(x, s_{157}, (s_{157})_{(02)}))^{(021)}$
$s_{324} \to s_{111} = s_{324}(s_{324}, x, z)$
$s_{325} \to s_{111} = s_{325}(s_{325}, x, z)$
$s_{342} \to s_{111} = s_{342}(s_{342}, x, z)$
$s_{343} \to s_{19} = s_{343}(x, s_{343}, (s_{343})_{(12)})$
$s_{345} \to s_{19} = s_{345}(x, s_{345}, (s_{345})_{(12)})$
$s_{346} \to s_{26} = s_{346}(x, s_{346}, (s_{346})_{(12)})$
$s_{396} \to s_{424} = s_{396}(s_{396}, (s_{396})_{(02)}, x)$
$s_{426} \to s_{111} = s_{426}(s_{426}, x, z)$

BOLYAI INSTITUTE
A. JÓZSEF UNIVERSITY
ARADI VÉRTANÚK TERE 1
SZEGED, HUNGARY
H-6720

References

[1] BAKER, K. A, A. F. PIXLEY, Polynomial interpolation and the Chinese Remainder Theorem for algebraic systems, *Math. Zeitschrift*, v. 143, 1975, pp. 165—174.
[2] BERMAN, J., Free spectra of 3-element algebras, Preprint.
[3] BONSALL, F. F., A down-to-earth view of mathematics, *Amer. Math. Monthly*, v. 89, No. 1, 1982. pp. 8—15.
[4] CSÁKÁNY, B., Affine spaces over prime fields, *Acta Sci. Math. (Szeged)*, v. 37, 1975, pp. 33—36.

[5] Csákány, B., Homogeneous algebras are functionally complete, *Algebra Universalis*, v. 11, 1980, pp. 149—158.
[6] Csákány, B., Three-element groupoids with minimal clones, *Acta Sci. Math. (Szeged)*, v. 45, 1983, pp. 111—117.
[7] Csákány, B., T. Gavalcová, Finite homogeneous algebras I, *Acta Sci. Math. (Szeged)*, v. 42, 1980, pp. 57—65.
[8] Fried, E., A. F. Pixley, The dual discriminator function in universal algebra, *Acta Sci. Math. (Szeged)*, v. 41, 1979, pp. 83—100.
[9] Grätzer, G., *Universal Algebra*, Van Nostrand, Princeton, 1968.
[10] Jablonskiĭ, S. V., Functional constructions in k-valued logic (Russian), *Trudy Mat. Inst. Steklov*, v. 51, 1958, pp. 5—142.
[11] Kepka, T., Quasi-trivial groupoids and balanced identities, *Acta Univ. Carolin.*, v. 22, 1981. pp. 49—64.
[12] Marčenkov, S. S., J. Demetrovics, L. Hannák, On the closed classes of selfdual functions in P_3 (Russian), *Diskret. Analiz.*, v. 34, 1980, pp. 38—73.
[13] Płonka, J., On algebras with n distinct essentially n-ary operations, *Algebra Universalis*, v. 1, 1971, pp. 73—79.
[14] Płonka, J., R-prime idempotent reducts of groups, *Arch. Math.*, v. 24, 1973, pp. 129—132.
[15] Post, E., *The two-valued iterative systems of mathematical logic*, Ann. of Math. Studies, No. 5, Princeton, 1941.
[16] Pöschel, R., L. A. Kalužnin, *Funktionen- und Relationenalgebren*, DVW, Berlin, 1979.
[17] Rosenberg, I. G., Über die funktionale Vollständigkeit in den mehrwertigen Logiken, *Rozpravy Československé Akad. Věd Ser. Math. Nat. Sci.* v. 80(4), 1970, pp. 3—93.
[18] Świerczkowski, S., Algebras which are independently generated by every n elements, *Fund. Math.*, v. 49, 1960, pp. 93—104.
[19] Takasaki, M., Abstraction of symmetric transformations (Japanese), *Tôhoku Math. J.*, v. 49, 1943, pp. 145—207.
[20] Werner, H., *Discriminator-Algebras*, Akademie-Verlag, Berlin, 1978.

(Received Dec. 12, 1982)

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

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

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

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

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

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

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

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

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.

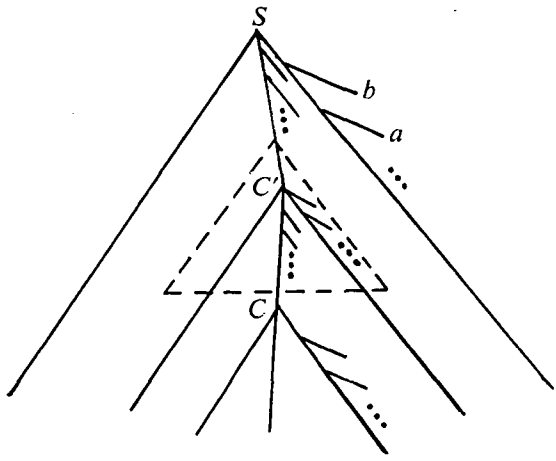


Fig. 9.1

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)

On the complexity of graph grammars

By GY. TURÁN

1. Introduction

Graph grammars generalize "usual" word grammars by considering graphs as basic objects instead of words. A derivation step consists of the replacement of a subgraph by another graph. The delicate part of the definition of graph grammars is the way the embedding of the new graph is specified (in the string case this problem does not appear).

This generalization appears to be quite natural and it has applications, too (Nagl [4]). However "nice" results of formal language theory characterizing classes of languages from different aspects (grammars, automata, algebraic and logical descriptions) does not seem to generalize for the case of graphs. (There are two possible explanations: either the right definitions are not found yet and one should not only try to generalize notions of string grammars, or the situation is indeed different.)

Another problematic aspect of graph grammars is the parsing of graph grammars (this is the topic we are going to discuss so we return to it later).

There are several theorems in graph theory describing a class of graphs as the class obtainable from a set of start graphs applying a finite set of operations (e.g. the theorem of Tutte on 3-connected graphs [7]). These theorems can actually be considered as positive results about graph grammars. Understanding the power of these operations could be of interest for graph theory as well. To return the problem of right definitions we remark that interesting operations of graph theory (e.g. Hajós' operations to generate non-k-colourable graphs [1]) quite often do not fit into the present framework of graph grammars.

As to our knowledge there are few results about the parsing of graph grammars. Grammars investigated are usually generalizations of context-free grammars, thus it is natural to try to generalize context-free parsing for the case of graphs. In the paper of Vigna—Ghezzi [8] it is shown that a certain parsing technique is exponential for their class of grammars and polynomial if further restrictions are imposed. Slisenko [6] gives a rather restricted class of grammars that can be parsed in polynomial time by a similar method (in fact he shows more: Hamiltonian cycles can be found in polynomial time when restricted to a context-free graph language). Results

of Janssens and Rozenberg [2] show that parsing their *node label controlled* (NLC) grammars is as hard as context-sensitive recognition.

One can ask the following questions about the parsing of graph grammars:
— what general parsing techniques exist?
— where is the borderline between "easy" and "hard" classes of grammars?

The theorem proved in this paper gives a step towards answering the second question. We introduce a natural restriction of NLC grammars by requiring graphs on the right-hand side of productions to consist of more than one vertex. Languages generated by these grammars are always in NP. We show that these grammars are strong enough to generate NP-complete languages. Thus no efficient parsing technique can be expected that is applicable for monotone NLC grammars.

2. Monotone node label controlled grammars

Following Janssens and Rozenberg [2] we define node label controlled (NLC) graph grammars as follows.

Definition. An NLC grammar $\mathcal{G}$ is a quintuple

$$(\Sigma, \Delta, G_0, \mathcal{P}, \mathcal{C})$$

where Σ is the (finite, nonempty) nonterminal alphabet; Δ is the (finite, nonempty) terminal alphabet (disjoint from Σ); G_0 is the start graph; $\mathcal{P}$ is the set of productions: a production P is a pair (a_i, G_i) where $a_i \in \Sigma$, G_i is a graph; $\mathcal{C}$ is the connection relation: $\mathcal{C} \subseteq (\Sigma \cup \Delta) \times (\Sigma \cup \Delta)$.

REMARK. Graphs considered are undirected, without loops and multiple edges. Edges are unlabeled, vertices are labeled with labels from $\Sigma \cup \Delta$.

Derivations and the language $L(\mathcal{G})$ generated by $\mathcal{G}$ are only described informally (see [2] for exact definitions). When applying a production $P=(a_i, G_i)$ to a graph G, we replace a vertex v_1 labeled with a_i by a graph isomorphic to G_i. If v_2 is a neighbour of v_1 in G labeled a_2 and v_3 is a vertex of G_i labeled a_3 then in the new graph G' v_2 and v_3 will be connected if and only if $(a_3, a_2) \in \mathcal{C}$ (see Fig. 1). Thus the embedding is controlled by the node labels only. $L(\mathcal{G})$ consists of graphs derivable from G_0 with all vertex labels belonging to Δ.

Definition. A *monotone* NCL *grammar* is an NLC grammar satisfying the following condition:

For every production $P=(a_i, G_i)$ the number of vertices of G_i is more than one.

This is the class of graph grammars we consider from the point of view of the complexity of languages generated.

3. The complexity of monotone NLC grammars

Proposition. If $\mathcal{G}$ is a monotone NLC grammar, than $L(\mathcal{G})$ is in NP (where, as usual, NP denotes the class of languages recognizable by nondeterministic Turing-machines in polynomial time).

Proof. As for every production $P=(a_i, G_i)$ G_i consists of more than one vertex, if a graph G has a derivation in $\mathcal{G}$ then the length of the derivation is at most $n-1$ where n is the number of vertices of G. Thus the derivation can be guessed and checked in polynomial time. □

Theorem. There exists a monotone NLC grammar $\mathcal{G}$ such that $L(\mathcal{G})$ is NP-complete.

Before turning to the proof we describe a property of graphs that will be used later on.

A graph $G=(V, E)$ has *cyclic bandwidth* $\leqq k$ if there exists a cyclic ordering $(v_1, \ldots, v_n)$ of the vertices s.t. if $(v_i, v_j)\in E$ then the cyclic distance of v_i and v_j is at most k. (The cyclic distance of v_i and v_j $(i<j)$ is $\min(j-i, n+i-j)$.)

The class of graphs with cyclic bandwidth $\leqq k$ is denoted by CB_k (here graphs are considered without vertex labels). The following result is mentioned in Johnson [3].

Theorem (Leung—Vornberger—Withoff). CB_2 is NP-complete. □

This result can be compared with the complexity of bandwidth (i.e. considering orders instead of cyclic orders): for any fixed k it can be decided in polynomial time whether the bandwidth of a graph is $\leqq k$ (Saxe [5]).

Now we give an informal description of our construction. By G_n we denote the graph on n vertices consisting of a cycle of length n and edges connecting vertices of distance 2 on the cycle. G_7 is shown on Fig. 2.

Every graph G on n vertices with cyclic bandwidth $\leqq 2$ is a subgraph of G_n. Thus G can be constructed by building G_n and then deleting the edges of G_n not belonging to G. G_n can be constructed by building a chain (shown on Fig. 3) and then closing the chain.

However, in order to be able to close the chain generated by an NLC grammar the edges connecting the "open" end of the chain with the "beginning" of the chain must always be present during the derivation and edges unnecessary after closing the chain must be forced to be deleted. These requirements can be fulfilled by defining the connection relation appropriately.

We remark that the grammar G used to prove the theorem is a rather large one, we did not try to make it as small as possible. Instead, we tried to make it easy to describe and analyze.

Proof of the theorem. First we describe the grammar $\mathcal{G}$ generating an NP-complete language.

The description of $\mathcal{G}$.

1) The nonterminal alphabet.

$$\Sigma = \{S, A_1, A_2, A_3, A_4, A_1', A_2', A_3', A_4'\} \cup \bigcup_{i=1}^{6} \mathcal{H}_i,$$

where

$$\mathcal{H}_i = \{B_i, C_i, D_i, E_i\} \cup \{C_i^{jk}, D_i^{jkm}, E_i^{jklm} : 0 \leqq j, k, l, m \leqq 1\}.$$

2) *The terminal alphabet.* $\Delta = \{x, y, z\}$.

3) *The start graph.* $G_0 = S$.

4) *The productions.* There are five groups of productions each playing different roles in the construction.

4.a) *Starting productions.* These productions can be applied at most once in every derivation and exactly one of them must be used in every derivation as a first step.

Consider the graph of Fig. 4 with 5 marked edges. Deleting all different subsets of these edges we get 32 graphs $H_1, \ldots, H_{32}$. The starting productions are of the form

$$S \Rightarrow H_i \quad \text{for} \quad i = 1, \ldots, 32.$$

4.b) *Chain-constructing productions.*

$$B_i \quad C_i \quad B_{i+1} \qquad \text{for} \quad i = 1, \ldots, 6.$$

Here and everywhere else in the construction addition and subtraction is meant cyclically, e.g. $6+1=1$. Using the productions belonging to this group a chain of arbitrary length can be generated. We use the following terminology: such a production *relabels the vertex labeled* B_i *by* C_i *and adds a new vertex labeled* B_{i+1}.

4.c) *Chain-closing productions.*

$$B_i \quad D_i \quad E_{i+1} \qquad \text{for} \quad i = 1, \ldots, 6.$$

The role of these productions is to close the chain generated by applying productions belonging to the previous group. Informally such a production relabels the vertex labeled B_i by D_i and adds a new vertex labeled E_{i+1} that becomes the last vertex of the chain.

4.d) *Edge-deleting productions.*

$$C_i \quad C_i^{jk} \quad y$$

$$D_i \quad D_i^{jkm} \quad y$$

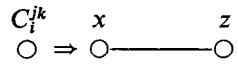
$$E_i \quad E_i^{jklm} \quad y \qquad \text{for} \quad i = 1, \ldots, 6, \quad 0 \leqq j, k, l, m \leqq 1.$$

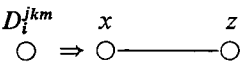
The role of these productions is to realize the deletion of edges. Vertex labeled C_i (resp. D_i, E_i) is relabeled C_i^{jk} (resp. D_i^{jkm}, E_i^{jklm}) and a new vertex labeled y is added. The binary vector (j, k, l, m) indicates the set of edges to be deleted (in G_n every vertex has degree 4).

$$A_i \quad A_i' \quad y \qquad \text{for} \quad i = 1, 2, 3, 4.$$

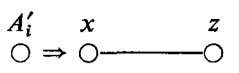
The role of these productions is to force the deletion of unnecessary edges if vertices labeled A_i "become terminal vertices too soon".

4.e) *Terminal productions.*





for $i = 1, \ldots, 6, \quad 0 \leqq j, k, l, m \leqq 1;$



for $i = 1, 2, 3, 4.$

5. *The connection relation.* As it is remarked already, the main regulating role in the derivation is played by the connection relation. Pairs belonging to the relation are divided into four groups.

5.a) *Pairs regulating the construction of the chain.*

$$(C_i, C_{i-1}),\ (C_i, C_{i-2}),\ (C_i, A_1),\ (C_i, A_2),$$

$$(B_i, C_{i-2}),\ (B_i, A_1),\ (B_i, A_2) \quad \text{for} \quad i = 1, \ldots, 6.$$

5.b) *Pairs regulating the closure of the chain.*

$$(D_i, C_{i-1}),\ (D_i, C_{i-2}),\ (D_i, A_1),\ (D_i, A_2)$$

$$(E_i, C_{i-2}),\ (E_i, A_1),\ (E_i, A_2) \quad \text{for} \quad i = 1, \ldots, 6.$$

5.c) *Pairs regulating the deletion of edges.*

$$(C_i^{jk}, N_{i+\delta}) \quad \text{for every} \quad N_{i+\delta} \in \mathscr{H}_{i+\delta}$$

if $(j = 1$ and $\delta = -2)$ or $(k = 1$ and $\delta = -1)$, or $(\delta = 1)$, or $(\delta = 2)$;

$$(D_i^{jkm}, N_{i+\delta}) \quad \text{for every} \quad N_{i+\delta} \in \mathscr{H}_{i+\delta}$$

if $(j = 1$ and $\delta = -2)$ or $(k = 1$ and $\delta = -1)$, or $(\delta = 1)$;

$$(E_i^{jklm}, N_{i+\delta}) \quad \text{for every} \quad N_{i+\delta} \in \mathscr{H}_{i+\delta}$$

if $(j = 1$ and $\delta = -2)$ or $(k = 1$ and $\delta = -1)$;

$$(C_1^{jk}, A_3),\ (C_1^{jk}, A_3') \quad \text{if} \quad j = 1;$$

$$(C_1^{jk}, A_4),\ (C_1^{jk}, A_4') \quad \text{if} \quad k = 1;$$

$$(C_2^{jk}, A_4),\ (C_2^{jk}, A_4') \quad \text{if} \quad j = 1;$$

$$(D_i^{jkm}, A_1),\ (D_i^{jkm}, A_1') \quad \text{if} \quad m = 1;$$

$$(E_i^{jklm}, A_1),\ (E_i^{jklm}, A_1') \quad \text{if} \quad l = 1;$$

$$(E_i^{jklm}, A_2),\ (E_i^{jklm}, A_2') \quad \text{if} \quad m = 1.$$

5.d) *Additional pairs*

$$(N, x) \quad \text{for every} \quad N \in \bigcup_{i=1}^{6} \mathcal{H}_i;$$

$$(x, M) \quad \text{for every} \quad M \in \Sigma \cup \Delta;$$

$$(A_i', x) \quad \text{for} \quad i = 1, 2, 3, 4;$$

$$(A_i', A_j) \quad \text{for} \quad 1 \leqq i, j \leqq 4.$$

Let G be an arbitrary graph without vertex labels. Define a graph G^* with vertices labeled x, y, z as follows:

1) label the vertices of G with x,
2) join two different vertices to each vertex of G and label them y and z respectively.

(An example is shown on Fig. 5.)

The theorem will be proved if we prove the following claim.

Claim. $L(\mathcal{G}) = \{G^*: G \in \mathrm{CB}_2$ and G has $\geqq 8$ vertices$\}$.

First we show the $\supseteq$ part of the claim.

Let $G \in \mathrm{CB}_2$ be a graph on $\geqq 8$ vertices. We describe a derivation of G^*.

Take a suitable circular order $(v_1, \ldots, v_n)$ of the vertices of G having circular bandwidth $\leqq 2$. Consider vertices $v_1, \ldots, v_7$ and label them $A_1, \ldots, A_4, C_1, C_2, B_3$. Take the subgraph spanned by $v_1, \ldots, v_7$ and add edges

$$(v_5, v_1),\ (v_5, v_2),\ (v_6, v_1),\ (v_6, v_2),\ (v_7, v_1),\ (v_7, v_2),$$

$$(v_5, v_6),\ (v_5, v_7),\ (v_6, v_7)$$

if they are not present yet.

The derivation of G^*.

1) Apply a suitable starting production to obtain the labeled graph on vertices $v_1, \ldots, v_7$ described above.

2) Apply chain-constructing productions $n-8$ times. (The applicable production is always unique.)

3) Apply a chain-closing production. (The applicable production is unique.)

4) For each vertex v_k, $5 \leqq k \leqq n$ define the binary vectors $\mathbf{v}_k := (i_{-2}^{(k)}, i_{-1}^{(k)}, i_1^{(k)}, i_2^{(k)})$; $\mathbf{v}_k' := (i_{-2}^{(k)}, i_{-1}^{(k)}, i_2^{(k)})$; $\mathbf{v}_k'' := (i_{-2}^{(k)}, i_{-1}^{(k)})$ where $i_j^{(k)} = 1 \Leftrightarrow (v_k, v_{k+j}) \in E$. For $k = n, n-1, \ldots, 5$ apply

$E_i \quad E_i^{\mathbf{v}_k} \quad y$
$\circ \Rightarrow \circ\!\!-\!\!-\!\!-\!\!-\!\!-\!\!-\!\!-\!\!-\!\!\circ$ if v_k is labeled E_i;

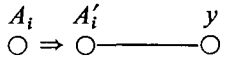
$D_i \quad D_i^{\mathbf{v}_k'} \quad y$
$\circ \Rightarrow \circ\!\!-\!\!-\!\!-\!\!-\!\!-\!\!-\!\!-\!\!-\!\!\circ$ if v_k is labeled D_i;

$C_i \quad C_{i_1}^{\mathbf{v}_k''} \quad y$
$\circ \Rightarrow \circ\!\!-\!\!-\!\!-\!\!-\!\!-\!\!-\!\!-\!\!-\!\!\circ$ if v_k is labeled C_i.

5) Apply terminal productions for each vertex labeled

$$E_i^{\mathbf{v}_k},\ D_i^{\mathbf{v}_k'} \quad \text{or} \quad C_i^{\mathbf{v}_k''}.$$

6) For $k=4, 3, 2, 1$ apply



7) Apply terminal productions for each vertex labeled A_i'.

After steps 1), 2), 3) we generated a labeled graph G' shown on Fig. 6, where $k \equiv n \pmod 6$. Vertices labeled A_1, A_2 are connected to every other vertex labeled C_i, D_i or E_i.

In step 4) unnecessary edges are deleted from vertices labeled C_i^{jk}, D_i^{jkm}, E_i^{jklm} and pendant vertices labeled y are added. As $A_i \notin \mathscr{H}_i$ and $A_i' \notin \mathscr{H}_i$, all edges connecting A_1 and A_2 to vertices labeled C_i^{jk} disappear and only the necessary edges connecting A_1 and A_2 to vertices labeled D_i^{jkm}, E_i^{jklm} remain in the graph. In step 5) vertices labeled C_i^{jk}, D_i^{jkm}, E_i^{jklm} are relabeled x and their adjacencies are not changed. In step 6) vertices labeled A_i are relabeled A_i' and pendant vertices labeled y are added. No change is made in this step in the edges, as edges between vertices labeled A_i and vertices outside the set of vertices labeled A_i are already disposed of, and internal edges are chosen correctly by the choice of the starting production. Finally in step 7) vertices labeled A_i originally get label x and pendant vertices labeled z are added.

Now we turn to the $\subseteq$ part of the claim.

Let G be a graph belonging to $L(G)$. We use the "relabeling" terminology introduced at the description of the productions. By the *history* of a vertex v we mean the sequence of labels appearing on v. The histories possible are the following.

1) B_i, C_i, C_i^{jk}, x for some i, j, k;

2) C_i, C_i^{jk}, x for $i = 1, 2$ and some j, k;

3) B_i, D_i, D_i^{jkm}, x for some i, j, k, m;

4) E_i, E_i^{jklm}, x for some i, j, k, l, m;

5) A_i, A_i', x for some i;

6) y;

7) z.

(The exceptional case 2 refers to the vertices labeled C_1, C_2 of the graph generated by the starting production.)

The graph generated (not considering vertex labels) will always be a subgraph of G_n'' of Fig. 7 for some n.

(The nonterminal label B_i can be replaced by C_i or D_i, and a new nonterminal B_{i+1} will appear in the graph unless B_i is replaced by D_i. Thus the generation of new vertices labeled B_i, C_i, D_i or E_i must end with a chain-closing production. Edge-deleting and terminal productions can be applied to nonterminals already present in the graph, thus making progress in the histories of each vertex, but these productions do not introduce new edges as $(y, .)$ and $(z, .)$ is not in the connection relation.)

The last point we have to check is that forbidden edges connecting vertices labeled originally A_1, A_2 and vertices ever labeled C_i do actually disappear. This can be shown considering the histories (C_i, C_i^{jk}, x) and (A_s, A_s', x) $(s=1, 2)$. There are no pairs (C_i^{jk}, A_s) or (A_s', C_i) in the connection relation so the edge (C_i, A_s) disappears whichever of the two histories makes progress first. The same holds for the pair (A_2, D_i).

Thus the second half of the claim is proved.

Finally it is obvious that CB_2 can be reduced to $L(\mathcal{G})$ in polynomial time by forming graphs G^*. □

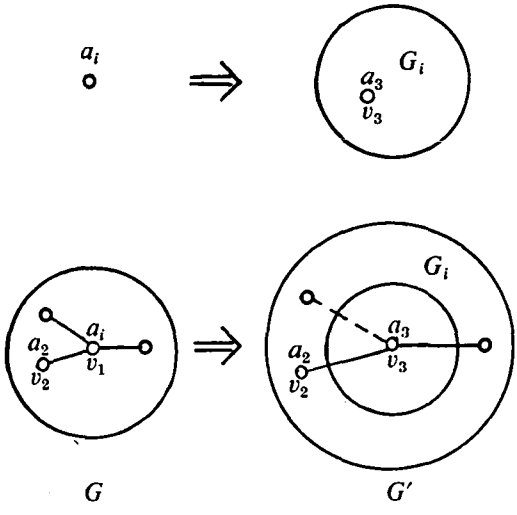


Fig. 1

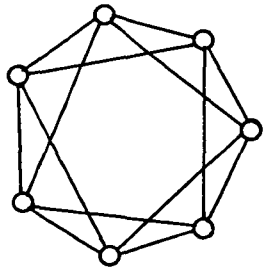


Fig. 2

Fig. 3

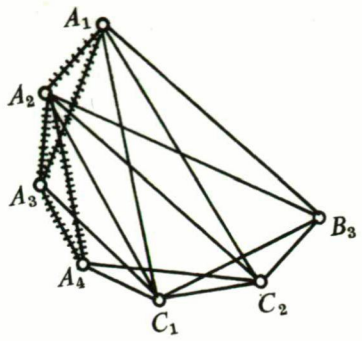


Fig. 4

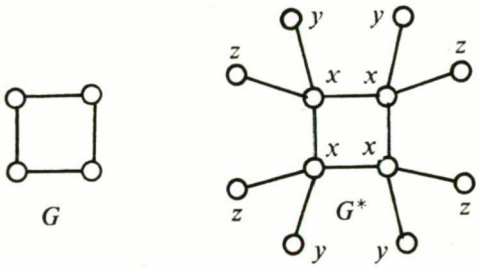


Fig. 5

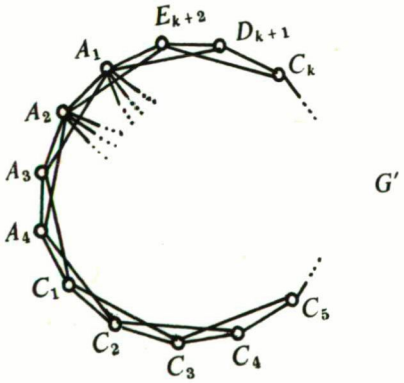


Fig. 6

G'

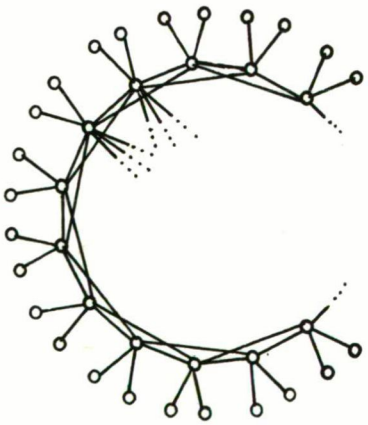


Fig. 7

Abstract

A problem in the theory of graph grammars is the following: for what classes of grammars can the languages generated be parsed in polynomial time? It is shown that a grammar belonging to a rather restricted class, the monotone node label controlled grammars can be strong enough to generate an NP-complete language.

RESEARCH GROUP ON THEORY OF AUTOMATA
HUNGARIAN ACADEMY OF SCIENCES
SOMOGYI U. 7
SZEGED, HUNGARY
H-6720

References

[1] HAJÓS, G., Über ein Konstruktion nicht *n*-färbbarer Graphen, *Wiss. Z. Martin-Luther Univ. Halle—Wittenberg Math.-Natur. Reihe*, v. 10, 1961, pp. 116—117.
[2] JANSSENS, D., G. ROZENBERG, Decision problems for node label controlled graph grammars, *J. Comput. System Sci.*, v. 22, 1981, pp. 147—177.
[3] JOHNSON, D. S., The NP-completeness column: an ongoing guide, *J. Algorithms*, v. 3, 1982, pp. 288—300.
[4] NAGL, M., Graph-Grammatiken, Theorie, Implementierung, Anwendungen, Vieweg, 1979.
[5] SAXE, J. B., Dynamic programming algorithms for recognizing small-bandwidth graphs in polynomial time, *SIAM J. Algebraic Discrete Methods*, v. 1, 1980, pp. 363—369.
[6] SLISENKO, A. O., Context-free grammars as a tool for describing polynomial-time subclasses of hard problems, *Inform. Process. Lett.*, v. 14, 1982, pp. 52—56.
[7] TUTTE, W. T., A theory of 3-connected graphs, *Indag. Math.*, v. 23, 1961, pp. 441—455.
[8] VIGNA, P. D., C. GHEZZI, Context-free graph grammars, *Inform. and Control.*, v. 37, 1978, pp. 207—233.

(Received Oct. 21, 1982)

General products and equational classes of automata

By Z. ÉSIK and F. GÉCSEG

The aim of this paper is to characterize those equational classes of automata which are obtained by means of the general product. It will be seen that such classes can be given by "patterns" of identities to be called p-identities. Moreover, these equational classes are either very large or very small.

1. Preliminaries

Let $\mathfrak{A}=(A, F, \delta)$ be an automaton, where A is the state set, F is the input set and δ is the next-state function of $\mathfrak{A}$. As it is well known $\mathfrak{A}$ can be considered an F-unoid (F-algebra with unary operational symbols) $\mathfrak{A}=(A, F)$ such that $af=\delta(a, f)$ $(a\in A, f\in F)$. Further on it will be supposed that F is finite. If A is also finite then we speak about a *finite* F-unoid.

In the sequel F and F' with or without indices will denote finite sets of unary operational symbols.

As usual F^* will stand for the free monoid freely generated by F. If $p=f_1 \dots f_k\in F^*$ is a word and x is a variable then xp is the F-polynomial symbol $(\dots(xf_1)\dots)f_k$.

Let K be a class of F-unoids. Then the operators $\mathbf{H}, \mathbf{S}$ and $\mathbf{P}$ on K are defined as follows:

$\mathbf{H}(K)$: homomorphic images of unoids from K,

$\mathbf{S}(K)$: subunoids of unoids from K,

$\mathbf{P}(K)$: direct products of nonvoid families of unoids from K.

By Birkhoff's Theorem (cf. [3]): For a nonvoid class K of F-unoids $\mathbf{HSP}(K)$ is the smallest equational class containing K.

Next we recall the concept of the products of automata (cf. [1]).

Let $\mathfrak{A}_i=(A_i, F_i)$ $(i\in I)$ be a non-void family of unoids, F a finite set of operational symbols and

$$\varphi\colon \Pi(A_i|i\in I)\times F \to \Pi(F_i|i\in I)$$

a mapping. Take the F-unoid $\mathfrak{A}=(A, F)$ with $A=\Pi(A_i|i\in I)$ and $\mathrm{pr}_i(\mathbf{a}f)=\mathrm{pr}_i(\mathbf{a})\,\mathrm{pr}_i(\varphi(\mathbf{a}, f))$ for arbitrary $\mathbf{a}\in A, f\in F$ and $i\in I$, where pr_i is the i^{th} projection. Then $\mathfrak{A}$ is the *(general) product* of $\mathfrak{A}_i$ $(i\in I)$ with respect to F and φ.

For arbitrary $\mathbf{a}\in A, f\in F$ and $i\in I$ let $\varphi_i(\mathbf{a}, f)$ be the i^{th} component of $\varphi(\mathbf{a}, f)$. If there exists a linear ordering $\leqq$ on I such that for every $i\in I$, φ_i is independent of its j^{th} component $(j\in I)$ whenever $j\geqq i$ then $\mathfrak{A}$ is an α_0-*product*. Obviously, if $F_i=F$ and $\varphi_i(\mathbf{a}, f)=f$ for arbitrary $i\in I$, $\mathbf{a}\in A$ and $f\in F$ then $\mathfrak{A}$ is the direct product of $\mathfrak{A}_i$ $(i\in I)$. Let us note that the formations of the product, the α_0-product and the direct product are transitive. Moreover, further on for α_0-products in $\varphi_i(\mathbf{a}, f)$ we shall indicate only those components on which φ_i may depend, i.e., f and $\mathrm{pr}_j(\mathbf{a})$ if $j<i$ $(j\in I)$.

Let K be a class of unoids (not necessarily of the same type). Then

$\mathbf{P}_g(K)$ is the class of all general products of unoids from K,

$\mathbf{P}_{f\alpha_0}(K)$ is the class of all α_0-products of unoids from K with finitely many factors, and

$\mathbf{K}_F$ is the similarity class of F-unoids.

To determine unoid identities preserved by products we recall the concept of an l-free system.

Take a unoid $\mathfrak{A}=(A, F)$, an element $a\in A$ and an integer $l\geqq 0$. The system $(\mathfrak{A}, a)$ is *l-free* if $ap\neq aq$ whenever $p\neq q$ and $|p|, |q|\leqq l$ $(p, q\in F^*)$, where $|p|$ denotes the length of p.

A state $a\in A$ is *ambiguous* if there are $f_1, f_2\in F$ such that $af_1\neq af_2$.

Obviously, every system $(\mathfrak{A}, a)$ is 0-free. Moreover, it easily follows from the proof of the Theorem in [2] that for a class K of unoids the following statements are equivalent:

(i) For an $l>0$ and all F there is an l-free system $(\mathfrak{A}, a)$ with $\mathfrak{A}=(A, F)\in \in\mathbf{P}_{f\alpha_0}(K)\cap K_F$,

(ii) K contains a $\mathfrak{B}=(B, F')$ such that for a $b\in B$ and a $p\in F'^*$ with $|p|=l-1$, bp is ambiguous. Therefore, if l is the greatest integer under which the above l-free system exists then for arbitrary $\mathfrak{B}=(B, F')\in K$, $b\in B$, $p\in F'^*$ with $|p|\geqq l$ and $f_1, f_2\in F'$, $bpf_1=bpf_2$.

2. Identities preserved by general products

Let K be an arbitrary nonvoid class of unoids. Then for every F, $\mathbf{HSP}_g(K)\cap K_F$ is an equational class since $\mathbf{HSP}_g(K)=\mathbf{HSPP}_g(K)$ obviously holds. Moreover, it is easy to show that $\mathbf{HSP}_g(K)$ is closed under the general product.

Now we introduce special identities to characterize $\mathbf{HSP}_g(K)\cap K_F$. A *p-identity* is

(i) $m=n$, or
(ii) $(k, m)=(k, n)$

where m, n and k are non-negative integers. A unoid $\mathfrak{A}=(A, F)$ *satisfies* p-identity (i) if $\mathfrak{A}$ satisfies all identities $xg_1\ldots g_m=yh_1\ldots h_n$ for arbitrary $g_1, \ldots, g_m, h_1, \ldots, h_n\in F$. Moreover, $\mathfrak{A}$ satisfies (ii) if it satisfies all identities $xf_1\ldots f_kg_1\ldots g_m= =xf_1\ldots f_kh_1\ldots h_n$ for arbitrary $f_1, \ldots, f_k, g_1, \ldots, g_m, h_1, \ldots, h_n\in F$. In these cases we also say that (i) or (ii) *holds* in $\mathfrak{A}$.

For a class K of unoids denote by K^* the class of all p-identities holding in every unoid from K. Moreover, K^{**} stands for the class of all unoids which satisfy every p-identity in K^*. Then we have the following

Theorem. For arbitrary F and nonvoid class K of unoids $\mathbf{HSP}_g(K)\cap K_F = K^{**}\cap K_F = \mathbf{HSPP}_{f\alpha_0}(K)\cap K_F$.

Proof. Obviously p-identities are preserved under general products. Thus $K^{**}\supseteq \mathbf{HSP}_g(K)$. Therefore, to prove the Theorem it is enough to show that $\mathbf{HSPP}_{f\alpha_0}(K)\cap K_F \supseteq K^{**}\cap K_F$ which follows from statements (i) and (ii) below.

(i) Let $xg_1\ldots g_m = yh_1\ldots h_n$ be an F-identity satisfied by $\mathbf{HSPP}_{f\alpha_0}(K)\cap K_F$. Then the p-identity $m=n$ is in K^*.

(ii) Let $xf_1\ldots f_k g_1\ldots g_m = xf_1\ldots f_k h_1\ldots h_n$ be an F-identity holding in $\mathbf{HSPP}_{f\alpha_0}(K)\cap \cap K_F$ such that $g_1\neq h_1$ if $m, n>0$. Then the p-identity $(k, m)=(k, n)$ is in K^*.

We shall prove (ii) only. Statement (i) can be shown in a similar way.

If for every l there are an $\mathfrak{A}=(A, F)\in \mathbf{P}_{f\alpha_0}(K)\cap K_F$ and an $a\in A$ such that $(\mathfrak{A}, a)$ is l-free then in $\mathbf{HSPP}_{f\alpha_0}(K)\cap K_F$ only the trivial identities hold. Therefore, $\mathbf{HSPP}_{f\alpha_0}(K)\supseteq K_F$.

Next assume that l is the greatest integer for which there exist an $\mathfrak{A}=(A, F)\in \in \mathbf{P}_{f\alpha_0}(K)\cap K_F$ and an $a\in A$ such that $(\mathfrak{A}, a)$ is l-free. Let the identity $xf_1\ldots f_k g_1\ldots \ldots g_m = xf_1\ldots f_k h_1\ldots h_n$ hold in $\mathbf{HSPP}_{f\alpha_0}(K)\cap K_F$ where $g_1\neq h_1$ if $m, n>0$. Suppose that the p-identity $(k, m)=(k, n)$ is not in K^*. Then we find a unoid $\mathfrak{A}'= =(A', F')\in K$, an element $a'\in A'$ and operational symbols $f_1', \ldots, f_k', g_1', \ldots \ldots, g_m', h_1', \ldots, h_n'\in F'$ under which

$$a'f_1'\ldots f_k'g_1'\ldots g_m' \neq a'f_1'\ldots f_k'h_1'\ldots h_n'.$$

Take the l-free system $(\mathfrak{A}, a)$ above, and form the α_0-product $\mathfrak{B}=(B, F)$ of $\mathfrak{A}$ and $\mathfrak{A}'$ given by the function $\varphi: A\times A'\times F\to F\times F'$ such that φ_1 is the identity mapping of F. Moreover,

$$\varphi_2(af_1\ldots f_i, f_{i+1}) = f_{i+1}' \quad \text{if} \quad i\leqq l,$$

$$\varphi_2(af_1\ldots f_k g_1\ldots g_i, g_{i+1}) = g_{i+1}' \quad \text{if} \quad k+i\leqq l$$

and

$$\varphi_2(af_1\ldots f_k h_1\ldots h_i, h_{i+1}) = h_{i+1}' \quad \text{if} \quad k+i\leqq l.$$

In all other cases φ_2 is defined arbitrarily. Then in $\mathfrak{B}$ we have

$$(a, a')f_1\ldots f_k g_1\ldots g_m = (af_1\ldots f_k g_1\ldots g_m, a'f_1'\ldots f_k'g_1'\ldots g_m') \neq$$

$$\neq (af_1\ldots f_k h_1\ldots h_n, a'f_1'\ldots f_k'h_1'\ldots h_n') = (a, a')f_1\ldots f_k h_1\ldots h_n,$$

which is a contradiction. This ends the proof of the Theorem.

Next we show that $\mathbf{HSP}_g(K)\cap K_F$ has a finite basis. As it has been noted if for arbitrary l and F' there are an $\mathfrak{A}=(A, F')\in\mathbf{HSP}_g(K)$ and an $a\in A$ such that $(\mathfrak{A}, a)$ is l-free then only the trivial identities hold in $\mathbf{HSP}_g(K)\cap K_F$. Thus we may assume that there exists such a maximal l which is also denoted by l.

(To check that the F-identities determined by the systems of p-identities below form a basis observe the existence of an l-free system $(\mathfrak{A}, a)$ with $\mathfrak{A}\in\mathbf{HSP}_g(K)\cap K_F$ such that for arbitrary $\mathfrak{B}=(B, F)\in\mathbf{HSP}_g(K)\cap K_F$ and $b\in B$ the mapping $\varphi: \{a\}\to\{b\}$ can be extended to a homomorphism of $\mathfrak{A}$ into $\mathfrak{B}$.)

I. K^* contains no p-identities of form $m=n$.

I. 1. There is a p-identity $(k, m)=(k, n)$ in K^* with $m<n$.

a) k_1 is minimal among all k occurring in p-identities $(k, m)=(k, n)$ from K^* with $m<n$,

b) m_1 is minimal among all m occurring in p-identities $(k_1, m)=(k_1, n)$ from K^* with $m<n$,

c) n_1 is minimal among all n occurring in p-identities $(k_1, m_1)=(k_1, n)$ from K^* with $m_1<n$,

d) k_2 is minimal among all k occurring in nontrivial* p-identities $(k, m)=$ $=(k, m)$ from K^*,

e) m_2 is minimal among all m occurring in nontrivial p-identities $(k_1, m)=$ $=(k_1, m)$ from K^*.

Then a suitable basis can be given in form

$$(k_1, m_1) = (k_1, n_1),\ (k_2^{(1)}, m_2^{(1)}) = (k_2^{(1)}, m_2^{(1)}), \ldots, (k_2^{(r)}, m_2^{(r)}) = (k_2^{(r)}, m_2^{(r)}),$$

where $k_2^{(1)}=k_2$, $m_2^{(1)}=m_2$, $k_2^{(1)}<\ldots<k_2^{(r)}<k_2+m_2$ and $k_2+m_2\geqq m_2^{(1)}>\ldots>m_2^{(r)}$. (Note that $k_1, k_2\leqq l$ and $k_1+n_1,\ k_2+m_2>l$.)

I. 2. K^* contains no p-identity $(k, m)=(k, n)$ with $m<n$. Then there is a basis of form $(k_2^{(1)}, m_2^{(1)})=(k_2^{(1)}, m_2^{(1)}), \ldots, (k_2^{(r)}, m_2^{(r)})=(k_2^{(r)}, m_2^{(r)})$ $(k_2^{(1)}=k_2,$ $m_2^{(1)}=m_2, k_2^{(1)}<\ldots<k_2^{(r)}<k_2+m_2,\ m_2^{(r)}<\ldots<m_2^{(1)}\leqq k_2+m_2)$ where k_2 and m_2 are obtained by d) and e) in I. 1.

II. K^* has a p-identity $m=n$.

Let m_1 be minimal among all m occurring in p-identities $m=n$ from K^*. Moreover let k_2 and m_2 be given by d) and e) in I. Then one of the bases has the form

$$m_1 = m_1,\ (k_2^{(1)}, m_2^{(1)}) = (k_2^{(1)}, m_2^{(1)}), \ldots, (k_2^{(r)}, m_2^{(r)}) = (k_2^{(r)}, m_2^{(r)}),$$

where again $k_2^{(1)}=k_2, m_2^{(1)}=m_2, k_2^{(1)}<\ldots<k_2^{(r)}<k_2+m_2$ and $k_2+m_2\geqq m_2^{(1)}>\ldots>m_2^{(r)}$.

If K consists of finitely many finite unoids then a finite basis can be given effectively. Therefore, for such a K and a finite $\mathfrak{A}=(A, F)$ it is decidable whether $\mathfrak{A}$ is contained by $\mathbf{HSP}_g(K)\cap K_F$.

Finally, it can be shown by a slight modification of the proof that the Theorem remains valid for infinite F, too.

DEPT. OF COMPUTER SCIENCE
A. JÓZSEF UNIVERSITY
ARADI VÉRTANÚK TERE 1
SZEGED, HUNGARY
H-6720

References

[1] GÉCSEG, F., Model theoretical methods in the theory of automata, Proceedings of the Symposium of Mathematical Foundations of Computer Science, High Tatras, 1973, pp. 57—63.

[2] GÉCSEG, F., Representation of automaton mappings in finite length, *Acta Cybernet.*, v. 2, 1975, pp. 285—289.

[1] Мальцев, А. И., *Алгебраические системы*, Москва, 1970.

(Received Jan. 14, 1983)

* A p-identity of form $(k, m)=(k, m)$ is *trivial* if $m=0$.

On identities preserved by general products of algebras

By Z. ÉSIK

Equational classes of automata (i.e. unoids) obtained by general product were characterized in [1]. Here we present similar results for tree automata, i.e., arbitrary algebras. We show that the main result $K^{**}=HSP_g(K)=HSP_{\alpha_0}(K)=HSPP_{f\alpha_0}(K)$ in [1] remains valid in this generality, too.

First we briefly introduce the basic notions to be used. For all unexplained notions coming from universal algebra and tree-automata theory the reader is referred to [3] and [2].

By a rank-type we mean an arbitrary subset R of the set of nonnegative integers. A type corresponding to a rank-type R is a collection of operational symbols $F=\biguplus(F_k|k\in\omega)$ such that $F_k\neq\emptyset$ if and only if $k\in R$. In the sequel we fix a ranktype R and by a type always mean a type corresponding to R.

Algebras of type F constitute a similarity class $\mathscr{K}_F$. An algebra $\mathfrak{A}\in\mathscr{K}_F$ is a pair $(A, \{f_{\mathfrak{A}}|f\in F\})$ — (A, F) for short —, where $f_{\mathfrak{A}}$ is a k-ary operation on the nonvoid set A for any $f\in F_k$. By a class of algebras we shall mean an arbitrary nonvoid class of algebras.

We are going to deal with certain products of algebras. Let I be a nonvoid set linearly ordered by $\leqq$. Given a system $\mathfrak{A}_i=(A_i, F_i)$ $(i\in I)$ of algebras, by a general product we mean an algebra $\mathfrak{A}=(A, F)=\Pi(\mathfrak{A}_i, \varphi|i\in I)$, where $A=\Pi(A_i|i\in I)$, φ is a family of mappings of $(\Pi(A_i|i\in I))^k\times F_k$ into $\Pi((F_i)_k|i\in I)$, and finally, the operations in $\mathfrak{A}$ are defined in accordence with φ as follows. Let $a_1, \ldots, a_k, a\in A, f\in F_k$. Then, $f_{\mathfrak{A}}(a_1, \ldots, a_k)=a$ if and only if $a_i=(f_i)_{\mathfrak{A}_i}(a_{1i}, \ldots, a_{ki})$ holds for every $i\in I$ with $f_i=(\varphi(a_1, \ldots, a_k, f))_i=\varphi_i\ (a_1, \ldots, a_k, f)$. If for every nonnegative integer k, $\varphi_i(a_1, \ldots, a_k, f)$ depends on f and $a_{1j}, \ldots, a_{kj}$ with $j<i$ only, then $\mathfrak{A}$ is a so called α_0-product of the $\mathfrak{A}_i$-s. We shall denote by P_g and P_{α_0} the operators corresponding to the formations of general and α_0-products, resp. $P_{f\alpha_0}$ will denote the formation of finite α_0-products. Finite α_0-products will be written as $\Pi(\mathfrak{A}_1, \ldots, \mathfrak{A}_n, \varphi)$ where $I=\{1, \ldots, n\}$ with the usual ordering. The operators H, S and P have their usual meaning.

Also we fix a countable set $X=\{x_1, x_2, \ldots\}$ of variables and treat polynomial symbols of type F as trees built on X and F. T_F will denote the set of all trees of type F. If $\mathfrak{A}\in\mathscr{K}_F$ and $p\in T_F$ then $p_{\mathfrak{A}}: A^\omega\to A$ is the polynomial induced by p in $\mathfrak{A}$. If $a_1, a_2, \ldots$ is an ω-sequence of elements of A then $p_{\mathfrak{A}}(a_1, a_2, \ldots)$ denotes the value of $p_{\mathfrak{A}}$ on $a_1, a_2, \ldots$. If $\mathfrak{A}$ is the general product described

previously then we can view φ as a mapping of $(\Pi(A_i|i\in I))^\omega \times T_F$ into $\Pi(T_{F_i}|i\in I)$ in a natural way. For each index $i\in I$ we shall denote by φ_i the i-th component-map of φ, as well.

The notion of subtrees of a tree p as well as the height $h(p)$ of a tree will be used in an unexplained but obvious way. A subtree q of a tree p is called proper if $q\neq p$. sub (p) denotes the set of all proper subtrees of p. Also we shall in a natural way speak about an occurence of a subtree in a tree, and about the substitution of a tree for occurences of a subtree in a tree. If p is a tree then rt (p) denotes the root of p.

By a relabeling we mean any mapping $\varphi: T_F \to T_{F'}$ with the following properties:

(i) if $p\in F_0$ then $\varphi(p)\in F'_0$,

(ii) if $p\in X$ then $\varphi(p)=p$,

(iii) if $p=f(p_1, ..., p_k)$ with $f\in F_k$, $k>0$, $p_1, ..., p_k\in T_F$ then there exist an $f'\in F'_k$ such that $\varphi(p)=f'(\varphi(p_1), ..., \varphi(p_k))$.

Now we are in the position to give the most fundamental definitions. Let K be an arbitrary class of algebras. Then $K^*=\{K_F^*|F$ is a type$\}$, where K_F^* is the set of all identities $p=q$ $(p, q\in T_F)$ such that $\varphi(p)=\varphi(q)$ is in the usual sence a valid identity in $K\cap\mathscr{K}_{F'}$ for any relabeling $\varphi: T_F\to T_{F'}$. An algebra $\mathfrak{A}\in\mathscr{K}_F$ is in K^{**} if and only if all identities belonging to K_F^* are valid in $\mathfrak{A}$. Thus, $K^{**}\cap\mathscr{K}_F$ is an equational class of algebras. If $p, q\in T_F$, we write $K^*\models p=q$ to mean that $K_F^*\models p=q$.

If we consider unoids, i.e. we take $R=\{1\}$, then for any type F and $p, q\in T_F$ we have $p=q\in K_F^*$ if and only if $p=q$ is valid in the equational class $HSP_{\alpha_0}(K)\cap\mathscr{K}_F$. Consequently, $K^{**}=HSP_{\alpha_0}(K)$, or even, $K^{**}=HSP_g(K)=HSP_{\alpha_0}(K)=HSPP_{f\alpha_0}(K)$ (cf. [1]).

In general, the first statement fails to hold. Indeed, take $R=\{1, 2\}$ and for every type F let $K\cap\mathscr{K}_F$ be the equational class determined by the identities $g(x_1)=h(x_1)$ $(g, h\in F_1)$. Supposing $f\in F_2$, identity $f(g(x_1), g(x_1))=f(h(x_1), h(x_1))$ is obviously valid in $HSP_{\alpha_0}(K)\cap\mathscr{K}_F$, but this identity is not in K_F^*. However, we still have a somewhat weaker result:

Theorem 1. Let $p, q\in T_F$ be arbitrary trees of type F. Then $p=q$ is a valid identity in an equational class $HSP_{\alpha_0}(K)\cap\mathscr{K}_F$ if and only if $K^*\models p=q$.

Proof. Sufficiency follows by observing that general product preserves K^*, that is, $P_g(K)\subseteq K^{**}$. Therefore, also $HSP_{\alpha_0}(K)\subseteq K^{**}$. In order to prove the necessity of our Theorem, let Σ contain those valid identities $p=q$ of the equational class $HSP_{\alpha_0}(K)\cap\mathscr{K}_F$ for which our statement does not hold. Supposing $\Sigma\neq\emptyset$, choose $p=q\in\Sigma$ in such a way that $|\text{sub}\,(p)\cup\text{sub}\,(q)|$ is minimal.

Now take an algebra $\mathfrak{A}=(A, F)$ freely generated by the sequence $a_1, a_2, ...$ in the equational class $HSP_{\alpha_0}(K)\cap\mathscr{K}_F$. First we show that if we have $r_{\mathfrak{A}}(a_1, a_2, ...)=s_{\mathfrak{A}}(a_1, a_2, ...)$ for some trees $r, s\in\text{sub}\,(p)\cup\text{sub}\,(q)$, then $r=s$, i.e., the trees r and s coincide. Assume to the contrary that there exist different trees $r, s\in\text{sub}\,(p)\cup\text{sub}\,(q)$ with $r_{\mathfrak{A}}(a_1, a_2, ...)=s_{\mathfrak{A}}(a_1, a_2, ...)$. Let us fix a tree $r\in\text{sub}\,(p)\cup\text{sub}\,(q)$ with the property that if $\bar{r}\in\text{sub}\,(p)\cup\text{sub}\,(q)$ and $\bar{r}_{\mathfrak{A}}(a_1, a_2, ...)=r_{\mathfrak{A}}(a_1, a_2, ...)$ then $h(r)\leqq h(\bar{r})$, and there is a distinct tree $s\in\text{sub}\,(p)\cup\text{sub}\,(q)$ with $r_{\mathfrak{A}}(a_1, a_2, ...)=s_{\mathfrak{A}}(a_1, a_2, ...)$. Given r, choose a different tree $s\in\text{sub}\,(p)\cup\text{sub}\,(q)$ such that $r_{\mathfrak{A}}(a_1, a_2, ...)=s_{\mathfrak{A}}(a_1, a_2, ...)$, and $h(\bar{s})\leqq h(s)$ whenever $\bar{s}\in\text{sub}\,(p)\cup\text{sub}\,(q)$ and

$s_{\mathfrak{A}}(a_1, a_2, \ldots) = \bar{s}_{\mathfrak{A}}(a_1, a_2, \ldots)$. Obviously, we have $h(r) \leqq h(s)$. If $s \in \mathrm{sub}(p)$ then let us substitute r for any occurrence of s in p, and denote the resulting tree by $\bar{p}$. If $s \notin \mathrm{sub}(p)$ then put $\bar{p} = p$. Similar procedure when applied to q will produce the tree $\bar{q}$. Of course we have $\mathrm{sub}(\bar{p}) \cup \mathrm{sub}(\bar{q}) \subseteq \mathrm{sub}(p) \cup \mathrm{sub}(q)$, or even, the choise of r and s garantees that $s \notin \mathrm{sub}(\bar{p}) \cup \mathrm{sub}(\bar{q})$. Thus, $|\mathrm{sub}(\bar{p}) \cup \cup \mathrm{sub}(\bar{q})| < |\mathrm{sub}(p) \cup \mathrm{sub}(q)|$. Similarly, $|\mathrm{sub}(r) \cup \mathrm{sub}(s)| < |\mathrm{sub}(p) \cup \mathrm{sub}(q)|$.

As $r_{\mathfrak{A}}(a_1, a_2, \ldots) = s_{\mathfrak{A}}(a_1, a_2, \ldots)$, it follows that $r = s$ is a valid identity in $HSP_{\alpha_0}(K) \cap \mathscr{K}_F$. As $|\mathrm{sub}(r) \cup \mathrm{sub}(s)| < |\mathrm{sub}(p) \cup \mathrm{sub}(q)|$ also $K^* \models r = s$. As $r = s$ is a valid identity in $HSP_{\alpha_0}(K) \cap \mathscr{K}_F$, also the equalities $p_{\mathfrak{A}}(a_1, a_2, \ldots) = \bar{p}_{\mathfrak{A}}(a_1, a_2, \ldots)$ and $q_{\mathfrak{A}}(a_1, a_2, \ldots) = \bar{q}_{\mathfrak{A}}(a_1, a_2, \ldots)$ are satisfied. Since $p = q$ was a valid identity in $HSP_{\alpha_0}(K) \cap \mathscr{K}_F$ and $\mathfrak{A}$ is freely generated by $a_1, a_2, \ldots$, also $\bar{p} = \bar{q}$ is a valid identity in $HSP_{\alpha_0}(K) \cap \mathscr{K}_F$. As $|\mathrm{sub}(\bar{p}) \cup \mathrm{sub}(\bar{q})| < |\mathrm{sub}(p) \cup \mathrm{sub}(q)|$, by the choise of the identity $p = q$, we obtain that $K^* \models \bar{p} = \bar{q}$. The construction of the trees $\bar{p}$ and $\bar{q}$ shows that $\{r = s, \bar{p} = \bar{q}\} \models p = q$. We have already seen that $K^* \models r = s$, thus, $K^* \models p = q$. This is a contradiction.

So far we have shown that the equality $r_{\mathfrak{A}}(a_1, a_2, \ldots) = s_{\mathfrak{A}}(a_1, a_2, \ldots)$ is satisfied by trees $r, s \in \mathrm{sub}(p) \cup \mathrm{sub}(q)$ if and only if $r = s$. Next we are going to prove that $p = q \in K_F^*$. As $K^* \models p = q$ holds in this case evidently, this would again be a contradiction.

Assume that $p = q \notin K_F^*$. Then there is a type F' and a relabeling $\varphi\colon T_F \to T_{F'}$ such that $\varphi(p) = \varphi(q)$ is not a valid indentity in the class $K \cap \mathscr{K}_{F'}$. Therefore, there is an algebra $\mathfrak{B} = (B, F') \in K$ and elements $b_1, b_2, \ldots \in B$ with

$$\varphi(p)_{\mathfrak{B}}(b_1, b_2, \ldots) \neq \varphi(q)_{\mathfrak{B}}(b_1, b_2, \ldots).$$

Let $\mathfrak{C} = (C, F)$ be any α_0-product $\Pi(\mathfrak{A}, \mathfrak{B}, \psi)$ with ψ satisfying the following conditions for every $f \in F_k$ $(k \geqq 0)$:

(i) $\psi_1(f) = f$,

(ii) $\psi_2((p_1)_{\mathfrak{A}}(a_1, a_2, \ldots), \ldots (p_k)_{\mathfrak{A}}(a_1, a_2, \ldots), f) = \mathrm{rt}(\varphi(f(p_1, \ldots, p_k)))$

if $f(p_1, \ldots, p_k)$ is a subtree of p or q.

In order to show that such an α_0-product exists, it is enough to see that whenever both $f(p_1, \ldots, p_k)$ and $f(q_1, \ldots, q_k)$ are subtrees of p or q and $(p_i)_{\mathfrak{A}}(a_1, a_2, \ldots) = (q_i)_{\mathfrak{A}}(a_1, a_2, \ldots)$ $(i = 1, \ldots, k)$ then $\mathrm{rt}(\varphi(f(p_1, \ldots, p_k))) = \mathrm{rt}(\varphi(f(q_1, \ldots q_k)))$. But this can be seen immediately as φ is a mapping and $(p_i)_{\mathfrak{A}}(a_1, a_2, \ldots) = (q_i)_{\mathfrak{A}}(a_1, a_2, \ldots)$ implies that $p_i = q_i$.

As $HSP_{\alpha_0}(K)$ is closed under α_0-products, we get $\mathfrak{C} \in HSP_{\alpha_0}(K) \cap \mathscr{K}_F$. On the other hand, $\varphi(p)_{\mathfrak{B}}(b_1, b_2, \ldots) \neq \varphi(q)_{\mathfrak{B}}(b_1, b_2, \ldots)$ implies that $p_{\mathfrak{C}}((a_1, b_1), (a_2, b_2), \ldots) \neq q_{\mathfrak{C}}((a_1, b_1), (a_2, b_2), \ldots)$, contrary to our assumption that $p = q$ is valid in $HSP_{\alpha_0}(K) \cap \mathscr{K}_F$.

A set of identities $\Delta \subseteq T_F^2$ is called closed if whenever $\Delta \models p = q$ is valid for trees $p, q \in T_F$ then $p = q \in \Delta$. It is known from universal algebra that Δ is closed if and only if the following five conditions are satisfied by Δ:

(i) $x_i = x_i \in \Delta$ $(i = 1, 2, \ldots)$,

(ii) $p = q \in \Delta$ implies that $q = p \in \Delta$,

(iii) $p = q$, $q = r \in \Delta$ implies that $p = r \in \Delta$,

(iv) if $p_i=q_i\in\Delta$ for all $i=1, \ldots, k$ $(k\geqq 0)$ and $f\in F_k$ then $f(p_1, \ldots, p_k)=f(q_1, \ldots, q_k)\in\Delta$,

(v) if $p=q\in\Delta$ and we get p' and q' from p and q by substituting all occurences of a variable x_i by an arbitrary tree $r\in T_F$ then $p'=q'\in\Delta$.

By virtue of the previous Theorem, if K_F^* is closed for every type F, then whenever $p=q$ is a valid identity in an equational class $HSP_{\alpha_0}(K)\cap\mathscr{K}_F$ then $p=q\in K_F^*$. Conversely, if $p=q\in K_F^*$ then $p=q$ is a valid identity in $HSP_{\alpha_0}(K)\cap\mathscr{K}_F$. As K_F^* always satisfies conditions from (i) to (v) above except (iv), a necessary and sufficient condition for K_F^* to be closed is to satisfy condition (iv). In this way we get the following

Corollary. Assume that K_F^* satisfies condition (iv) for every type F. Then an identity $p=q$ is valid in an equational class $HSP_{\alpha_0}(K)\cap\mathscr{K}_F$ if and only if $p=q\in K_F^*$. Conversely, if we have the equivalence $p=q$ is valid in an equational class $HSP_{\alpha_0}(K)\cap\mathscr{K}_F$ if and only if $p=q\in K_F^*$ then K_F^* satisfies condition (iv).

Further on we shall need the following

Lemma. Let $\mathfrak{A}=(A, F)=\Pi(\mathfrak{A}_i, \varphi)|i\in I)$ be an arbitrary infinite α_0-product of algebras $\mathfrak{A}_i=(A_i, F_i)$ and let $J\subseteq I$ and $T\subseteq T_F$ be finite sets. For every sequence $a_1, a_2, \ldots\in A$ there is a finite α_0-product $\mathfrak{B}=(B, F)=\Pi(\mathfrak{A}_i, \psi|i\in J_1)$ with $J\subseteq J_1$ and such that $\psi_i(a_{1J_1}, a_{2J_1}, \ldots, p)=\varphi_i(a_1, a_2, \ldots, p)$ for any $p\in T$ and $i\in J$.[1]

Proof. Put $h=\max\{h(p)|p\in T\}$. If $h=0$ then our statement is obviously valid. We proceed by induction on h. Let $h>0$ and assume that the proof is done for $h-1$. For every $k>0$ and $f\in F_k$ set

$$U_f=\{p|p\in T,\quad h(p)=h,\quad \mathrm{rt}\,(p)=f\},$$

$$V_f=\{p|p\in\bigcup(\mathrm{sub}\,(q)|q\in T)\cup T,\quad \mathrm{rt}\,(p)=f\}.$$

Let $(p, q, i)\in U_f\times V_f\times J$ — say $p=f(p_1, \ldots, p_k)$, $q=f(q_1, \ldots, q_k)$ — be arbitrary. If $\varphi_i(p_{1\mathfrak{A}}(a_1, a_2, \ldots), \ldots, p_{k\mathfrak{A}}(a_1, a_2, \ldots), f)\neq\varphi_i(q_{1\mathfrak{A}}(a_1, a_2, \ldots), \ldots, q_{k\mathfrak{A}}(a_1, a_2, \ldots), f)$ then choose an index $i_0<i$ with $(p_{t\mathfrak{A}}(a_1, a_2, \ldots))_{i_0}\neq(q_{t\mathfrak{A}}(a_1, a_2, \ldots))_{i_0}$ for some $t\in\{1, \ldots, k\}$. Denote by I_0 the set of indices obtained in this way, and put $J'=J\cup I_0$, $T'=\bigcup(\mathrm{sub}\,(p)|p\in T)\cup\{p\in T|h(p)<h\}$. By the induction hypothesis, there exist a finite set J_1' and an α_0-product $\mathfrak{B}'=(B', F)=\Pi(\mathfrak{A}_i, \psi'|i\in J_1')$ with $J'\subseteq J_1'$ and satisfying $\psi_i'(a_{1_{J_1'}}, a_{2_{J_1'}}, \ldots, p)=\varphi_i(a_1, a_2, \ldots, p)$ for each $p\in T'$ and $i\in J_1'$.

Set $J_1=J_1'$ and define the α_0-product $\mathfrak{B}=(B, F)=\Pi(\mathfrak{A}_i, \psi|i\in J_1)$ so that the following two conditions are satisfied:

(i) $\psi(b_1, \ldots, b_k, f)=\psi'(b_1, \ldots, b_k, f)$ if $f\in F_k$ $(k\geqq 0)$ and there exist trees $p_1, \ldots, p_k\in T_F$ with $f(p_1, \ldots, p_k)\in T'$ and $b_t=p_{t\mathfrak{B}'}(a_{1_{J_1}}, a_{2_{J_1}}, \ldots)$ $(t=1, \ldots, k)$,

(ii) $\psi_i(b_1, \ldots, b_k, f)=\varphi_i(c_1, \ldots, c_k, f)$ if $i\in I$, $f\in F_k$ $(k>0)$, and there exist trees $p_1, \ldots, p_k\in T_F$ with $f(p_1, \ldots, p_k)\in U_f$ and $b_t=p_{t\mathfrak{B}'}(a_{1_{J_1}}, a_{2_{J_1}}, \ldots)$, $c_t=p_{t\mathfrak{A}}(a_1, a_2, \ldots)$ $(t=1, \ldots, k)$.

[1] The ordering on J_1 is the restriction of the ordering on I to J_1. If $a\in\Pi(A_i|i\in I)$ then $a_{J_1}\in\Pi(A_i|i\in J_1)$ is determined by $(a_{J_1})_i=a_i$ for any $i\in J_1$.

Such an α_0-product exist, since otherwise we would have an index $i \in I$ together with distinct trees $p=f(p_1, \ldots, p_k) \in U_f$, $q=f(q_1, \ldots, q_k) \in V_f$ $(f \in F_k, k>0, p_t, q_t \in T_F)$ such that $(p_{t\varrho_I}(a_1, a_2, \ldots))_j = (q_{t\varrho_I}(a_1, a_2, \ldots))_j$ $(t=1, \ldots, k)$ for all $j \prec i$ but $\varphi_i(p_{1\varrho_I}(a_1, a_2, \ldots), \ldots, p_{k\varrho_I}(a_1, a_2, \ldots), f) \neq \varphi_i(q_{1\varrho_I}(a_1, a_2, \ldots), \ldots, q_{k\varrho_I}(a_1, a_2, \ldots), f)$. Also the equalities $\psi_i(a_{1_{J_1}}, a_{2_{J_1}}, \ldots, p) = \varphi_i(a_1, a_2, \ldots, p)$ $(i \in I, p \in T)$ follow in an easy way.

Theorem 2. $HSPP_{f\alpha_0}(K) = HSP_{\alpha_0}(K) = HSP_g(K) = K^{**}$ holds for any class K of algebras.

Proof. The last two equalities immediately follow by Theorem 1 and Birkhoff's Theorem. $HSPP_{f\alpha_0}(K) \subseteq HSP_{\alpha_0}(K)$ is obvious. We claim that also $HSP_{\alpha_0}(K) \subseteq$ $\subseteq HSPP_{f\alpha_0}(K)$. This can be seen by showing that if F is an arbitrary type and an identity $p=q$ $(p, q \in T_F)$ is not valid in $P_{\alpha_0}(K) \cap \mathscr{K}_F$ then the same holds for $P_{f\alpha_0}(K) \cap \mathscr{K}_F$. But this is a trivial consequence of our Lemma.

Theorem 2 is in a close connection with the characterization theorem of metrically complete systems of algebras in [2]. It turns out that a system K of algebras having finite types is metrically complete if and only if K^* contains only trivial identities. In other words this means that K is complete (that is, $HSP_g(K)$ is the class of all algebras) if and only if K is metrically complete.

DEPT. OF COMPUTER SCIENCE
A. JÓZSEF UNIVERSITY
ARADI VÉRTANÚK TERE 1
SZEGED, HUNGARY
H-6720

References

[1] ÉSIK, Z. and F. GÉCSEG, General products and equational classes of automata, *Acta Cybernet.* v. 6, 1983, pp. 281—284.
[2] GÉCSEG, F., On a representation of deterministic frontier-to-root tree transformations, *Acta Sci. Math. (Szeged)*, v. 45, 1983, pp. 177—187.
[3] GRÄTZER, G., *Universal algebra,* D. Van Nostrand Company, 1968.

(Received Jan. 18, 1983)

Deterministic ascending tree automata II

By J. VIRÁGH

To the memory of my Mother

In [12] we started a systematic study of deterministic ascending (called also root-to-frontier or top-down) tree automata. The present second part is entirely devoted to the investigation of the product of such automata. We generalize the notion of the product of ordinary automata due to Gluskov [cf. 7] and that of the special products defined by Gécseg in [3]. Some other generalizations can be found in [9], [10] and [11] for the case of bottom-up (known also as frontier-to-root) tree automata.

1. Preliminaries

The reader is assumed to be familiar with the fundamental concepts concerning tree automata and tree transducers. To keep the size of the paper within reasonable limits we give only a brief account on notions defined elsewhere but used in our treatment, too. For terminology not defined here, see [1], [5] and/or [6].

The concepts of a *type* F, a *deterministic ascending* F*-algebra* $\mathfrak{A}=\langle A, F\rangle$, a *deterministic ascending* F*-automaton* $\mathbf{A}=\langle\mathfrak{A}, a', \mathbf{a}\rangle$ and *the forest* $T(\mathbf{A})\subseteq T_{F,X_n}$ *recognized by* $\mathbf{A}$ are used in the same sense as in [12]. In the sequel F-algebra (F-automaton) means a deterministic ascending F-algebra (F-automaton). When F is not specified we speak simply about algebra and automaton. Furthermore, all algebras and automata are assumed to be finite and have no nullary operations.

Now we shall introduce some additional terminology. $|A|$ denotes the cardinality of the set A. A *rank type* R is a finite nonvoid subset of the set $N=\{1, 2, \ldots\}$ of natural numbers. The type F has rank type $R(F)=\{n|F_n\neq\varnothing\}$.

Let $\mathbf{A}=\langle\mathfrak{A}, a', \mathbf{a}\rangle$, $\mathbf{a}=(A^{(1)}, \ldots, A^{(n)})$ and $\mathbf{B}=\langle\mathfrak{B}, b', \mathbf{b}\rangle$, $\mathbf{b}=(B^{(1)}, \ldots, B^{(n)})$ be two F-automata with the associated algebras $\mathfrak{A}=\langle A, F\rangle$ and $\mathfrak{B}=\langle B, F\rangle$. Then $\mathfrak{B}$ is called a *subalgebra of* $\mathfrak{A}$ if $B\subseteq A$ and for all $k\in R(F)$, $f\in F_k$ and $b\in B$, $f^{\mathfrak{A}}(b)=f^{\mathfrak{B}}(b)\in B^k$ holds. The automaton $\mathbf{A}$ is *connected* if all states $a\in A$ are reachable from the initial state a' by suitable operations. (For a formal description see [5].)

Next we recall some concepts and results from [5].

A *homomorphism of the algebra* $\mathfrak{A}$ into $\mathfrak{B}$ is a mapping $\varphi: A \to B$ such that

(i) for all $k \in R(F)$, $f \in F_k$ and $a \in A$, $f^{\mathfrak{B}}(\varphi(a)) = (\varphi(a_1); \ldots; \varphi(a_k))$, where $(a_1, \ldots, a_k) = f^{\mathfrak{A}}(a)$. If, in addition

(ii) $\varphi(a') = b'$ and

(iii) for all $i = 1, \ldots, n$; $\varphi(A_i) = B_i$ and $\varphi^{-1}(B_i) = A_i$ hold, then φ is a *homomorphism of the automaton* $\mathbf{A}$ into $\mathbf{B}$. In case of $\varphi(A) = B$ we call $\mathfrak{B}$ a *homomorphic image* of $\mathfrak{A}$. If φ is also bijective then it is called an *isomorphism*. We say that $\mathfrak{A}$ and $\mathfrak{B}$ are *isomorphic* and write $\mathfrak{A} \cong \mathfrak{B}$ if there exists an isomorphism $\varphi: \mathfrak{A} \to \mathfrak{B}$. The same terminology is used for automata.

A *congruence relation of the algebra* $\mathfrak{A}$ is defined as an equivalence relation ϱ on A such that

(i) for all $k \in R(F)$, $f \in F_k$ and $a, a' \in A$, $a \varrho a'$ implies $a_i \varrho a_i'$ where $i = 1, \ldots, k$, $f^{\mathfrak{A}}(a) = (a_1, \ldots, a_k)$ and $f^{\mathfrak{A}}(a') = (a_1', \ldots, a_k')$.

Moreover, ϱ is a *congruence relation of the automaton* $\mathbf{A}$ if the additional condition

(ii) for all $i = 1, \ldots, n$ and $a \in A$, $a \in A^{(i)}$ implies $\varrho(a) \subseteq A^{(i)}$

holds.

NOTATION. The two trivial congruences of $\mathfrak{A}$ will be denoted by $\iota = A \times A$ and $\omega = \{(a, a) | a \in A\}$, respectively. $\mathfrak{A}$ is *simple* if it has only trivial congruences.

For any state $a \in A$ let $\mathbf{A}(a)$ denote the automaton $\langle \mathfrak{A}, a, \mathbf{a} \rangle$. The state a is called a 0-*state* if $T(\mathbf{A}(a)) = \varnothing$. We say that $\mathbf{A}$ is *normalized* if, for all $k \in R(F)$, $f \in F_k$ and $a \in A$, either all of the components of $f^{\mathfrak{A}}(a)$ are 0-states or none of them is a 0-state. The automaton $\mathbf{A}$ is *minimal* if $|A| \leqq |B|$ whenever $T(\mathbf{A}) = T(\mathbf{B})$.

The following results are from [5].

Proposition 1.1. If $\mathbf{B}$ is a homomorphic image of $\mathbf{A}$, then $T(\mathbf{A}) = T(\mathbf{B})$.

Proposition 1.2. If the minimal automaton $\mathbf{A}$ is equivalent to the normalized and connected automaton $\mathbf{B}$ then $\mathbf{A}$ is a homomorphic image of $\mathbf{B}$.

In the rest of this paper we consider only algebras belonging to the class $K(R)$ of all finite algebras of the fixed rank type R. Let $F, F^1, \ldots, F^k$ be ranked alphabets of rank type R and consider the F^i-algebras $\mathfrak{A}_i = (A_i, F^i)$ $(i = 1, \ldots, k)$. Furthermore, let

$$\psi: A_1 \times \ldots \times A_k \times F \to F^1 \times \ldots \times F^k$$

be an arity-preserving mapping, i.e., for every $m \in R$, $f \in F_m$ and $\mathbf{a} \in \prod_{i=1}^{k} A_i$, $\psi(\mathbf{a}, f) = (f^1, \ldots, f^k)$ implies $f^i \in F_m^i$ $(i = 1, \ldots, k)$. Then by the *general product* or, shortly *G-product* of $\mathfrak{A}_1, \ldots, \mathfrak{A}_k$ with respect to the *feedback function* ψ we mean the F-algebra $\mathfrak{A} = \langle A, F \rangle = \prod_{i=1}^{k} \mathfrak{A}_i[F, \psi]$ with $A = \prod_{i=1}^{k} A_i$ and for arbitrary $m \in R$, $f \in F_m$ and $\mathbf{a} \in A$

$$f^{\mathfrak{A}}(\mathbf{a}) = \big((\pi_1(f^1(\pi_1(\mathbf{a}))), \ldots, \pi_1(f^k(\pi_k(\mathbf{a})))), \ldots$$

$$\ldots(\pi_m(f^1(\pi_1(\mathbf{a}))), \ldots, \pi_m(f^k(\pi_k(\mathbf{a}))))\big),$$

where $(f^1, \ldots, f^k) = \psi(\mathbf{a}, f)$ and π_l denotes the l^{th} projection.

To define special types of products let us write ψ in the form $\psi=(\psi^{(1)}, \ldots, \psi^{(k)})$, where for arbitrary $\mathbf{a}\in A$ and $f\in F_m$, $\psi(\mathbf{a}, f)=(\psi^{(1)}(\mathbf{a}, f), \ldots, \psi^{(k)}(\mathbf{a}, f))$. We say that $\mathfrak{A}$ is an α_i-*product* $(i=0, 1, \ldots)$ if for arbitrary j $(1\leqq j\leqq k)$ $\psi^{(j)}$ is independent of its u^{th} component if $i+j\leqq u\leqq k$. If ψ is independent of $\prod_{i=1}^{k} A_i$, i.e., ψ is a mapping of F into $\prod_{i=1}^{k} F^i$, then $\mathfrak{A}$ is a *quasidirect-product* (shortly Q-product).

Let θ-product mean any of the α_i-products, the Q-product or the G-product. Now take a class K of algebras. Then $H_\theta(K)$ denotes the class of all algebras which can be given as homomorphic images of subalgebras of θ-products of algebras from K. Similarly, $I_\theta(K)$ stands for the class consisting of all algebras which are isomorphic to subalgebras of θ-products of algebras from K. The class K is *homomorphically (isomorphically)* complete with respect to the θ-product if $H_\theta(K)=K(R)$ $(I_\theta(K)=K(R))$ holds. Finally, K is *forest complete* with respect to the θ-product if for every forest $T\subseteqq T_{F, X_n}$ recognizable by deterministic ascending automata there exists a θ-product $\mathfrak{A}=\langle A, F\rangle$ of algebras from K and an automaton $\mathbf{A}=\langle\mathfrak{A}, a', \mathbf{a}\rangle$ satisfying $T(\mathbf{A})=T$.

2. Some general properties of the products

It is obvious that every isomorphically θ-complete system is homomorphically θ-complete as well. For the converse we note

Remark 2.1. For every θ there exists a homomorphically θ-complete system $M\subseteqq K(R)$ which is not isomorphically θ-complete.

To verify this statement take an arbitrary isomorphically θ-complete system M. Since $I_\theta(M)$ contains the one-element algebras there is an $\mathfrak{A}=\langle A, G\rangle\in M$ and an $a\in A$ such that

$(*)$ for all $r\in r(G)$ there is a $g\in G_r$ satisfying $g(a)=(a, \ldots, a)$

holds. Now take the system $M^*=\{\mathfrak{A}^*|\mathfrak{A}\in M\}$ where

$$\mathfrak{A}^*=\langle A\cup B, G\rangle, \quad B=\{a^*|a\in A \text{ and } a \text{ satisfies } (*)\}.$$

The operations of $\mathfrak{A}^*$ are defined for all $g\in G$, $a\in A$ and $a^*\in B$ in the following way: $g^{\mathfrak{A}^*}(a^*)=g^{\mathfrak{A}}(a)$ and

$$g^{\mathfrak{A}^*}(a)=\begin{cases} g^{\mathfrak{A}}(a) & \text{if } g^{\mathfrak{A}}(a)\neq(a, \ldots, a) \\ (a^*, \ldots, a^*) & \text{otherwise.}\end{cases}$$

Evidently, M^* cannot be isomorphically θ-complete but it will turn out to be homomorphically θ-complete.

Let $\mathfrak{C}=\langle C, H\rangle$ be an arbitrary algebra. Assume that $\mathfrak{C}$ is isomorphic to a subalgebra $\mathfrak{D}$ of the θ-product $\prod_{i=1}^{k}\mathfrak{A}_i[H, \psi]$ from M. Constructing the θ-product $\prod_{i=1}^{k}\mathfrak{A}_i^*[H, \psi^*]$ from M^* it is not difficult to prove that $\mathfrak{C}$ is a homomorphic

image of a subalgebra $\mathfrak{D}^*$ of this product. Here ψ^* is defined by $\psi^*(\mathbf{a}, h)=\psi(\hat{\mathbf{a}}, h)$ where $\hat{\mathbf{a}}$ can be obtained 'by removing the stars', i.e., if $\pi_i(\mathbf{a})\in A$ then $\pi_i(\hat{\mathbf{a}})=\pi_i(\mathbf{a})$ else if $\pi_i(\mathbf{a})=a^*\in B$ then $\pi_i(\hat{\mathbf{a}})=a$ for all $i=1, \ldots, k$.

For ordinary automata the notion of the completeness with respect to the automaton mappings have been introduced. (See, e.g. [4].) Now we shall define a similar concept concerning 'tree automaton mappings', i.e., top-down tree transformations.

In the sequel we shall use the general terms such as *top-down tree transducers* and *top-down tree transformations* induced by them, *deterministic, connected* or *minimal transducers* in their usual meaning (c.f. [1], [2] or [6]).

A top-down tree transducer $\mathscr{A}=\langle T_{F,X_n}, A, T_{G,Y_m}, A', \Sigma_{\mathscr{A}}\rangle$ is *uniform* if each rule $af\to p(a\in A, f\in F_l, l\in R(F), p\in T_{G,Y_m\cup A\Xi_l})$ can be written in the form $af\to q(a_1\xi_1, \ldots, a_l\xi_l)$ for some $q\in T_{G,Y_m\cup\Xi_l}$. In this section by a transducer $\mathscr{A}$ we shall mean a deterministic uniform top-down tree transducer having exactly one rule $af\to p$ for every $(a,f)\in A\times F$. Moreover, all transducers are assumed to have the fixed input rank type R.

Let $\mathscr{A}=\langle T_{F,X_n}, A, T_{G,Y_m}, a', \Sigma_{\mathscr{A}}\rangle$ and $\mathscr{B}=\langle T_{F,X_n}, B, T_{G,Y_m}, b', \Sigma_{\mathscr{B}}\rangle$ be transducers and take a mapping $\varphi: A\to B$. If the following three conditions are satisfied for arbitrary $af\to q(a_1\xi_1, \ldots, a_l\xi_l)$ and $ax_i\to t\in\Sigma_{\mathscr{A}}$ then φ is called a *homomorphism* of $\mathscr{A}$ into $\mathscr{B}$

(i) if $af\to q(a_1\xi_1, \ldots, a_l\xi_l)\in\Sigma_{\mathscr{A}}$ then
$bf\to q(b_1\xi_1, \ldots, b_l\xi_l)\in\Sigma_{\mathscr{B}}$ where
$b=\varphi(a)$, $b_j=\varphi(a_j)$ $(j=1, \ldots, l)$,
(ii) if $ax_i\to t\in\Sigma_{\mathscr{A}}$ then
$bx_i\to t\in\Sigma_{\mathscr{B}}$ where
$b=\varphi(a)$,
(iii) $\varphi(a')=\varphi(b')$.

If φ is surjective then $\mathscr{B}$ is a *homomorphic image* of $\mathscr{A}$.

The following result has been obtained in [2].

Proposition 2.2. If there is a homomorphism from $\mathscr{A}$ into $\mathscr{B}$ then $\tau_{\mathscr{A}}=\tau_{\mathscr{B}}$.

The n-ary F-automaton $\mathbf{A}=\langle\langle A, F\rangle, a', \mathbf{a}\rangle$ *belongs to the* transducer $\mathscr{A}=\langle T_{F,X_n}, A, T_{G,Y_m}, a', \Sigma_{\mathscr{A}}\rangle$ if

(i) for all $a\in A$, $k\in R(F)$ and $f\in F_k$, $f^{\mathfrak{A}}(a)=(a_1, \ldots, a_k)$ implies
$af\to p(a_1\xi_1, \ldots, a_k\xi_k)\in\Sigma_{\mathscr{A}}$ for some $p\in T_{G,Y_m\cup\Xi_k}$ and
(ii) for $1\leqq i\leqq n$, $a\in A^{(i)}$ iff $ax_i\to q\in\Sigma_{\mathscr{A}}$ for some $q\in T_{G,Y_m}$.

$\mathrm{Aut}(\mathscr{A})$ denotes the class of all automata belonging to $\mathscr{A}$. Now we can introduce the class $\mathrm{Alg}(\mathscr{A})$ of all algebras belonging to $\mathscr{A}$:

$\mathfrak{A}=\langle A, F\rangle\in\mathrm{Alg}(\mathscr{A})$ iff there is an automaton $\mathbf{A}=\langle\mathfrak{A}, a', \mathbf{a}\rangle\in\mathrm{Aut}(\mathscr{A})$.

A system $M\subseteq K(R)$ is *complete* with respect to the θ-product if for every tree transformation $\tau: T_{F,X_n}\to T_{G,Y_m}$ there is a transducer $\mathscr{A}$ and a θ-product $\mathfrak{A}$ of algebras from M such that $\tau=\tau_{\mathscr{A}}$ and $\mathfrak{A}\in\mathrm{Alg}(\mathscr{A})$ hold.

In the proof of the following theorem we need the concept of the paths of a tree p. For arbitrary type F, $n\in N$ and $p\in T_{F,X_n}$, $\mathrm{path}(p)$ stands for the smallest subset of $(F\times N)^*$ satisfying

(i) if $p=x_i$ $(1\leqq i\leqq n)$ then $\mathrm{path}(p)$ consist of the empty word e, and

(ii) if $p=f(p_1, \ldots, p_k)$, $k\in R$, $f\in F_k$, $p_1, \ldots, p_k\in T_{F,X_n}$ then

$$\text{path}(p)=\bigcup_{i=1}^{k}(f, i)\,\text{path}(p_i).$$

Moreover, for arbitrary set $T\subseteq T_{F,X_n}$ define $\text{path}(T)=\bigcup(\text{path}(t)|t\in T)$. The *realization of the path* $v\in\text{path}(T_{F,X_n})$ in the F-algebra $\mathfrak{A}=\langle A, F\rangle$ is the mapping $v^{\mathfrak{A}}: A\to A$ given by

(i) $ae^{\mathfrak{A}}=a$ for all $a\in A$ and

(ii) $av^{\mathfrak{A}}=c$ iff $v=u(f, i)$, $au^{\mathfrak{A}}=b$ and $\pi_i(f(b))=c$ holds for $u\in\text{path}(T_{F,X_n})$, $k\in R, f\in F_k$, $1\leqq i\leqq k$ and $b\in A$.

Theorem 2.3. With respect to arbitrary θ-product the homomorphic completeness, the completeness and the forest completeness are equivalent to each other.

Proof. (1) Let the system $M\subseteq K(R)$ be homomorphically complete with respect to the θ-product. Further, let τ be an arbitrary transformation induced by the connected transducer $\mathscr{A}=\langle T_{F,X_n}, A, T_{G,Y_m}, a', \Sigma_{\mathscr{A}}\rangle$ and let $\mathbf{A}=\langle\mathfrak{A}, a', \mathbf{a}\rangle\in\in\text{Aut}(\mathscr{A})$.

As M is homomorphically θ-complete there exist a θ-product $\mathfrak{C}=\langle C, F\rangle=\prod_{i=1}^{s}\mathfrak{C}_i[F, \psi]$ from M and a subalgebra $\overline{\mathfrak{C}}$ of $\mathfrak{C}$ such that $\mathfrak{A}$ is a homomorphic image of $\overline{\mathfrak{C}}$ under some homomorphism φ. Taking the subalgebra $\mathfrak{C}^*$ of $\overline{\mathfrak{C}}$ generated by a $c'\in\varphi^{-1}(a')$ it follows easily that $\mathbf{A}$ is a homomorphic image of the connected automaton $\mathbf{C}^*=\langle\mathfrak{C}^*, c', \mathbf{c}\rangle$ under φ. (The final state vector $\mathbf{c}$ of $\mathbf{C}^*$ can be given using the inverse of φ.) Let us consider the transducer $\mathscr{C}^*=\langle T_{F,X_n}, C^*, T_{G,Y_m}, c', \Sigma_{\mathscr{C}^*}\rangle$ satisfying

(i) $cf\to p(c_1\xi_1, \ldots, c_k\xi_k)\in\Sigma_{\mathscr{C}^*}$ iff $\varphi(c)=a$, $\varphi_i(c_i)=a_i$ $(i=1, \ldots, k)$ and $af\to p(a_1\xi_1, \ldots, a_k\xi_k)\in\Sigma_{\mathscr{A}}$,

(ii) $cx_i\to q\in\Sigma_{\mathscr{C}^*}$ iff $\varphi(c)=b$ and $bx_i\to q\in\Sigma_{\mathscr{A}}$ for all $f\in F$, $c\in C^*$ and $1\leqq i\leqq n$.

This construction ensures that $\varphi: \mathscr{C}^*\to\mathscr{A}$ is a homomorphism. Hence, by Proposition 2.2, $\tau_{\mathscr{C}^*}=\tau_{\mathscr{A}}$. But taking the transducer $\mathscr{C}=\langle T_{F,X_n}, C, T_{G,Y_m}, c', \Sigma_{\mathscr{C}}\rangle$ where

$$\Sigma_{\mathscr{C}}=\Sigma_{\mathscr{C}^*}\cup(cf\to q_{c,f}(c_1\xi_1, \ldots, c_k\xi_k)|c\notin C^*, f\in F, f^{\mathfrak{C}}(c)=(c_1, \ldots, c_k)),$$

and $q_{c,f}$ is an arbitrary tree from $T_{G,Y_m}\cup\Xi_k)$ it is obvious that $\tau_{\mathscr{C}}=\tau_{\mathscr{C}^*}=\tau$ and $\mathfrak{C}\in\text{Alg}(\mathscr{C})$ which proves the θ-completeness of M.

(2) Now let M be a θ-complete system. Take an arbitrary algebra $\mathfrak{A}=\langle A, F\rangle$ with $A=(a_0, a_1, \ldots, a_{n-1})$. Without loss of generality we may assume $n>1$. Choose a $j\in R$ and construct the algebra $\hat{\mathfrak{A}}=\langle A, G\rangle$ with $G_i=F_i$ if $i\neq j$ and $G_j=F_j\cup\{h\}$ where h is a new operational symbol. For all $g\in G$ and $a_i\in A$ realize g such that

$$g^{\hat{\mathfrak{A}}}(a_i)=\begin{cases}\underbrace{(a_{i+1(\text{mod}\,n)}, \cdots, a_{i+1(\text{mod}\,n)})}_{j\text{ times}} & \text{if } g=h \text{ and}\\ g^{\mathfrak{A}}(a_i) & \text{otherwise.}\end{cases}$$

Define the associated transducer $\hat{\mathscr{A}}=(T_{G,X_1}, A, T_{G,Y_n}, a_0, \Sigma_{\hat{\mathscr{A}}})$ by the following rules for all $g\in G$ and $a_i\in A$

(i) $a_i g \to g(a_1\xi_1, \ldots, a_k\xi_k) \in \Sigma_{\hat{\mathscr{A}}}$ iff
$g^{\hat{\mathfrak{A}}}(a_i) = (a_1, \ldots, a_k)$ and
(ii) $a_i x_1 \to y_{i+1} \in \Sigma_{\hat{\mathscr{A}}}$.

The θ-completeness of M ensures that there exists a θ-product $\mathfrak{B} = \prod_{i=1}^{s} \mathfrak{B}_i[G, \psi]$ from M and a transducer $\mathscr{B} = \langle T_{G,X_1}, B, T_{G,Y_n}, b_0, \Sigma_{\mathscr{B}} \rangle$ equivalent to $\hat{\mathscr{A}}$ such that $\mathfrak{B} \in \mathrm{Alg}(\mathscr{B})$. Take the connected subtransducer $\mathscr{B}^* = \langle T_{G,X_1}, B^*, T_{G,Y_n}, b_0, \Sigma_{\mathscr{B}^*} \rangle$ of $\mathscr{B}$ and the corresponding connected subalgebra $\mathfrak{B}^* = \langle B^*, G \rangle$ of $\mathfrak{B}$. Now we are going to prove that $\hat{\mathfrak{A}}$ is a homomorphic image of this $\mathfrak{B}^*$.

To this end define the correspondence $\varphi: B^* \to A$ by $\varphi(b_0 u^{\mathfrak{B}^*}) = a_0 u^{\hat{\mathfrak{A}}}$ for every $u \in \mathrm{path}(T_{G,X_1})$. Since $\mathfrak{B}^*$ and $\hat{\mathfrak{A}}$ are connected φ is defined for all $b \in B^*$ and $\varphi(B^*) = A$. We claim that φ is a well defined mapping, i.e. $b = b_0 u^{\mathfrak{B}^*} = b_0 v^{\mathfrak{B}^*}$ implies $a_0 u^{\hat{\mathfrak{A}}} = a_0 v^{\hat{\mathfrak{A}}}$ for all $u, v \in \mathrm{path}(T_{G,X_1})$. Assume to the contrary that there are $u, v \in \mathrm{path}(T_{G,X_1})$ such that $b = b_0 u^{\mathfrak{B}^*} = b_0 v^{\mathfrak{B}^*}$ and $a_i = a u^{\hat{\mathfrak{A}}} \neq a_0 v^{\hat{\mathfrak{A}}} = a_j$.

The realization of h ensures the existence of trees $p, q \in T_{G,X_1}$ with the following properties

(i) $u \in \mathrm{path}(p)$, $v \in \mathrm{path}(q)$,
(ii) if $z \in \mathrm{path}(p)$ then $a_0 z^{\hat{\mathfrak{A}}} = a_i$ and
if $w \in \mathrm{path}(q)$ then $a_0 w^{\hat{\mathfrak{A}}} = a_j$.

Then we have

(iii) $\mathrm{fr}(\tau_{\hat{\mathscr{A}}}(p)) \in \{y_{i+1}\}^*$, $\mathrm{fr}(\tau_{\hat{\mathscr{A}}}(q)) \in \{y_{j+1}\}^*$.

Taking two arbitrary trees $\bar{p}, \bar{q} \in T_{G,X_1}$ with $u \in \mathrm{path}(\bar{p})$ and $v \in \mathrm{path}(\bar{q})$ we can construct the trees p, q satisfying (ii) by substituting the leaves of $\bar{p}$ and $\bar{q}$ by suitable trees from $T_{\{h\},X_1}$.

From the equivalence of $\hat{\mathscr{A}}$ and $\mathscr{B}^*$ it follows easily that the transducer $\mathscr{B}^*$ is nondeleting. This, by property (iii) means that during the translation of p in $\mathscr{B}^*$ we have to apply some production $bx_1 \to t$ where $\mathrm{fr}(t) \in \{y_{i+1}\}^*$. On the other hand, the translation of q requires a production $bx_1 \to \bar{t}$ with $\mathrm{fr}(\bar{t}) \in \{y_{j+1}\}^*$. Hence, by the assumption $a_i \neq a_j$ the contradiction $bx_1 \to t$, $bx_1 \to \bar{t} \in \Sigma_{\mathscr{B}^*}$ and $t \neq \bar{t}$ follows.

At last, by the definition of φ

$$f(\varphi(b)) = f(a_0 v^{\hat{\mathfrak{A}}}) = (a_0 v(f,1)^{\hat{\mathfrak{A}}}, \ldots, a_0 v(f,k)^{\hat{\mathfrak{A}}}) =$$
$$= (\varphi(b_0 v(f,1)^{\mathfrak{B}^*}), \ldots, \varphi(b_0 v(f,k)^{\mathfrak{B}^*})) = \varphi(f(b)) \quad \text{holds}$$

for all $v \in \mathrm{path}(T_{G,X_1})$, $b = b_0 v^{\mathfrak{B}^*} \in B^*$, $k \in R$ and $f \in F_k$ proving that φ is a homomorphism. Now it is evident that $\hat{\mathfrak{A}}$ is a homomorphic image under φ of the subalgebra $\bar{\mathfrak{B}}^* = \langle B^*, F \rangle$ of the θ-product $\bar{\mathfrak{B}} = \prod_{i=1}^{s} \mathfrak{B}_i[F, \bar{\psi}]$ where $\bar{\psi}$ is the restriction of ψ to $\prod_{i=1}^{s} B_i \times F$.

(3) It is quite obvious that every homomorphically θ-complete system M is forest complete with respect to the θ-product as well (cf. Proposition 1.1).

(4) At last, assume that M is a forest complete system with respect to the θ-product. Take an arbitrary algebra $\mathfrak{A} = \langle A, F \rangle$ with $A = \{a_0, \ldots, a_{n-1}\}$. Choosing

a new operational symbol h and proceeding in the same way as in case (2) construct the algebra $\hat{\mathfrak{A}}=\langle A, G\rangle$. The definition of h ensures that the automaton $\hat{\mathbf{A}}=\langle\hat{\mathfrak{A}}, a_0, \{a_0\}\rangle$ is connected and normalized. Moreover, by the proof of Theorem 8 in [5] $\hat{\mathbf{A}}$ is a minimal automaton since it has no two different equivalent states.

The forest completeness of M implies the existence of an automaton $\mathbf{C}=\langle\mathfrak{C}, c', \mathbf{c}\rangle$ equivalent to $\hat{\mathbf{A}}$ where $\mathfrak{C}=\prod_{i=1}^{t}\mathfrak{C}_i[G, \psi]$ is a θ-product from M. As the realization of h results that every connected automaton equivalent to $\hat{\mathbf{A}}$ is normalized and, even more, it has no 0-states, the connected subautomaton $\mathbf{C}^*=\langle\mathfrak{C}^*, c', \mathbf{c}^*\rangle$ of $\mathbf{C}$ is normalized, too. Therefore, by Proposition 1.2 the minimal automaton $\hat{\mathbf{A}}$ is a homomorphic image of $\mathbf{C}^*$. Now it is trivial that omitting h the algebra $\mathfrak{A}=\langle A, F\rangle$ is a homomorphic image of the subalgebra $\overline{\mathfrak{C}}^*=\langle C^*, F\rangle$ of the θ-product $\overline{\mathfrak{C}}=\prod_{i=1}^{t}\mathfrak{C}_i[F, \bar{\psi}]$, where $\bar{\psi}$ is the restriction of the feedback function ψ to $\prod_{i=1}^{t}C_i\times F$. □

3. Complete systems with respect to some special types of products

In this section we shall investigate the isomorphically θ-complete systems if $\theta=Q, \alpha_0$ and G, and derive some properties of the homomorphically G-complete systems as well.

For the sake of brevity let us introduce the relation $\mathfrak{A}\prec_\theta\mathfrak{B}$ iff $\mathfrak{A}$ can be isomorphically embedded into a θ-product of $\mathfrak{B}$ with a single factor. When $\theta=Q$ we have

Theorem 3.1. A system $K\subseteq K(R)$ is isomorphically complete with respect to the quasidirect product iff for every simple algebra $\mathfrak{A}$ there is a $\mathfrak{B}\in K$ such that $\mathfrak{A}\prec_Q\mathfrak{B}$ holds.

Proof. The sufficiency of the condition can easily be derived from the transitivity of the relation $\prec_Q$ and from the following assumption

(*) For an arbitrary algebra $\mathfrak{C}$ a simple algebra $\mathfrak{A}$ satisfying $\mathfrak{C}\prec_Q\mathfrak{A}$ can be constructed.

To verify (*), take the algebra $\mathfrak{C}=\langle C, F\rangle$, $C=\{c_0, \ldots, c_k\}$. We define the algebra $\mathfrak{A}=\langle A, G\rangle$ as follows. The base set of $\mathfrak{A}$ is the disjoint union $A=C\cup\{c_{k+1}, \ldots, c_{p-1}\}$, where p is an arbitrary prime number with $p-1>k$. Suppose that $j\in R$. In this case let $G_i=F_i$ for all $i\neq j$ and $G_j=F_j\cup\{h\}$ where h is a new operational symbol. The realization of the operations in $\mathfrak{A}$ is given by

$$g^{\mathfrak{A}}(c_i)=\begin{cases} g^{\mathfrak{C}}(c_i) & \text{if } g\in F \text{ and } 0\leqq i\leqq k,\\ \underbrace{(c_i, \ldots, c_i)}_{n \text{ times}} & \text{if } n\in R,\ g\in F_n \text{ and } k<i\leqq p-1,\\ \underbrace{(c_{i+1(\mathrm{mod}\,p)}, \cdots, c_{i+1(\mathrm{mod}\,p)})}_{j \text{ times}} & \text{if } g=h \text{ and } 0\leqq i\leqq p-1.\end{cases}$$

The introduction of the new operation h guarantees the simplicity of $\mathfrak{A}$. $\mathfrak{C} \prec_Q \mathfrak{A}$ follows evidently by considering the product $\mathfrak{A}[F, \psi]$ with the feedback function $\psi(f)=f$ for every $f \in F$.

Conversely, let K be isomorphically Q-complete. Hence for arbitrary simple algebra $\mathfrak{A}=\langle A, F\rangle$ there is a Q-product $\mathfrak{B}=\prod_{i=1}^{k} \mathfrak{B}_i[F, \psi]$ from K such that $\mathfrak{A}$ is isomorphic to a subalgebra of $\mathfrak{B}$. Let φ denote a suitable isomorphism. Now we can introduce the relations ϱ_i $(1 \leqq i \leqq k)$ on A in the following manner:

$$a \varrho_i b \quad \text{iff} \quad \pi_j(\varphi(a)) = \pi_j(\varphi(b)) \quad \text{for all} \quad 1 \leqq j \leqq i.$$

The fact that $\mathfrak{B}$ is a Q-product yields that all the ϱ_i are congruence relations and

$$\iota \supseteq \varrho_1 \supseteq \ldots \supseteq \varrho_k = \omega.$$

As $\mathfrak{A}$ is simple, all this relations are trivial, i.e., there is a natural number m $(1 \leqq m \leqq k)$ such that

$$\varrho_m = \omega$$

holds. Now we proceed to show that in this case $\mathfrak{A} \prec_Q \mathfrak{B}_m$. Take the Q-product $\mathfrak{C}=\langle B_m, F\rangle=\mathfrak{B}_m[F, \xi]$ with the feedback function $\xi(f)=\pi_m(\psi(f))$ for all $f \in F$. It can immediately be shown that the mapping $\eta: A \to B_m$ defined by $\eta(a)=\pi_m(\varphi(a))$ for all $a \in A$ is an isomorphic embedding of $\mathfrak{A}$ into $\mathfrak{B}_m$. The choice of m ensures the injectivity of η. On the other hand, η is the composition of the m^{th} projection with the isomorphism φ, hence η must be a homomorphism.

Corollary 3.2. There exists no minimal isomorphically Q-complete system of algebras.

Proof. Take an isomorphically Q-complete system $M \subseteqq K(R)$ and an arbitrary $\mathfrak{C}$ from M. We shall verify that the system $M_1=M-\{\mathfrak{C}\}$ satisfies the conditions of Theorem 3.1 as well. Let $\mathfrak{B}$ be a simple algebra. From the isomorphic completeness of M it follows that $\mathfrak{B} \prec_Q \mathfrak{A}$ holds for some $\mathfrak{A} \in M$. Now we claim that $\mathfrak{B} \prec_Q \overline{\mathfrak{A}}$ holds for some $\overline{\mathfrak{A}} \in M_1$, too. We distinguish the following two cases

(1) If $\mathfrak{A} \neq \mathfrak{C}$, then we put $\overline{\mathfrak{A}}=\mathfrak{A}$.

(2) In the case of $\mathfrak{A}=\mathfrak{C}$ we can, by assumption $(*)$, construct a simple algebra $\mathfrak{D}$ with $|\mathfrak{C}|<|\mathfrak{D}|$ and $\mathfrak{C} \prec_Q \mathfrak{D}$. But M is isomorphically Q-complete thus it contains an algebra $\mathfrak{E}$ satisfying $\mathfrak{D} \prec_Q \mathfrak{E}$. Of course, $\mathfrak{A} \neq \mathfrak{E}$ hence $\mathfrak{E} \in M_1$ and the transitivity of $\prec_Q$ implies $\mathfrak{B} \prec_Q \mathfrak{E}$. □

In the case of α_0-products we can state similar results.

Theorem 3.3. A system $K \subseteqq K(R)$ is isomorphically complete with respect to the α_0-product iff for every simple algebra $\mathfrak{A}$ there is a $\mathfrak{B} \in K$ satisfying $\mathfrak{A} \prec_{\alpha_0} \mathfrak{B}$.

Proof. The equivalence $\mathfrak{A} \prec_Q \mathfrak{B}$ iff $\mathfrak{A} \prec_{\alpha_0} \mathfrak{B}$ combined with Theorem 3.1 obviously implies the sufficiency of the condition.

The proof of the necessity can be performed as in Theorem 3.1 so it will be omitted. □

Inspecting Theorems 3.1, 3.2 and 3.3 we can infer that there exist no minimal isomorphically α_0-complete systems. Moreover, a system $K \subseteqq K(R)$ is isomorphically Q-complete iff it is isomorphically α_0-complete.

For isomorphic G-completeness we have

Theorem 3.4. A system $K \subseteq K(R)$ is isomorphically complete with respect to the general product iff K contains an algebra $\mathfrak{A}=\langle A, F\rangle$ having two distinct elements a_1 and a_2 such that for arbitrary $r\in R$, $a\in\{a_1, a_2\}$ and $\mathbf{a}\in\{a_1, a_2\}^r$ there exists an $f\in F_r$ satisfying $f^{\mathfrak{A}}(a)=\mathbf{a}$.

Proof. Suppose that K is isomorphically G-complete. Let $\mathfrak{B}=\langle\{b_1, b_2\}, G\rangle$ be an algebra such that for all $r\in R$, $b\in B$ and $\mathbf{b}\in\{b_1, b_2\}^r$ there exists a $g\in G_r$ with $g^{\mathfrak{B}}(b)=\mathbf{b}$. Because of the G-completeness of K there is a G-product $\mathfrak{A}=\prod_{i=1}^{k}\mathfrak{A}_i[G, \psi]$ from K and a subalgebra $\overline{\mathfrak{A}}$ of $\mathfrak{A}$ satisfying $\varphi(\mathfrak{B})=\overline{\mathfrak{A}}$ under a suitable isomorphism φ. Let $(a_1', \ldots, a_k')$ and $(a_1'', \ldots, a_k'')$ be the φ-image of b_1 and b_2, respectively. Because of $b_1\neq b_2$ an index j satisfying $a_j'\neq a_j''$ can be selected. We shall prove that in this case the algebra $\mathfrak{A}_j$ fulfils the conditions of the Theorem. To this end take an $r\in R$, $a\in\{a_j', a_j''\}$, and $\mathbf{a}\in\{a_j', a_j''\}^r$. The algebra $\mathfrak{B}$ satisfies the requirements of the Theorem as well. Hence it can be given a $g\in G_r$, $b\in B$ and $\mathbf{b}\in B^r$ with the properties $g^{\mathfrak{B}}(b)=\mathbf{b}$, $\pi_j(\varphi(b))=a$ and $(\pi_j(\varphi(b_1)), \ldots, \pi_j(\varphi(b_r)))=\mathbf{a}$. From these equalities we can conclude that the operation $f=\psi^{(j)}(\varphi(b), g)\in F_r^j$ satisfies $f^{\mathfrak{A}_j}(a)=\mathbf{a}$.

Now assume that the elements a_1, a_2 of the algebra $\mathfrak{A}=\langle A, F\rangle\in K$ meet the requirements of the Theorem. Take an arbitrary algebra $\mathfrak{B}=\langle B, G\rangle$. Choose an injective mapping $\varphi: B\to\{a_1, a_2\}^k$ for a suitable $k\in N$, and construct the G-product $\mathfrak{C}=\prod_{i=1}^{k}\mathfrak{A}_i[G, \psi]$ where $\mathfrak{A}_i=\mathfrak{A}$ $(i=1, \ldots, k)$. For all $\mathbf{c}=(c_1, \ldots, c_k)\in A^k$, $r\in R$ and $g\in G_r$ let $\psi(\mathbf{c}, g)=(f^1, \ldots, f^k)$ be defined for all $i=1, \ldots, k$ by

$$f^i=\begin{cases}\text{an } f\in F_r \text{ satisfying } f^{\mathfrak{A}}(c_i)=(c_i^1, \ldots, c_i^r),\\ \text{if } \mathbf{c}=\varphi(b),\ g^{\mathfrak{B}}(b)=(b_1, \ldots, b_r) \text{ and } \varphi(b_j)=(c_1^j, \ldots, c_k^j)\\ \text{for } j=1, \ldots, r,\\ \text{an arbitrary element from } F_r \text{ otherwise.}\end{cases}$$

By virtue of this definition of the feedback function ψ an easy computation shows that φ is an isomorphic embedding of $\mathfrak{B}$ into the product $\mathfrak{C}$. □

By Theorem 3.4, there exists an algorithm to decide for arbitrary finite $K\subseteq K(R)$ whether K is isomorphically complete with respect to the general product.

Turning to the problem of homomorphic G-completeness we give a rephrased version of the known result from [8] for the case $R=\{1\}$ i.e., for unoids.

Proposition 3.5. A system $K\subseteq K(\{1\})$ is homomorphically complete with respect to the general product iff K contains a unoid $\mathfrak{A}=\langle A, F\rangle$ having an element a, two operational symbols f_1, f_2 and two polynomials p_1, p_2 satisfying

$$a_1=f_1(a)\neq f_2(a)=a_2$$

and

$$p_1(a_1)=p_2(a_2)=a.$$

This proposition implies that every minimal homomorphically G-complete system in $K(\{1\})$ is a singleton. The subsequent constructions show that this situation is bounded to the case $R=\{1\}$.

Let $R=\{r_1, \ldots, r_i\}\neq\{1\}$ be an arbitrary rank type. For every $r\in R$ and $1\leqq j\leqq r$ take a two-element algebra $\mathfrak{A}_{rj}=\langle\{a^1_{rj}, a^2_{rj}\}, F^{rj}\rangle$ having for every $(m, n)\in \in\{1, 2\}^2$ exactly one r-ary operational symbol $f_{mn}\in F^{rj}_r$ such that

$$(*)\qquad f_{mn}(a^k_{rj})=\begin{cases}(a^m_{rj}, \ldots, \overset{j^{\text{th}}}{\underset{\smile}{a^n_{rj}}}, \ldots, a^m_{rj}) & \text{if } k=m\\ (a^k_{rj}, \ldots, a^k_{rj}) & \text{otherwise}\end{cases}$$

and

$$(**)\qquad g(a^m_{rj})=(a^m_{rj}, \ldots, a^m_{rj}) \text{ for all } a^m_{rj}\in A_{rj},\quad g\in F^{rj} \text{ and } g\notin F^{rj}_r$$

hold.

Lemma 3.6. The system $K=\{\mathfrak{A}_{rj}|r\in R,\ 1\leqq j\leqq r\}$ is homomorphically G-complete and minimal.

Proof. Let $\mathfrak{C}=\langle\{1, 2\}, F\rangle\in K(R)$ be an algebra satisfying

$$F=\cup(\{f_{st}|s\in\{1, 2\}, \mathbf{t}\in\{1, 2\}^{r_i}\}|r_i\in R)$$

and

$$f_{st}(k)=\begin{cases}\mathbf{t} & \text{if } s=k\\ \underbrace{(s, \ldots, s)}_{r_i \text{ times}} & \text{otherwise}\end{cases}$$

for all $r_i\in R$ and $f_{st}\in F_{r_i}$.

Since the system $\{\mathfrak{C}\}$ is isomorphically G-complete to prove our lemma it is enough to show $\mathfrak{C}\in H_G(K)$. To this end take the G-product $\mathfrak{A}=\langle A, F\rangle=$ $=\prod_{\substack{r\in R\\ 1\leqq j\leqq r}}\mathfrak{A}_{rj}[F, \psi]$. If $\mathbf{a}\in A$ then let $v(\mathbf{a})$ denote the sum of the upper indices occurring in $\mathbf{a}$. For all $r_i\in R$, $f_{st}\in F_{r_i}$, $\mathfrak{A}_{rj}\in K$ and $\mathbf{a}\in A$, ψ corresponds to f_{st} the operation $f_{sv}\in F^{rj}_{r_i}$ where

$$v=\begin{cases}\pi_j(\mathbf{t}) & \text{if } r_i=r \text{ and } (-1)^{v(\mathbf{a})}=(-1)^s,\\ s & \text{otherwise.}\end{cases}$$

Define the mapping $\varphi: A\to\{1, 2\}$ in the following way:

$$\varphi(a)=\begin{cases}1 & \text{if } v(a) \text{ is an odd number,}\\ 2 & \text{otherwise.}\end{cases}$$

Now, using the previous definition of ψ, it can be proved that $\varphi: \mathfrak{A}\to\mathfrak{C}$ is a homomorphism.

Take an arbitrary algebra $\mathfrak{A}_{ri}\in M$. By virtue of the construction of M it is evident that $\mathfrak{A}_{sj}\in M$ and $(s, j)\neq(r, i)$ implies for all $f\in F^{sj}_r$ and $a\in A_{sj}$, $\pi_i(f(a))=a$. From this assumption it follows directly that the system $K-\{\mathfrak{A}_{ri}\}$ is not homomorphically G-complete. □

By similar methods one can prove

Theorem 3.7. Let s be an arbitrary natural number satisfying $1 \leqq s \leqq \sum_{r_i \in R} r_i$. Then a minimal homomorphically G-complete system K consisting of s algebras can effectively be constructed.

Finally we would like to remark that in the proof of Lemma 3.1 in [12] there is a mistake. Its correction can be found in [6].

DEPT. OF COMPUTER SCIENCE
A. JÓZSEF UNIVERSITY
ARADI VÉRTANÚK TERE 1
SZEGED, HUNGARY
H-6720

References

[1] ENGELFRIET, J., Bottom-up and top-down tree transformations, A comparison, *Math. Systems Theory*, v. 9, 1975, pp. 198—231.
[2] ÉSIK, Z., Decidability results concerning tree transducers I, *Acta Cybernet.*, v. 5, 1980, pp. 1—26.
[3] GÉCSEG, F., Composition of automata, Automata, Languages and Programming 2nd Colloquium, Lecture Notes in Computer Science, v. 14, 1974, pp. 351—363.
[4] GÉCSEG, F. and I. PEÁK, *Algebraic theory of automata*, Akadémiai Kiadó, Budapest, 1972.
[5] GÉCSEG, F. and M. STEINBY, Minimal ascending tree automata, *Acta Cybernet.*, v. 4, 1978, pp. 37—44.
[6] GÉCSEG, F. and M. STEINBY, *Tree automata*, Akadémiai Kiadó, Budapest, (to appear).
[7] GLUŠKOV, V. M., Абстрактная теория автоматов, *Успехи матем. наук*, v. 16 (101), 1961, pp. 3—62.
[8] LETIČEVSKIĬ, A. A., Условия полноты для конечных автоматов, *Журн. вычисл. матем. и матем. физ.*, v. 1, 1961, pp. 702—710.
[9] MAGIDOR, M. and G. MORAN, Finite automata over finite trees, *Tech. Rep. Hebrew Univ.*, Jerusalem, No. 30, 1969.
[10] RICCI, G., Cascades of tree-automata and computations in universal algebras, *Math. Systems Theory*, v. 7, 1973, pp. 201—208.
[11] STEINBY, M., On the structure and realization of tree automata, Second Coll. sur les Arbres en Algebre et en Programmation, Lille, 1977.
[12] VIRÁGH, J., Deterministic ascending tree automata I, *Acta Cybernet.*, v. 5, 1980, pp. 33—42.

(Received March 21, 1983)

Decidability results concerning tree transducers II

By Z. ÉSIK

1. Introduction

Let $\tau \subseteq T_F \times T_G$ be an arbitrary tree transformation induced by a top-down or bottom-up tree transducer **A**. It is said that **A** preserves regularity if $\tau(R)$ is a regular forest for each regular forest $R \subseteq T_F$. It is natural to raise the question whether the regularity preserving property of tree transducers is decidable or not. This question was positively answered for bottom-up transducers in [4]. Even more, it was shown that a bottom-up transducer preserves regularity if and only if it is equivalent to a linear bottom-up transducer. Concerning top-down transducers we have quiet different results. Although every linear top-down transducer preserves regularity as linear top-down tree transformations form a (proper) subclass of linear bottom-up transformations (cf. [2]), there are deterministic regularity preserving top-down tree transducers having no linear bottom-up equivalent. Another distinction lies in the fact that there is no algorithm which can decide the regularity preserving property of top-down transducers (cf. Theorem 2). However, restricting ourselves to deterministic top-down transducers we obtain positive result (cf. Theorem 1).

The notations will be used in accordance with [1]. Recall that a top-down tree transducer $\mathbf{A}=(F, A, G, A_0, \Sigma)$ is called uniform if each rewriting rule in Σ is of the form $af \to q(a_1x_1, \ldots, a_nx_n)$ where $n \geqq 0$, $f \in F_n$, $a, a_1, \ldots, a_n \in A$ and $q \in T_{G,n}$. In addition, if q is always linear (cf. [2]) then **A** is called linear. These concepts extend to top-down tree transducers with regular look-ahead, as well. Furthermore, one-state top-down tree transducers and their induced transformations will be called homomorphisms. If **A** is a homomorphism then we omit the single state in the presentation of Σ.

2. Deterministic top-down transducers

Let $\mathbf{A}=(F, A, G, a_0, \Sigma)$ be an arbitrary deterministic top-down transducer kept fixed in this section. Put $\tau=\tau_{\mathbf{A}}$. If there exist

$$n_1, n_2, m_1, m_2 \geqq 0, \quad \mathbf{a}\in A^{n_1}, \quad \mathbf{b}\in A^{m_1}, \quad \mathbf{c}\in A^{n_2}, \quad \mathbf{d}\in A^{m_2}, \quad p_0, p_1\in \hat{T}_{F,1}$$

$$p_2\in T_F, \quad q_0\in \hat{T}_{G,n_1+m_1}, \quad \mathbf{q}_1\in \hat{T}^{n_1}_{G,n_2}, \quad \mathbf{r}_1\in \hat{T}^{m_1}_{G,m_2}, \quad \mathbf{q}_2\in T^{n_2}_G, \quad \mathbf{r}_2\in T^{m_2}_G$$

such that we have

$$a_0 p_0 \overset{*}{\Rightarrow} q_0(\mathbf{a}x_1^{n_1}, \mathbf{b}x_1^{m_1}),$$

$$\mathbf{a}_1 p_1^{n_1} \overset{*}{\Rightarrow} \mathbf{q}_1(\mathbf{c}x_1^{n_2}), \quad \mathbf{b}x_1^{m_1} \overset{*}{\Rightarrow} \mathbf{r}_1(\mathbf{d}x_1^{m_2}),$$

$$\mathbf{c}p_2^{n_2} \overset{*}{\Rightarrow} \mathbf{q}_2, \quad \mathbf{d}p_2^{m_2} \overset{*}{\Rightarrow} \mathbf{r}_2,$$

$$\{a_i | i\in[n_1]\} = \{c_i | i\in[n_2]\}, \quad \{b_i | i\in[m_1]\} = \{d_i | i\in[m_2]\},$$

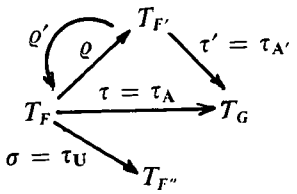
and both $\mathbf{q}_1$ and $\mathbf{r}_1$ contain an occurence of a symbol from G then we say that $\mathbf{A}$ satisfies condition $(*)$. Observe that our conditions imply that $n_i, m_i>0$ $(i=1, 2)$.

We are going to prove that $\mathbf{A}$ preserves regularity if and only if $(*)$ is not satisfied by $\mathbf{A}$. The necessity of this statement can be proved easily.

Lemma 1. If $\mathbf{A}$ preserves regularity then $\mathbf{A}$ does not satisfy condition $(*)$.

Proof. Assume that $\mathbf{A}$ satisfies condition $(*)$. Then, using the notations of the definition above, set $R=\{p_0(\underbrace{p_1(\ldots(p_1}_{n\text{-times}}(p_2))\ldots))|n\geqq 0\}$. R is regular and $\tau(R)$ consists of trees $q_0(\mathbf{r}_n, \mathbf{s}_n)$ $(n\geqq 0, \mathbf{r}_n\in T_G^{n_1}, \mathbf{s}_n\in T_G^{m_1})$ with the property that $n<\mathrm{rn}(\mathbf{r}_n)<\mathrm{rn}(\mathbf{r}_{n+1})$, $n<\mathrm{rn}(\mathbf{s}_n)<\mathrm{rn}(\mathbf{s}_{n+1})$[1]. Suppose that $\tau(R)$ is recognizable by a deterministic tree automaton $\mathbf{D}=(G, D, D_0)$. Let $n>m_1(1+\nu(G)+\ldots+\nu(G)^{|D|-1})$ be an arbitrary fixed integer. As $(q_0(\mathbf{r}_n, \mathbf{s}_n))_{\mathbf{D}}\in D_0$ also there is a vector of trees $\mathbf{s}\in T_G^{m_1}$ with $\mathrm{dp}(\mathbf{s})<|D|$ and $(q_0(\mathbf{r}_n, \mathbf{s}))_{\mathbf{D}}\in D_0$. However, as $\mathrm{dp}(\mathbf{s})<|D|$ we obtain that $\mathrm{rn}(\mathbf{s})\leqq$ $\leqq m_1(1+\nu(G)+\ldots+\nu(G)^{|D|-1})$. This contradicts $\tau(R)=T(\mathbf{D})$. Therefore, $\tau(R)$ is not regular, as was to be proved.

To prove the converse of Lemma 1 first we show that $\tau(\mathrm{dom}\,\tau)$ is regular if $\mathbf{A}$ does not satisfy $(*)$. This will be carried out by constructing a linear deterministic top-down tree transducer with regular look-ahead such that $\tau(\mathrm{dom}\,\tau)=$ $=\tau_{\mathbf{A}'}(\mathrm{dom}\,\tau_{\mathbf{A}'})$. The construction of $\mathbf{A}'$ will be made by the help of other tree transformations. Thus, we shall have the transformations indicated by the figure below:



[1] $\mathrm{rn}(\mathbf{r}_n)=\mathrm{rn}(r_{n_1})+\ldots+\mathrm{rn}(r_{n n_1})$, $\mathrm{rn}(\mathbf{s}_n)$ is similarly defined.

We begin with the definition of F''. First let $\bar{F}=\bigcup_{n\geqq 0} F_n$, $\bar{F}_n=\{(f, C, \varphi, \psi)|f\in F_n, C\subseteq B, \varphi: B\to P(A), \psi: A\rightarrowtail A \text{ for a subset } B\subseteq A\}$, i.e. φ is a mapping of B into the power-set of A and ψ is a partial function on A. Now the type F'' is defined by $F''_n=F_n\cup\bar{F}_n$ $(n\geqq 0)$.

The $\bar{F}$-depth $(\overline{\mathrm{dp}}(p))$ and $\bar{F}$-width $(\overline{\mathrm{wd}}(p))$ of a tree $p\in T_{F''}$ are defined by

$$\overline{\mathrm{dp}}(p)=0,\quad \overline{\mathrm{wd}}(p)=\overline{\mathrm{wd}_0}(p)=0 \quad \text{if} \quad p\in F_0,$$

$$\overline{\mathrm{dp}}(p)=1,\quad \overline{\mathrm{wd}}(p)=\overline{\mathrm{wd}_0}(p)=1 \quad \text{if} \quad p\in\bar{F}_0,$$

$$\overline{\mathrm{dp}}(p)=\max\{\overline{\mathrm{dp}}(p_i)|i\in[n]\},\quad \overline{\mathrm{wd}}(p)=\max\left\{\sum_{i=1}^{n}\overline{\mathrm{wd}_0}(p_i),\ \overline{\mathrm{wd}}(p_i)|i\in[n]\right\},$$

$$\overline{\mathrm{wd}_0}(p)=\sum_{i=1}^{n}\overline{\mathrm{wd}_0}(p_i) \quad \text{if} \quad p=f(p_1, \ldots, p_n) \quad \text{with} \quad n>0,\quad f\in F_n,$$

$$p_1, \ldots, p_n\in T_{F''},$$

$$\overline{\mathrm{dp}}(p)=1+\max\{\overline{\mathrm{dp}}(p_i)|i\in[n]\},\quad \overline{\mathrm{wd}}(p)=\max\{1, \overline{\mathrm{wd}}(p_i)|i\in[n]\},$$

$$\overline{\mathrm{wd}_0}(p)=1 \quad \text{if} \quad p=f(p_1, \ldots, p_n) \quad \text{where} \quad n>0,\quad f\in\bar{F}_n,$$

$$p_1, \ldots, p_n\in T_{F''}.$$

If n, m are given nonnegative integers then $T_{(n,m)}$ denotes the set of all trees $p\in T_{F''}$ with $\overline{\mathrm{dp}}(p)<n$ and $\overline{\mathrm{wd}}(p)\leqq m$.

We shall frequently use an equivalence relation denoted by $\sim$ on $T_{F''}$. Given $p, q\in T_{F''}$, $p\sim q$ if and only if one of the following three conditions holds:

(i) $p, q\in T_F$,

(ii) $p=f(p_1, \ldots, p_n)$, $q=f(q_1, \ldots, q_n)$ with $n\geqq 0$, $f\in\bar{F}_n$, $p_i, q_i\in T_{F''}$ and $p_i\sim q_i$ $(i\in[n])$,

(iii) $p=p_0(p_1, \ldots, p_n)$, $q=q_0(q_1, \ldots, q_n)$ with $n>0$, $p_0, q_0\in\tilde{T}_{F,n}$, $p_i, q_i\in T_{F''}$, $p_i\sim q_i$, $\mathrm{rt}(p_i), \mathrm{rt}(q_i)\in\bar{F}$ $(i\in[n])$.

If $p\in T_{F''}$ then $[p]$ denotes the block containing p under the partition induced by $\sim$.

The next statement can be proved in an easy way.

Lemma 2. $[p]$ is a regular forest for any $p\in T_{F''}$.

Now we introduce the transducer $\mathbf{U}$. $\mathbf{U}=(F, U, F'', u_0, \Sigma'')$ where

$$U=\{(B, B', C, \varphi, \psi)|B\subseteq A,\ B'\subseteq B,\ C\subseteq B,\ \varphi: B\to P(A),\ \psi: A\rightarrowtail A\},$$

$a_0=(\{a_0\}, \emptyset, \emptyset, \varphi, \psi)$ with $\varphi(a_0)=\{a_0\}$ and $\psi(a)=b$ if and only if $a=b=a_0$. Σ'' is determined as follows.

Let $u=(B, B', C, \varphi, \psi)$ be an arbitrary element of U, $f\in F_n$ $(n\geqq 0)$. Assume that in Σ there is a rule with left side af for any $a\in\bigcup\varphi(B)$. That is, $\bigcup\varphi(B)=\{a_1, \ldots, a_l, a_{11}, \ldots, a_{l_1 1}, \ldots, a_{1n}, \ldots, a_{l_n n}\}$ $(l, l_1, \ldots, l_n\geqq 0)$ and

$$a_i f\to q_i(\mathbf{b}_{i1}\mathbf{x}_1^{k_{i1}}, \ldots, \mathbf{b}_{in}\mathbf{x}_n^{k_{in}})\in\Sigma \quad (i\in[l]),$$

$$a_{ij}f\to c_{ij}x_j\in\Sigma \quad (i\in[l_j],\ j\in[n]),$$

where $k_{ij} \geqq 0$ $(i \in [l], j \in [n])$, $q_i \in \hat{T}_{G,k_i} - X$, $k_i = \sum_{j=1}^{n} k_{ij}$ $(i \in [l])$, $\mathbf{b}_{ij} \in A^{k_{ij}}$ $(i \in [l], j \in [n])$, $c_{ij} \in A$ $(i \in [l_j], j \in [n])$.

Then Σ'' is the smallest set of top-down rewriting rules satisfying (i) and (ii) below.

(i) If $\left|\{a \in B | \varphi(a) \cap \{a_1, \ldots, a_l\} \neq \emptyset\}\right| \geqq 2$ or

$\left|\{a \in B' | \varphi(a) \cap \{a_1, \ldots, a_l\} \neq \emptyset\}\right| \geqq 1$ or

$\left|\{a \in B - C | \varphi(a) \cap \{a_1, \ldots, a_l\} \neq \emptyset\}\right| \geqq 1$ or

$|C| \geqq 2$ and $\left|\{a \in B | \varphi(a) \cap \{a_1, \ldots, a_l\} \neq \emptyset\}\right| \geqq 1$

and

$$u_i = (B, B', C', \varphi_i, \psi_i) \quad (i \in [n]),$$

$$C' = \{a \in B | \varphi(a) \cap \{a_1, \ldots, a_l\} \neq \emptyset\},$$

$$\varphi_i(a) = \cup(B_{ji} | a_j \in \varphi(a)) \cup \{c_{ji} | a_{ji} \in \varphi(a)\} \quad (a \in B,\ i \in [n]),$$

where B_{ji} denotes the set of components of the vector $\mathbf{b}_{ji}$ $(j \in [l],\ i \in [n])$, $\psi_i(a) = b$ if and only if $a = b \in \cup \varphi_i(B)$ $(i \in [n])$, $f = (f, C', \varphi, \psi)$ then

$$uf \to \bar{f}(u_1 x_1, \ldots, u_n x_n) \in \Sigma''.$$

(ii) If not (i), i.e. $l = 0$ or
$l = 1$, $C = \{a_1\}$ and $a_1 \notin B'$
and for each $i \in [n]$
$u_i = (B, B', C, \varphi_i, \psi_i)$,
φ_i is the same as in the previous case,
$\psi_i = \psi \circ \psi_i'$ with $\psi_i'(a) = b$ if and only if $a = a_{ji}$, $b = c_{ji}$ $(j \in [l_i])$,
then

$$uf \to f(u_1 x_1, \ldots, u_n x_n) \in \Sigma''.$$

Observe that $\mathbf{U}$ is a deterministic top-down relabeling. The following properties of $\mathbf{U}$ will be used without any reference. First, if $up \overset{*}{\underset{\mathbf{u}}{\Rightarrow}} q(\mathbf{v}(x_1, \ldots, x_n))$ $(n \geqq 0,\ u \in U,\ v \in U^n)$ and $p, q \in \hat{T}_{F,n}$ then $p = q$. Secondly, let $\mathbf{a} \in A^k$ $(k \geqq 0)$ be arbitrary and identify $\mathbf{a}$ with the state $u = (B, B', \emptyset, \varphi, \psi)$ where $B = \{a_i | i \in [k]\}$, $B' = \{a_i | i \in [k],\ \exists j \in [k]\ i \neq j,\ a_i = a_j\}$, $\varphi(a) = \{a\}$ if $a \in B$ and $\psi(a) = b$ if and only if $a = b \in B$. Denote by $\sigma_{\mathbf{a}}$ the transformation $\tau_{\mathbf{U}(u)}$ and similarly, put $\tau_{a_i} = \tau_{\mathbf{A}(a_i)}$ $(i \in [k])$. Then, for any $p \in T_F$, $p \in \operatorname{dom} \sigma_{\mathbf{a}}$ if and only if $p \in \bigcap_{i=1}^{k} \operatorname{dom} \tau_{a_i}$.

In the next few lemmata we shall point to further connections between $\mathbf{A}$ and $\mathbf{U}$.

Lemma 3. If $\mathbf{A}$ does not satisfy condition $(*)$ then $\overline{\operatorname{dp}}(\sigma_{\mathbf{a}}(p)) < 2|A|^2 \|A\|^2$ holds for any $\mathbf{a} \in A^k$ $(k \geqq 0)$ and $p \in T_F$ provided that $\sigma_{\mathbf{a}}(p)$ is defined and there exist trees $r \in \hat{T}_{F,1}$ and $r' \in \hat{T}_{G,k}$ with $a_0 r \overset{*}{\underset{\mathbf{A}}{\Rightarrow}} r'(\mathbf{a}x_1^k)$.

Proof. Let $L = |A|^2 \|A\|^2$ and suppose that $a_0 r \overset{*}{\underset{\mathbf{A}}{\Rightarrow}} r'(\mathbf{a}x_1^k)$ and $\overline{\operatorname{dp}}(\sigma_a(p)) \geqq 2L$. Then $k \geqq 2$ and there exist $p_0, \ldots, p_{2L-1} \in \hat{T}_{F,1}$, $p_{2L} \in T_F$, $q_0, \ldots, q_{2L-1} \in \hat{T}_{F'',1}$,

$q_{2L}\in T_{F''}$, $u_1, \ldots, u_{2L}\in U$ such that

$$p = p_0(\ldots(p_{2L})\ldots), \quad q = q_0(\ldots(q_{2L})\ldots),$$

$$\mathbf{a}p_0 \underset{\mathbf{u}}{\overset{*}{\Rightarrow}} q_0(u_1 x_1), \quad u_i p_i \underset{\mathbf{u}}{\overset{*}{\Rightarrow}} q_i(u_{i+1}x_1) \quad (i = 1, \ldots, 2L-1), \quad u_{2L}p_{2L} \underset{\mathbf{u}}{\overset{*}{\Rightarrow}} q_{2L},$$

furthermore, $\mathrm{rt}(q_i)\in\bar{F}$, say, $\mathrm{rt}(q_i) = (f_i, C_i, \varphi_i, \psi_i)$ $(i=1, \ldots, 2L)$. Let $D_1 = C_1\cup C_2, \ldots, D_L = C_{2L-1}\cup C_{2L}$. It is not difficult to see by the definition of $\mathbf{U}$ that for any $i\in[L]$ there exist indices $j_i\neq k_i$ $(j_i, k_i\in[k])$ with $a_{j_i}, a_{k_i}\in D_i$. On the other hand, as $L=|A|^2\|A\|^2$, there exist $i_1<i_2$ $(i_1, i_2\in[L])$ such that $a_{j_{i_1}} = a_{j_{i_2}}$, $a_{k_{i_1}} = a_{k_{i_2}}$, $S_{i_1} = S_{i_2}$ and $T_{i_1} = T_{i_2}$ where S_i and T_i are defined by $S_i = \varphi_{2i-1}(a_{j_i})$ and $T_i = \bigcup(\varphi_{2i-1}(a_j)\,|\,j\in[k], j\neq j_i)$. Without loss of generality we may take $j_{i_1} = j_{i_2} = 1$ and $k_{i_1} = k_{i_2} = 2$.

As $\sigma_a(p)$ is defined also $\tau_{a_i}(p)$ is defined for any $i\in[k]$. Thus, if $r_1 = p_0(\ldots(p_{2i_1-2})\ldots)$, $r_2 = p_{2i_1-1}(\ldots(p_{2i_2-2})\ldots)$, $r_3 = p_{2i_2-1}(\ldots(p_{2L})\ldots)$ then the derivations

$$a_1 r_1 \underset{\mathbf{A}}{\overset{*}{\Rightarrow}} s_1(\mathbf{c}_1\mathbf{x}_1^{n_1}), \quad (a_2, \ldots, a_k)\mathbf{r}_1^{k-1} \underset{\mathbf{A}}{\overset{*}{\Rightarrow}} \mathbf{t}_1(\mathbf{d}_1\mathbf{x}_1^{m_1}),$$

$$\mathbf{c}_1\mathbf{r}_2^{n_1} \underset{\mathbf{A}}{\overset{*}{\Rightarrow}} \mathbf{s}_2(\mathbf{c}_2\mathbf{x}_1^{n_2}), \quad \mathbf{d}_1\mathbf{r}_2^{m_1} \underset{\mathbf{A}}{\overset{*}{\Rightarrow}} \mathbf{t}_2(\mathbf{d}_2\mathbf{x}_1^{m_2}),$$

$$\mathbf{c}_2\mathbf{r}_3^{n_2} \underset{\mathbf{A}}{\overset{*}{\Rightarrow}} \mathbf{s}_3, \quad \mathbf{d}_2\mathbf{r}_3^{m_2} \underset{\mathbf{A}}{\overset{*}{\Rightarrow}} \mathbf{t}_3$$

exist where $s_1\in\hat{T}_{G,n_1}$, $\mathbf{t}_1\in\hat{T}^{k-1}_{G,m_1}$, $\mathbf{s}_2\in\hat{T}^{n_1}_{G,n_2}$, $\mathbf{t}_2\in\hat{T}^{m_1}_{G,m_2}$, $\mathbf{s}_3\in T^{n_2}_G$, $\mathbf{t}_3\in T^{m_2}_G$ and $\mathbf{c}_i\in A^{n_i}$, $\mathbf{d}_i\in A^{m_i}$ $(i=1, 2)$.

Since $1, 2\in D_{i_1}$ we have that both $\mathbf{s}_2$ and $\mathbf{t}_2$ contain an occurence of a symbol from G. Furthermore, as the sets $S_{i_1}, S_{i_2}, T_{i_1}$ and T_{i_2} coincide with the set of components of $\mathbf{c}_1, \mathbf{c}_2, \mathbf{d}_1$ and $\mathbf{d}_2$, respectively, it follows that $\mathbf{c}_1$ and $\mathbf{d}_1$ have the same set of components as $\mathbf{c}_2$ and $\mathbf{d}_2$.

By

$$a_0 r(r_1) \underset{\mathbf{A}}{\overset{*}{\Rightarrow}} r'(s_1(\mathbf{c}_1\mathbf{x}_1^{n_1}), \quad \mathbf{t}_1(\mathbf{d}_1\mathbf{x}_1^{m_1})),$$

$$\mathbf{c}_1\mathbf{r}_2^{n_1} \underset{\mathbf{A}}{\overset{*}{\Rightarrow}} \mathbf{s}_2(\mathbf{c}_2\mathbf{x}_1^{n_2}), \quad \mathbf{d}_1\mathbf{r}_2^{m_1} \underset{\mathbf{A}}{\overset{*}{\Rightarrow}} \mathbf{t}_2(\mathbf{d}_2\mathbf{x}_1^{m_2}),$$

$$\mathbf{c}_2\mathbf{r}_3^{n_2} \underset{\mathbf{A}}{\overset{*}{\Rightarrow}} \mathbf{s}_3, \quad \mathbf{d}_2\mathbf{r}_3^{m_2} \underset{\mathbf{A}}{\overset{*}{\Rightarrow}} \mathbf{t}_3$$

this yields that $\mathbf{A}$ satisfies condition $(*)$, which is a contradiction.

Lemma 4. Let $\mathbf{a}\in A^k$ $(k\geqq 0)$ be arbitrary. Put $B=\{a_i\,|\,i\in[k]\}$ and assume that

$$\mathbf{a}p_0 \underset{\mathbf{u}}{\overset{*}{\Rightarrow}} p_0(\mathbf{u}(x_1, \ldots, x_n)), \quad \mathbf{a}p_0' \underset{\mathbf{u}}{\overset{*}{\Rightarrow}} p_0'(\mathbf{u}'(x_1, \ldots, x_n)),$$

$$\mathbf{u}\mathbf{p} \underset{\mathbf{u}}{\overset{*}{\Rightarrow}} \mathbf{q}, \quad \mathbf{u}'\mathbf{p}' \underset{\mathbf{u}}{\overset{*}{\Rightarrow}} \mathbf{q}',$$

$$\mathrm{rt}(\mathbf{q}) = \mathrm{rt}(\mathbf{q}')\in\bar{F}^n$$

where $n\geqq 0$, $p_0, p_0'\in\hat{T}_{F,n}$, $\mathbf{p}, \mathbf{p}'\in T^n_F$, $\mathbf{q}, \mathbf{q}'\in T^n_{F''}$, $\mathbf{u}, \mathbf{u}'\in U^n$.

Then $n\leqq|A|$ and $\tau_b(p_0(\mathbf{p})) = \tau_b(p_0'(\mathbf{p}))$ for any $b\in B$.

Proof. Suppose that $\mathrm{rt}(q_i)=(f_i, C_i, \varphi_i, \psi_i)$ $(i\in[n])$. It is not difficult to see by the definition of $\mathbf{U}$ that for any $i\in[n]$ there is a state $b\in B$ with $\psi_i(b)$ being defined and $bp_0 \overset{*}{\underset{\mathbf{A}}{\Rightarrow}} \psi_i(b)x_i$. Therefore, $n\leqq|B|$ and also $n\leqq|A|$. Similarly, for each $b\in B$ there is an integer $i\in[n]$ such that $\psi_i(b)$ is defined and $bp_0 \overset{*}{\underset{\mathbf{A}}{\Rightarrow}} \psi_i(b)x_i$, $bp_0' \overset{*}{\underset{\mathbf{A}}{\Rightarrow}} \psi_i(b)x_i$. From this $\tau_b(p_0(\mathbf{p}))=\tau_b(p_0'(\mathbf{p}))$ follows immediately.

Lemma 5. Let $\mathbf{a}\in A^k$ $(k>0)$ and define the set B as previously. Set $B'=\{a_i | i\in[k],\ \exists j\in[k]\ i\neq j,\ a_i=a_j\}$ and assume that

$$\mathbf{a}p_0(f(p_1, \ldots, p_{i-1}, x_1, p_{i+1}, \ldots, p_n)) \overset{*}{\underset{\mathbf{u}}{\Rightarrow}} r_0(\bar{f}(r_1, \ldots, r_{i-1}, ux_1, r_{i+1}, \ldots, r_n)),$$

$$\mathbf{a}p_0'(f(p_1', \ldots, p_{i-1}', x_1, p_{i+1}', \ldots, p_n')) \overset{*}{\underset{\mathbf{u}}{\Rightarrow}} r_0'(\bar{f}(r_1', \ldots, r_{i-1}', ux_1, r_{i+1}', \ldots, r_n')),$$

$$uq_0 \overset{*}{\underset{\mathbf{u}}{\Rightarrow}} q_0(\mathbf{v}(x_1, \ldots, x_m)), \quad uq_0' \overset{*}{\underset{\mathbf{u}}{\Rightarrow}} q_0'(\mathbf{v}'(x_1, \ldots, x_m)),$$

$$\mathbf{v}\mathbf{q} \overset{*}{\underset{\mathbf{u}}{\Rightarrow}} \mathbf{s}, \quad \mathbf{v}'\mathbf{q}' \overset{*}{\underset{\mathbf{u}}{\Rightarrow}} \mathbf{s}',$$

$$\mathrm{rt}(\mathbf{s}) = \mathrm{rt}(\mathbf{s}')\in\bar{F}^m,$$

where $n>0$, $m\geqq 0$, $i\in[n]$, $p_0, p_0'\in\hat{T}_{F,1}$, $f\in F_n$, $\bar{f}=(f, C, \varphi, \psi)\in\bar{F}_n$, $p_j, p_j'\in T_F$, $r_0, r_0'\in\hat{T}_{F'',1}$, $r_j,\ r_j'\in T_{F''}$ $(j\in[n]-\{i\})$, $q_0, q_0'\in\hat{T}_{F,m}$, $\mathbf{q}, \mathbf{q}'\in T_F^m$, $\mathbf{s}, \mathbf{s}'\in T_{F''}^m$, $u\in U$, $\mathbf{v}, \mathbf{v}'\in U^m$.

If $|C|\geqq 2$ or $C\cap B'\neq\emptyset$ then $\tau_b(p_0(f(p_1, \ldots, p_{i-1}, q_0(\mathbf{q}), p_{i+1}, \ldots, p_n)))=\tau_b(p_0(f(p_1, \ldots, p_{i-1}, q_0'(\mathbf{q}), p_{i+1}, \ldots, p_n)))$ is valid for any $b\in B$. If $|C|=1$ and $C\cap B'=\emptyset$ then we have the same equality for any $b\in B-C$. Furthermore, $m\leqq|A|$.

Proof. Similar to the proof of Lemma 4.

By succesive applications of the previous two lemmata we obtain

Lemma 6. Assume that $\mathbf{a}\mathbf{p} \overset{*}{\underset{\mathbf{A}}{\Rightarrow}} \mathbf{q}$ where $\mathbf{a}\in A^k$, $\mathbf{p}\in T_F^k$, $\mathbf{q}\in T_G^k$, $k\geqq 0$. If $\sigma_{\mathbf{a}}(p_1)\sim\ldots\sim\sigma_{\mathbf{a}}(p_k)$ then there is a tree $p_0\in T_F$ with $\sigma_{\mathbf{a}}(p_0)\sim\sigma_{\mathbf{a}}(p_1)$ and $\mathbf{a}p_0^k \overset{*}{\underset{\mathbf{A}}{\Rightarrow}} \mathbf{q}$. Furthermore, if $r\in\bigcap_{i=1}^{k} \mathrm{dom}\,\tau_{a_i}$ then $\overline{\mathrm{wd}}(\sigma_{\mathbf{a}}(r))\leqq|A|$.

Lemma 7. Let $\mathbf{a}\in A^k$ $(k\geqq 0)$, $f\in F_n$ $(n\geqq 0)$, $\mathbf{b}_{ij}\in A^{m_{ij}}$ $(m_{ij}\geqq 0,\ i\in[k],\ j\in[n])$ and $q_i\in\hat{T}_{G,m_i}$ $(i\in[k],\ m_i=\sum_{j=1}^{n} m_{ij})$. Assume that each of the productions $a_i f\to q_i(\mathbf{b}_{i1}\mathbf{x}_1^{m_{i1}}, \ldots, \mathbf{b}_{in}\mathbf{x}_n^{m_{in}})$ $(i\in[k])$ is in Σ. Furthermore, let $p_i, p_i'\in T_F$, $\mathbf{c}_i=(\mathbf{b}_{1i}, \ldots, \mathbf{b}_{ki})$ $(i\in[n])$. Then $\sigma_{\mathbf{c}_i}(p_i)\sim\sigma_{\mathbf{c}_i}(p_i')$ $(i\in[n])$ implies $\sigma_{\mathbf{a}}(f(p_1, \ldots, p_n))\sim\sigma_{\mathbf{a}}(f(p_1', \ldots, p_n'))$.

Proof. The proof will be carried out in case of $n=1$ only. As $n=1$ we may simplify our notations: put $p=p_1$, $p'=p_1'$, $\mathbf{b}_i=\mathbf{b}_{i1}$ $(i\in[k])$, $\mathbf{c}=\mathbf{c}_1$. Moreover, let $B=\{a_i | i\in[k]\}$, $B'=\{a_i | i\in[k],\ \exists j\in[k]\ i\neq j,\ a_i=a_j\}$, $C=\{c_i | i\in[\sum_{j=1}^{k} m_j]\}$, $C'=\{c_i | i\in[\sum_{j=1}^{k} m_j],\ \exists i'\in[\sum_{j=1}^{k} m_j]\ i'\neq i,\ c_i=c_{i'}\}$.

As $p, p' \in \bigcap_{i=1}^{k} \bigcap_{j=1}^{m_i} \operatorname{dom} \tau_{b_{ij}}$ and the productions above exist also $f(p), f(p') \in \in \bigcap_{i=1}^{k} \operatorname{dom} \tau_{a_i}$. This implies that both $\sigma_a(f(p))$ and $\sigma_a(f(p'))$ are defined.

In the remaining part of the proof we shall make some transformations on the trees $f(\sigma_c(p))$ and $f(\sigma_c(p'))$ by the help of a deterministic top-down tree transducer $\mathbf{V} = (F'', V, F'', v_0, \Sigma_V)$. In this transducer $V = \{v_0\} \cup \{(D, \psi) | D \subseteq B, \psi: A \rightarrowtail A\}$ and Σ_V consisits of the following five types of rules:

(i) If $q_i = x_1$ for every $i \in [k]$ then

$$v_0 f \to f((\varnothing, \psi) x_1) \in \Sigma_V$$

where $\psi(a) = b$ if and only if $a = a_i$ and $b = b_{i1}$ for an index $i \in [k]$.

(ii) If $D = \{a_i | i \in [k], q_i \neq x_1\}$ is not empty then

$$v_0 f \to (f, D, \varphi, \psi)((D, \psi_1) x_1) \in \Sigma_V$$

where $\varphi: B \to P(A)$, $\varphi(a) = \{a\}$ $(a \in B)$; moreover, $\psi(a) = a$ if $a \in B$, $\psi(a)$ is undefined if $a \notin B$; $\psi_1(a) = a$ if $a \in C$, otherwise $\psi_1(a)$ is undefined.

(iii) $(D, \psi) g \to g((D, \psi) x_1, \ldots, (D, \psi) x_l) \in \Sigma_V$ for any $(D, \psi) \in V$ and $g \in F_l$ $(l \geqq 0)$.

(iv) If $(D, \psi) \in V$, $D' \subseteq C$ and either $|D| > 1$ or $D \cap B' \neq \varnothing$ or $\{a_i | \{b_{i1}, \ldots, b_{im_i}\} \cap D' \neq \varnothing\} \neq D$ then

$$(D, \psi)(g, D', \varphi', \psi') \to (g, D'', \varphi'', \psi'')((D'', \psi_1) x_1, \ldots, (D'', \psi_l) x_l) \in \Sigma_V$$

for any $(g, D', \varphi', \psi') \in \bar{F}_l$ $(l \geqq 0)$ with $\varphi': C \to P(A)$ where $D'' = \{a_i | i \in [k], \{b_{i1}, \ldots, b_{im_i}\} \cap D' \neq \varnothing\}$; $\varphi'': B \to P(A)$ and $\varphi''(a_i) = \bigcup_{j=1}^{m_i} \varphi'(b_{ij})$ $(i \in [k])$; $\psi'' = \psi \circ \psi'$ and $\psi_i(a) = b$ if and only if $a = b$ and a occurs in the right side of a rule $cg \to s \in \Sigma$ with $c \in \bigcup \varphi'(C)$.

(v) If $(D, \psi) \in V$, $D' \subseteq C$, furthermore $|D| = 1$, $D \cap B' = \varnothing$ and $\{a_i | i \in [k], \{b_{i1}, \ldots, b_{im_i}\} \cap D' \neq \varnothing\} = D$ then for every $(g, D', \varphi', \psi') \in \bar{F}_l$ with $\varphi': C \to P(A)$

$$(D, \psi)(g, D', \varphi', \psi') \to g((D, \psi_1) x_1, \ldots, (D, \psi_l) x_l) \in \Sigma_V$$

where $\psi_i = \psi \circ \eta_i$ and $\eta_i(a) = b$ if and only if $ag \to bx_i \in \Sigma$.

It can be seen that $\tau_V(f(\sigma_c(p))) = \sigma_a(f(p))$ and $\tau_V(f(\sigma_c(p'))) = \sigma_a(f(p'))$. On the other hand, by $\sigma_c(p) \sim \sigma_c(p')$ it follows that $\tau_V(f(\sigma_c(p))) \sim \tau_V(f(\sigma_c(p')))$. Therefore, $\sigma_a(f(p)) \sim \sigma_a(f(p'))$, as was to be proved.

We now turn to the definition of F'. For every integer $i \geqq 1$ let K_i denote the maximal number of occurences of the variable x_i in the right side of a rule in Σ. Put $K = \max\{1, K_i | i \in [v(F)]\}$, $F'_{nK} = F_n$ $(n \geqq 0)$ and $F'_m = \varnothing$ otherwise.

As it was mentioned we introduce two homomorphisms $\varrho \subseteq T_F \times T_{F'}$ and $\varrho' \subseteq T_{F'} \times T_F$ connecting T_F and $T_{F'}$. The rules defining ϱ are $f \to f(\mathbf{x}_1^K, \ldots, \mathbf{x}_n^K)$ $(f \in F_n, n \geqq 0)$, while the rules corresponding to ϱ' are $f \to f(x_{i_1}, \ldots, x_{i_n})$ $(f \in F_n, n \geqq 0)$ with $i_1 \in [K], \ldots, i_n \in [nK] - [(n-1)K]$. Observe that ϱ is deterministic and we have $\varrho'(\varrho(p)) = \{p\}$ for any $p \in T_F$.

We continue by defining the transducer $\mathbf{A}' = (F', A', G, a'_0, \Sigma')$. In this

system $A'=\{(a, B, B')|a\in B, B\subseteq A, B'\subseteq B\}$, $a_0'=(a_0, \{a_0\}, \emptyset)$ and Σ' is the smallest set of rewriting rules with the following property.

Let $l>0$, $B=\{a_1, \ldots, a_l\}\subseteq A$, $B'=\{a_{m_1}, \ldots, a_{m_k}\}$ $(1\leqq m_1<\ldots<m_k\leqq l)$, $a=a_1$. Assume that the rules $a_if\to q_i(\mathbf{a}_{i1}\mathbf{x}_1^{k_{i1}}, \ldots, \mathbf{a}_{in}\mathbf{x}_n^{k_{in}})$ are in Σ where $n\geqq 0$, $f\in F_n$, $k_{ij}\geqq 0$, $\mathbf{a}_{ij}\in A^{k_{ij}}$, $q_i\in\hat{T}_{G,k_{i1}+\ldots+k_{in}}$ $(i\in[l], j\in[n])$. Furthermore, let $r_j\in T(2|A|^2\|A\|^2, |A|)$, and set $R_j=\{p\in T_{F'}, |\varrho'(p)\leqq\sigma_{\mathbf{b}_j}^{-1}([r_j])\}$ $(j\in[n])$, where $\mathbf{b}_j=(\mathbf{a}_{1j}, \ldots, \mathbf{a}_{lj}, \mathbf{a}_{m_1j}, \ldots, \mathbf{a}_{m_kj})$. R_j is regular by Lemma 2 and some results in [2]. Finally, denote by B_j the set of components of $\mathbf{b}_j$ and put $B_j'=\{b\in A|b$ occurs at least twice in $\mathbf{b}_j\}$ $(j\in[n])$, $c_{ij}=a_{1ij}$ $(i\in[n], j\in[k_{1i}])$, $k_i=k_{1i}$ $(i\in[n])$. Then the rule

$$((a, B, B')f\to q_1((c_{11}, B_1, B_1')x_1, \ldots, (c_{1k_1}, B_1, B_1')x_{k_1}, \ldots$$

$$\ldots, (c_{n1}, B_n, B_n')x_{(n-1)K+1}, \ldots, (c_{nk_n}, B_n, B_n')x_{(n-1)K+k_n}),$$

$$\underbrace{R_1, \ldots, R_1}_{K\text{-times}}, \ldots, \underbrace{R_n, \ldots, R_n}_{K\text{-times}})$$

is in Σ'.

Observe that with the definition above $\mathbf{A}'$ becomes a linear deterministic top-down tree transducer with regular look-ahead. Just as in case of $\mathbf{A}''$ we may treat any vector $\mathbf{a}\in A^l$ — but now with $l>0$ — as an element of A': if $\mathbf{a}\in A^l$ $(l>0)$ then identify $\mathbf{a}$ with (a_1, B, B') where $B=\{a_i|i\in[l]\}$, $B'=\{a_i|i\in[l], J_j\in[l]\ i\neq j, a_i=a_j\}$. Assume that $\mathbf{a}p\overset{*}{\underset{\mathbf{A}'}{\Rightarrow}}q$ $(p\in T_{F'}, q\in T_G)$. Then one can easily prove that $\varrho'(p)\subseteq$ $\subseteq\bigcap_{i=1}^{l}\operatorname{dom}\tau_{a_i}$. However, there is a much more close connection between $\mathbf{A}$ and $\mathbf{A}'$. This is shown by Lemmata 8 and 9. In these Lemmata we shall assume that $\mathbf{A}$ does not satisfy condition $(*)$.

Lemma 8. $\tau(\operatorname{dom}\tau)\subseteq\tau'(\operatorname{dom}\tau')$.

Proof. We shall prove that if $a_0p_0\overset{*}{\underset{\mathbf{A}}{\Rightarrow}}q_0(\mathbf{a}\mathbf{x}_1^k)$ and $\mathbf{a}\mathbf{p}^k\overset{*}{\underset{\mathbf{A}}{\Rightarrow}}\mathbf{q}$ where $k>0$, $p_0\in\hat{T}_{F,1}$, $p\in T_F$, $q_0\in\hat{T}_{G,k}$, $\mathbf{q}\in T_G^k$, $\mathbf{a}\in A^k$ then also $\mathbf{a}\varrho(p)\overset{*}{\underset{\mathbf{A}'}{\Rightarrow}}q_1$. From this the statement follows by taking $p_0=x_1$.

If $\operatorname{dp}(p)=0$, i.e. $p\in F_0$, then $\mathbf{a}\varrho(p)\overset{*}{\underset{\mathbf{A}'}{\Rightarrow}}q_1$ is obviously valid. We proceed by induction on $\operatorname{dp}(p)$. Therefore, suppose that $\operatorname{dp}(p)>0$ and the proof is done for trees with depth less than $\operatorname{dp}(p)$. Then $p=f(p_1, \ldots, p_n)$ where $n>0$, $f\in F_n$, $p_1, \ldots, p_n\in T_F$ and $\operatorname{dp}(p_i)<\operatorname{dp}(p)$ $(i\in[n])$. As the generalization to arbitrary n is straigthforward we shall deal with $n=1$ only. Since $\mathbf{a}\mathbf{p}^k\overset{x}{\underset{\mathbf{A}}{\Rightarrow}}\mathbf{q}$ there exist rules $a_if\to r_i(\mathbf{b}_i\mathbf{x}_1^{l_i})\in\Sigma$ $(i\in[k], l_i\geqq 0, r_i\in\hat{T}_{G,l_i}, \mathbf{b}_i\in A^{l_i})$ such that $\mathbf{b}_i\mathbf{p}_1^{l_i}\overset{*}{\underset{\mathbf{A}}{\Rightarrow}}\mathbf{s}_i$ and $q_i=r_i(\mathbf{s}_i)$ hold for some $\mathbf{s}_i\in T_G^{l_i}$. Put $l=l_1+\ldots+l_k$, $\mathbf{b}=(\mathbf{b}_1, \ldots, \mathbf{b}_k)$, $B=\{b|b$ occurs in $\mathbf{b}\}$, $B'=\{b|b$ occurs at least twice in $\mathbf{b}\}$. As $a_0p_0(f(x_1))\overset{*}{\underset{\mathbf{A}}{\Rightarrow}}q_0(r_1(\mathbf{b}_1\mathbf{x}_1^{l_1}), \ldots, r_k(\mathbf{b}_k\mathbf{x}_1^{l_k}))$ and $\mathbf{b}\mathbf{p}_1^l\overset{*}{\underset{\mathbf{A}}{\Rightarrow}}(\mathbf{s}_1, \ldots, \mathbf{s}_k)$ we have that $\sigma_{\mathbf{b}}(p_1)$ is defined, $\sigma_{\mathbf{b}}(p_1)\in T(2|A|^2\|A\|^2, |A|)$ (cf. Lemmata 3 and 6). Set $R=\{p'\in T_{F'}|\varrho'(p')\subseteq\sigma_{\mathbf{b}}^{-1}([\sigma_{\mathbf{b}}(p_1)])$. By the construction of

$\mathbf{A}'$ we know that $(af \to r_1((b_{11}, B, B')x_1, \ldots, (b_{1l_1}, B, B')x_{l_1}), \underbrace{R, \ldots, R}_{K\text{-times}})$ is in Σ. Now, if $l_1=0$ then we get $\mathbf{a}\varrho(p) \underset{\mathbf{A}}{\overset{*}{\Rightarrow}} q_1$ immediately. If $l_1>0$ then we obtain $(b_{11}, B, B')\varrho(p_1) \underset{\mathbf{A}'}{\overset{*}{\Rightarrow}} s_{11}, \ldots, (b_{1l_1}, B, B')\varrho(p_1) \underset{\mathbf{A}'}{\overset{*}{\Rightarrow}} s_{1l_1}$ by the induction hypothesis. As $\varrho(p_1)\in R$ we again have $\mathbf{a}\varrho(p) \underset{\mathbf{A}'}{\overset{*}{\Rightarrow}} q_1$.

Lemma 9. $\tau'(\mathrm{dom}\,\tau') \subseteq \tau(\mathrm{dom}\,\tau)$.

Proof. We are going to show that if $\mathbf{a}p' \underset{\mathbf{A}'}{\overset{*}{\Rightarrow}} q$ where $\mathbf{a}\in A^l$ $(l>0)$ $p'\in T_{F'}$, $q\in T_G$ then there exist trees $r\in T_{F''}$ and $p\in\sigma_{\mathbf{a}}^{-1}([r])$ with $\varrho'(p')\subseteq\sigma_{\mathbf{a}}^{-1}([r])$ and $a_1 p \underset{\mathbf{A}}{\overset{*}{\Rightarrow}} q$. If $\mathrm{dp}(p')=0$ then it is trivial: take $p=p'$, $r=\sigma_{\mathbf{a}}(p)$. Assume now that this statement is valid for trees with depth less than $\mathrm{dp}(p')$ and $\mathrm{dp}(p')\geqq 1$. Then $p'=f(p'_1, \ldots, p'_{nK})$ $(n>0)$ with $\mathrm{dp}(p'_1), \ldots, \mathrm{dp}(p'_{nK}) < \mathrm{dp}(p')$. We shall restrict ourselves to the case $n=1$. Since $\mathbf{a}p' \underset{\mathbf{A}'}{\overset{*}{\Rightarrow}} q$ we get

$$(\mathbf{a}f \to q_0((b_1, B, B')x_1, \ldots, (b_k, B, B')x_k), \underbrace{R, \ldots, R}_{K\text{-times}}) \in \Sigma',$$

$$(b_i, B, B')p'_i \underset{\mathbf{A}'}{\overset{*}{\Rightarrow}} q_i \quad (i\in[k]), \quad p'_i\in R \quad (i\in[K]),$$

$$q = q_0(q_1, \ldots, q_k)$$

for some k $(0\leqq k\leqq K)$, $b_1, \ldots, b_k\in A$, $B, B'\subseteq A$ with $\{b_1, \ldots, b_k\}\subseteq B$, $B'\subseteq B$, $q_0\in\hat{T}_{G,k}$, $q_1, \ldots, q_k\in T_G$ and a regular forest $R=\{s\in T_{F'} \mid \varrho'(s)\subseteq\sigma_{\mathbf{c}}^{-1}([r_1])\}$ where $r_1\in T_{F''}$ and $\mathbf{c}$ is an arbitrary vector containing one component c_i for each element c_i of B and a distinct component c_j for each element c_j of B'. We have by the definition of $\mathbf{A}'$ that $a_1 f \to q_0(b_1 x_1, \ldots, b_k x_1)\in\Sigma$. Furthermore, as $\varrho'(p'_1), \ldots, \varrho'(p'_K)\subseteq\sigma_{\mathbf{c}}^{-1}([r_1])$, by Lemma 7 we have $\varrho'(f(p'_1, \ldots, p'_K))\subseteq\sigma_{\mathbf{a}}^{-1}([r])$ for a suitable $r\in T_{F''}$.

If $k=0$ then let $\bar{p}\in\varrho'(p'_1)$ be arbitrary, $p=f(\bar{p})$. $a_1 p \underset{\mathbf{A}}{\overset{*}{\Rightarrow}} q$ follows obviously. By $\bar{p}\in\varrho'(p'_1)$ also $f(\bar{p})\in\varrho'(f(p'_1, \ldots, p'_K))$. Thus, $p=f(\bar{p})\in\sigma_{\mathbf{a}}^{-1}([r])$.

If $k>0$ then there are trees $p_1, \ldots, p_k\in\sigma_{\mathbf{c}}^{-1}([r_1])$ with $b_1 p_1 \underset{\mathbf{A}}{\overset{*}{\Rightarrow}} q_1, \ldots, b_k p_k \underset{\mathbf{A}}{\overset{*}{\Rightarrow}} q_k$. From this, by an application of Lemma 6, it follows that there is a tree $\bar{p}\in\sigma_{\mathbf{c}}^{-1}([r_1])$ with $b_1\bar{p} \underset{\mathbf{A}}{\overset{*}{\Rightarrow}} q_1, \ldots, b_k\bar{p} \underset{\mathbf{A}}{\overset{*}{\Rightarrow}} q_k$. Put $p=f(\bar{p})$. Again, we have $a_1 p \underset{\mathbf{A}}{\overset{*}{\Rightarrow}} q$. On the other hand, $p\in\sigma_{\mathbf{a}}^{-1}([r])$. Indeed, let $\bar{p}_1\in\varrho'(p'_1)$ be arbitrary. Then, as $\sigma_{\mathbf{c}}(\bar{p})\sim\sigma_{\mathbf{c}}(\bar{p}_1)$, $\sigma_{\mathbf{a}}(f(\bar{p}))\sim\sigma_{\mathbf{a}}(f(p_1))$ follows by Lemma 7. By $f(\bar{p}_1)\in\sigma_{\mathbf{a}}^{-1}([r])$ this means that $f(\bar{p})\in\sigma_{\mathbf{a}}^{-1}([r])$.

Now we are ready to state the main result of this section:

Theorem 1. A deterministic top-down tree transducer $\mathbf{A}$ preserves regularity if and only if $(*)$ is not satisfied by $\mathbf{A}$. The regularity preserving property of deterministic top-down transducers is decidable.

Proof. The necessity of the first statement of our Theorem is valid by Lemma 1. To prove the converse suppose that $\mathbf{A}=(F, A, G, a_0, \Sigma)$ does not satisfy condition $(*)$, and take a regular forest $R\subseteq T_F$. R is recognizable by a deterministic tree automaton $\mathbf{B}=(F, B, B_0)$. Without loss of generality we may assume that $\mathbf{B}$ is connected, i.e., for any state $b\in B$ there is a tree $p\in T_F$ with $(p)_\mathbf{B}=b$.

First let B_0 be a singleton set, say $B_0=\{b_0\}$, and take the deterministic top-down tree transducer $\mathbf{A}'=(H, A\times B, G, (a_0, b_0), \Sigma')$ where $H_n=\{(f, b_1, \ldots, b_n)|\ |f\in F_n, b_1, \ldots, b_n\in B\}$ $(n\geqq 0)$

$$\Sigma' = \{(a, b)(f, b_1, \ldots, b_n) \to q((a_1, b_{i_1})x_{i_1}, \ldots, (a_m, b_{i_m})x_{i_m})|$$

$$|m, n \geqq 0, \quad a, a_1, \ldots, a_m\in A, \quad b_1, \ldots, b_n\in B, \quad i_1, \ldots, i_m\in[n],$$

$$af \to q(a_1x_{i_1}, \ldots, a_mx_{i_m})\in\Sigma, \quad b = (f)_\mathbf{B}(b_1, \ldots, b_n)\}.$$

It is not difficult to see that $\tau_\mathbf{A}(R)=\tau_{\mathbf{A}'}(\mathrm{dom}\,\tau_{\mathbf{A}'})$. On the other hand $\mathbf{A}'$ does not satisfy $(*)$. By Lemmata 8 and 9, and the fact that linear top-down transducers with regular look-ahead preserve regularity (cf. [2], [3]), this implies that $\tau_\mathbf{A}(R)$ is regular.

The general case, i.e. when B_0 is arbitrary, is reducible to the previous one. Indeed, if $B=\{b_1, \ldots, b_n\}$ then put $\mathbf{B}_i=(F, B, \{b_i\})$, $R_i=T(\mathbf{B}_i)$ $(i\in[n])$. Obviously, $\tau_\mathbf{A}(R)=\bigcup_{i=1}^{n}\tau_\mathbf{A}(R_i)$. As all the $\tau_\mathbf{A}(R_i)$ are regular and regular forests are closed under union, it follows that $\tau_\mathbf{A}(R)$ is regular, as well.

The second statement of Theorem 1 is a consequence of the first one because it is decidable whether $(*)$ is satisfied by $\mathbf{A}$.

As every uniform deterministic top-down transducer is equivalent to a nondeterministic bottom-up transducer, by the characterization theorem for regularity preserving bottom-up transducers in [4], it follows that a uniform deterministic top-down transducer preserves regularity if and only if it is equivalent to a linear bottom-up transducer. In general, we do not know any similar characterization for regularity preserving deterministic top-down transducers.

3. Nondeterministic top-down tree transducers

In this section we prove

Theorem 2. The regularity preserving property of nondeterministic top-down tree transducers is undecidable.

Proof. Let H be an arbitrary type containing unary operational symbols only. Take a Post Correspondence Problem $(\boldsymbol{\alpha}, \boldsymbol{\beta})$ $(\boldsymbol{\alpha}, \boldsymbol{\beta}\in H^{+m}, m>0)$ and choose l in such a way that $|\alpha_i|, |\beta_i|<l$ $(i\in[m])$. Set $F_0=\{\#\}$, $F_1=[m]$ $([m]\cap H=\emptyset)$, $F=F_0\cup F_1$, $G_0=F_0$, $G_1=F_1\cup H\cup\{f\}$ $(f\notin F_1\cup H)$, $G_2=\{g\}$, $G=G_0\cup G_1\cup G_2$. We shall give a top-down tree transformation $\tau\subseteq T_F\times T_G$ such that τ preserves regularity if and only if $(\boldsymbol{\alpha}, \boldsymbol{\beta})$ has no solution.

Consider the top-down transducer $\mathbf{A}_1=(F, \{a_0, a_1, a_2, b_1, b_2, b_3\}, G, a_0, \Sigma)$ with Σ consisiting of the rules from (1) to (8) where $i\in[m]$:

(1) $a_0 i \to a_0 x_1$,

(2) $a_0 i \to g(f(a_1 x_1), \alpha_i(b_1 x_1))$, [2]

$a_0 i \to g(f(a_1 x_1), w(b_2 x_1))$ $(w\in H^*, |w| \leqq |\alpha_i|, w \neq \alpha_i)$,

(3) $a_1 i \to f(a_1 x_1)$, $a_1 \# \to \#$,

(4) $b_1 i \to \alpha_i(b_1 x_1)$, $b_1 i \to w(b_2 x_1)$ $(w\in H^*, |w| \leqq \alpha_i, w \neq \alpha_i)$,

(5) $b_2 i \to w(b_2 x_1)$ $(w\in H^*, |w| \leqq \alpha_i, w \neq \alpha_i)$, $b_2 \# \to \#$,

(6) $a_0 i \to g(a_2 x_1, w(b_3 x_1))$ $(w\in H^*, 1 \leqq |w| \leqq l)$,

$a_0 i \to g(f(a_2 x_1), w(b_3 x_1))$ $(w\in H^*, |\alpha_i| < |w| \leqq l)$,

(7) $a_2 i \to a_2 x_1$, $a_2 i \to f(a_2 x_1)$, $a_2 \# \to \#$,

(8) $b_3 i \to w(b_3 x_1)$ $(w\in H^*, |\alpha_i| \leqq |w| \leqq l)$, $b_3 \# \to \#$.

Denote $\tau_{\mathbf{A}_1}$ by τ_1. It can be seen that τ_1 consists of all pairs $(i_1 \dots i_k(\#), g(f^{k-j}(\#), w(\#)))$ where $k\geqq 1$, $0\leqq j\leqq k$, $w\in H^*$, $0\leqq|w|\leqq kl$ and $w\neq\alpha_{i_{j+1}} \dots \alpha_{i_k}$. Similarly, a top-down tree transducer $\mathbf{A}_2$ inducing τ_2 can be constructed with τ_2 containing the same pairs as τ_1 with the exception that $w\neq\beta_{i_{j+1}}\dots\beta_{i_k}$. Taking the disjoint sum of $\mathbf{A}_1$ and $\mathbf{A}_2$ we obtain a top-down transducer $\mathbf{A}$ inducing $\tau=\tau_1\cup\tau_2$.

Assume that (α, β) has a solution. Then let $i_1 \dots i_k$ be a solution to (α, β) with minimal length. Put $L=\{(i_1 \dots i_k)^n(\#) | n\geqq 0\}$, $w=\alpha_{i_1}\dots\alpha_{i_k} (=\beta_{i_1}\dots\beta_{i_k})$, $T=\overline{\tau(L)}\cap \cap\{g(f^r(\#), v(\#)) | r\geqq 0, v\in H^*\}$, $R=\{g(f^{kn}(\#), w^n(\#)) | n\geqq 0\}$. We are going to show that $T=R$. As the class of regular forests is closed under complementation and meet, furthermore, the forest $\{g(f^r(\#), v(\#)) | r\geqq 0, v\in H^*\}$ is regular while R is not, from this follows that $\tau(L)$ is not regular. Since L is regular this implies that τ does not preserve regularity.

Suppose that $g(f^{kn}(\#), w^n(\#))\in\tau(L)$. Then there exists an integer r $(0\leqq n\leqq r)$ with $g(f^{kn}(\#), w^n(\#))\in\tau((i_1 \dots i_k)^n(\#))$. Therefore, either $w^n \neq (\alpha_{i_1}\dots\alpha_{i_k})^n$ or $w^n\neq(\beta_{i_1}\dots\beta_{i_k})^n$. As $i_1 \dots i_k$ is a solution to (α, β) both cases yield a contradiction. Thus, $R\subseteq T$. To prove the converse suppose that $g(f^r(\#), v(\#))\notin\{g(f^{kn}(\#), w^n(\#)) | n\geqq 0\}$ $(r\geqq 0, v\in H^*)$. Let $n\geqq\max\{r, |v|/l\}$ be the least integer divisible by k, $j_1 \dots j_n=(i_1 \dots i_k)^{n/k}$. If r is a multiple of k, say $r=kt$, then $v\neq w^t$, i.e. $v\neq\alpha_{j_{r+1}}\dots\alpha_{j_n}$. If r is not a multiple of k then, as $i_1 \dots i_k$ was a minimal solution to (α, β), $j_{r+1} \dots j_n$ is not a solution to (α, β). Therefore, either $v\neq\alpha_{j_{r+1}}\dots\alpha_{j_n}$ or

[2] If F is a unary type and $v=f_1 \dots f_k\in F^*$ then we denote by v the tree $f_1(\dots(f_k(x_1))\dots)\in T_{F,1}$ as well.

$v \neq \beta_{j_{r+1}} \ldots \beta_{j_n}$. Moreover, as $n \geqq |v|/l$, in both cases $|v| \leqq ln$. This together with $n > 0$ means that $g(f^r(\#), v(\#)) \in \tau(j_1 \ldots j_n(\#)) \subseteq \tau(L)$, as was to be proved.

Next assume that (α, β) has no solution. Then $\tau(L) = \{g(f^r(\#), v(\#)) | r \geqq 0, v \in H^*\} - \{g(\#, \#)\}$ holds for any infinite $L \subseteq T_F$. Consequently, A preserves regularity.

DEPT. OF COMPUTER SCIENCE
A. JÓZSEF UNIVERSITY
ARADI VÉRTANÚK TERE 1
SZEGED, HUNGARY
H-6720

References

[1] ÉSIK, Z., Decidability results concerning tree transducers I., *Acta Cybernet.*, v. 5, 1980, pp. 1—20.
[2] ENGELFRIET, J., Bottom-up and top-down tree transformations, A comparison, *Math. Systems Theory*, v. 9, 1975, pp. 198—231.
[3] ENGELFRIET, J., Top-down tree transducers with regular lookahead, *Math. Systems Theory*, v. 10, 1977, pp. 289—303.
[4] GÉCSEG, F., Tree transformations preserving recognizability, Proc., Finite algebra and multiple-valued logic, North Holland, 1981, pp. 251—273.

(Received Feb. 7, 1983)

An implementation of the HLP

By T. GYIMÓTHY*, E. SIMON*, Á. MAKAY**

Introduction

The Helsinki Language Processor (HLP) system was designed originally [7] for description of programming languages and for automatic generation of compilers. Saving the descriptional metalanguages different implementations have a great freedom with respect to the applications of parsing, semantic evaluation and software generation technics. Our implementation chooses SIMULA 67 [1] as base language which influences the collection of semantic functions usable for description of semantic features and the structure of the generated compiler too.

This text follows the steps of the generating process. A source language L is assumed which has a lexical description on the lexical metalanguage and a syntactic-semantic description on that metalanguage of the HLP.

There are two hand-written lexical analyzers for the metalanguages. One of them receives the lexical description of L and produces the input for the generator of the lexical analyzer of L. This will be constructed as a finite automaton. The other works on the syntactic-semantic description of L fundamentally in the form of an attribute grammar, producing the input for the semantic evaluator and for the pure syntax constructor. Because there may be different token class names and terminal strings in the lexical and syntactical description, unification of the symbol table of the generated lexical analyzer must be executed after that two lexical analysis. In the semantic description of L we can use attributes as SIMULA types involving simple types, classes, expressions, functions, statements and predefined standard procedures.

Having the pure syntax of L, the parser generator checks the grammar being of type LR (1) [2]. If it is so it constructs the table of the optimal parser of type LR (1), LALR (1), SLR (1) or LR (0).

We can choose one of the modified strategies ASE [3] or OAG [4] for computing the necessary passes and order of the evaluation of the attribute values in the generated compiler. For each syntax rule a SIMULA class is constructed, which contains the actions of parsing and evaluation decomposed to passes, anywhere this rule is applied in a derivation. One-pass compilation is possible if we have only synthesized attributes and it means, that the values of all attribute occurences are evaluable

during parsing bottom up. This is a sufficient condition and so a proposition with respect to the formulation of the grammar.

By this means we have all components of the combined parser and semantic evaluator. Working under the control of the parsing table new objects of types predefined in the above SIMULA classes are created and connected as the derivation tree. Subsequent passes are executed by reactivating and deactivating the objects as the inner structures of the classes prescibe the evaluation order.

The generators have the same structure as the generated compiler so there are possibilities to generate new variations of the lexical analysers and the parser by the system itself. We have written these parts of the HLP in the metalanguages of its own.

Structure of the generated compiler

The nucleus of the generated compiler (GC) consists of a parser based on a grammar G from the class or subclass of the LR (1) grammars. It constructs the derivation tree in the grammar G from the token stream produced from the incoming text $p \in L(G)$ by the generated lexical analyzer GL. The nodes of the derivation tree are the SIMULA objects of types (SIMULA classes) representing the rewriting rules in the grammar G. Local pointers inside the objects ensure the connections — edges — toward the nodes on a lower level of the tree.

The objects contain the local variables of the attribute occurrences too together with the calling sequence, which represents the attribute evaluation strategy predefined from the attribute dependencies of the grammar G by one of the algorithms ASE or OAG. During parsing, when a new object is activated not only a new node is generated in the derivation tree (bottom up) but those attributes are evaluated, which depend on previously evaluated attributes. After that the object — the procedural part of the object — detaches itself while accessing the contents of the variables of the attribute occurrences just evaluated is possible. These are usable by the objects on a higher level of the derivation tree. Reactivating an object a new package of attribute occurences not evaluated yet is evaluable. Of course during evaluation this object activates other objects too going up or down in the tree in the order of the strategy. After finite number of activating-deactivating action pairs an object together with all the objects on the lower levels have no attribute occurences not evaluated. This part of the tree is unnecessary so it is destroyed. Finally we have only the root of the derivation tree together with one or more attributes of the initial nonterminal of the grammar G. Generally these attributes serve the purposes of the target code generation.

Of course we can describe and so generate not only a compiler for a programming language by an attribute grammar — as the metalanguage of the system — but other special purpose systems based on context-free languages too: schemes of data bases, machine architectures, picture description and processing, and so on. The common feature of these tasks is, that there exist a class of very similar algorithms, each of which we can specify by a context-free grammar together with several special attributes. The result is, that we have a generated software system specialized to one task only and the gain is in time or space complexity. It is the case of a compiler too: GC has a parser for one grammar and one strategy for the evaluation of a given attribute set.

Although it is possible to describe the generation of the target code by an attribute in the metalanguage too, we recommend a final pass for it based on the other attribute values evaluated earlier. Several procedures well defined for this purpose can help the users in that — target language dependent — job. So far we have neglected this aspect because we need experiences in large-sized and complicated languages.

Lexical metalanguage

The lexical metalanguage is used to describe the lexical structure of the source language for automatic construction of the lexical analyzer which forms tokens from the character strings of the source program. A description on the lexical metalanguage consists of five parts. In the first part a collection of *character sets* is defined. Specification of *token classes* by regular expressions can be found in the second part. The description of *transformations* concerns characters and token classes too. Transformations are performed during the isolation of a character or tokens. In *action blocks* the scanning sequence, screening of keywords from token classes and the way the isolated tokens are sent to the syntax analyzer, are given.

To give an idea of what a lexical description looks like we refer to the description of a simple block structured language called BLOCK_HLP given in Appendix.

The syntactic and semantic metalanguage

The definition of an attribute grammar is divided into five parts. First the *inherited* and *synthesized attributes* must be defined by SIMULA types. It should be noted that the concept of global attributes was not implemented. Global attributes can be replaced by SIMULA objects. In *nonterminal declaration* those nonterminals are declared which appear in the production list as the left-hand side of at least one production. Each nonterminal declaration has a possibly empty attribute list associated with it. An attribute from this list is associated with all nonterminals appering in the nonterminal list. The third part of the description is the declaration of the *start symbol.* We assume the grammar to be reduced. The auxiliary SIMULA variables, classes, functions and procedures which are used in the semantic rules and code generation are declared in the *procedure declaration* part.

As in the original HLP system we employ BNF (Backus Naur Form) description method for the syntax of the source language. *Semantic rules* and *code generation* are built in the *productions.* Note that if the semantic part is empty for one production, then the use of ECF (Extended Context Free) description [5] on the right-hand side is allowable. According to the SIMULA features the original notation of an attribute occurence is modified. In a production, if an attribute is associated to the left-hand side nonterminal, then only the attribute name must occur. Other attribute occurences are denoted by

nonterminal name · attribute name.

In Appendix the syntactic semantic description of the language BLOCK_HLP can be found too.

Attribute grammars

An attribute grammar (AG) can be considered as an extension of a context free (CF) grammar with attributes and semantic rules defining values of attributes. These attributes serve to describe the semantic features of the language elements.

An AG is a 3-tuple

$$AG = (G, A, F),$$

where $G=(V_N, V_T, P, S)$ is a reduced CF grammar, V_N, V_T, P and S denote the nonterminals, terminals, productions and the start symbol of the grammar respectively.

A production $p\in P$ has the form

$$p\colon X_0 \to X_1 \ldots X_{n_p}, \quad \text{where} \quad n_p \geqq 0, \quad X_0 \in V_N, \quad X_i \in V_N \cup V_T \quad (1 \leqq i \leqq n_p).$$

The finite set A is the set of attributes. There is a fixed set $A(X)$ associated with each nonterminal $X\in V_N$ denoting the attributes of X. For an $X\in V_N$, $p\in P$ and $a\in A(X)$ $X\cdot a$ denotes an attribute occurrence in p. An attribute can be either inherited or synthesized, so each $A(X)$ is partitioned into two disjoint subsets, $I(X)$ and $S(X)$.

The set

$$A_p = \bigcup_{i=0}^{n_p} \bigcup_{a \in A(X_i)} X_i \cdot a$$

denotes all attribute occurrences in a syntactic rule p.

The set F consists of semantic rules associated with syntactic rules too. A semantic rule is a function type defined on attribute occurrences as argument types. For each attribute we have a set of attribute values (the domain of attribute) and for each semantic rule a semantic function defined on the sets which are related to its type. Formally, let us denote by F_p the rules associated with syntactic rule p, then

$$F = \bigcup_{p \in P} F_p .$$

We classify the set A_p into an output attribute occurrence set

$$\mathrm{OA}_p = \{X_i \cdot a \mid (i = 0 \quad \text{and} \quad a \in S(X_i)) \quad \text{or} \quad (i > 0 \quad \text{and} \quad a \in I(X_i))\}$$

and an input attribute occurrence set [6].

$$\mathrm{IA}_p = A_p - \mathrm{OA}_p$$

We assume, that for each $X_i\cdot a\in \mathrm{OA}_p$ there is exactly one semantic rule $f\in F_p$ the function related to it defines the value of $X_i\cdot a$. An AG is in normal form provided that only input occurrences appear as arguments of the semantic rules.

Evaluation of attribute values

Denote by t a derivation tree in the grammar G. If a node of t is labeled by X, then we can augment it by the attribute occurrences of X and their semantic rules defined by two syntactic rules. One of these is applied on the level over X in t and defines the inherited occurrences, while the other on the level under X determines the synthesized ones. Naturally, the root has no inherited and the leaves have no synthesized occurrences. (Leaves have attributes defined by the lexical analyzer which can be considered as synthesized ones.) A rule p, so an attribute occurrence may occur several times in t. We distinguish them and if it would be confusing we say occurrence in a tree t. Denote by T_{AG} the set of the augmented derivation trees in AG.

Let be given the set of semantic functions $\tilde{F}$ associated with F. The value of each attribute occurrence in t is computed by one of these functions and it is computable only if the argument values are computed. Therefore we have dependencies among attribute occurrences in the tree t. We denote by $(X_i \cdot a \rightarrow X_j \cdot b)$ the fact, that the function defining the value of $X_j \cdot b$ in t has the value of $X_i \cdot a$ as an argument. We say that $X_j \cdot b$ depends on $X_i \cdot a$ in t. By this relation the set F and the tree t induce a dependency graph D_t. If D_t has no cycle it determines an evaluation order for the computation of the values of all attribute occurrences in t. An attribute grammar is noncircular if there is no derivation tree with dependency graph containing a cycle. The decision whether an AG is noncircular requires algorithms of exponential complexity.

To determine the dependency graph to each derivation is time-consuming during compilation. For several subclasses of AG's it is possible to determine an evaluation strategy based on the grammar only. Such a strategy consists of an ordering of the attribute occurrences in the rules of the grammar in the form of a dependency graph D and means, that wherever an occurrence $X \cdot a$ appears in any tree t, it is computable if the occurrences on which it depends by D are already evaluated. The problem is determine D from the AG. During compilation we have to follow the evaluation order defined by D for each derivation tree. Naturally it is a tree traversal strategy and one travers may be seen as a pass of the compilation.

Two subclasses of AG's are considered in our system in accordance with them two algorithms, ASE and OAG serve to generate evaluation strategies.

ASE

The ASE algorithm is based on a fixed tree traversal strategy. An AG is ASE if any $t \in T_{AG}$ is evaluable during m alternating depht-first, left-to-right (L—R) and depht-first right-to-left (R—L) tree traversal passes.

The attribute evaluation during an L—R traversal can be illustrated for a syntactic rule $p: X_0 \rightarrow X_1 \ldots X_{n_p}$ as follows.

PROCEDURE TRAVERSE (X_0);
BEGIN
FOR $i := 1$ **STEP** 1 **UNTIL** n_p **DO**

BEGIN EVAL $(I(X_i))$; TRAVERSE (X_i) **END;**
EVAL $(S(X_0))$;
END OF TRAVERSE;

During an R—L pass the **FOR** statement above has the form
FOR $i:=n_p$ **STEP** -1 **UNTIL** 1 **DO**

The ASE algorithm makes a membership test for an AG by this traversal procedure, and assigns attributes to passes. By the EVAL procedure we denoted the computation of the values of the attributes. The different instances of the same attribute is evaluated during the same pass.

Our experiences show that the ASE subclass is large enough and can be applied well in a compiler writing system. But it needs some modification in the original algorithm to use it in a practical system. For example we need not traverse a subtree during the ith pass if there is no evaluable attribute in this subtree. It can be decided by the following test.

Denote $H(X)$ the set of nonterminals which can be derived from an $X \in V_N$. It is easy to generate these sets by the transitive closure using P.

Let $K(X)= \bigcup_{Y \in H(X)} A(Y)$, and denote by A_j the set of attributes which can be evaluated during the jth pass.

If $(K(X) \cup S(X)) \cap A_j = \varnothing$, then for an $X_i = X$ we will not call the TRAVERSE (X_i) during the jth pass.

In the ASE algorithm the tree traversal and the attribute evaluation starts from the root of the derivation tree. In our system we use bottom-up tree constructor and many synthesized attributes can be evaluated interleaved with the construction of the derivation tree. These synthesized attributes can be easily assigned by the TRAVERSE procedure often decreasing the number of evaluation passes of an AG.

We can ensure an efficient space management technique for a generated compiler by using an extended version of ASE algorithm. We test for each $p \in P$ whether after the ith pass the attributes of the subtrees which can be derived from p are computed or not. If each of them are computed we generate a statement for the rule p which releases these subtrees. This technique is based on a garbage collector and is very efficient, because large parts of a derivation tree are released during the construction of the tree.

The ASE algorithm is pessimistic in the sense that it considers all dependencies for an attribute a. E.g. there are dependencies for an attribute a in the rules p and q, but there is no derivation tree containing the rules p and q together. Generally this does not occur in practical programming languages but it causes problems in some types of languages. Whether there is a derivation tree containing the rules p and q together may be decided by a simple algorithm using the sets $H(x)$.

OAG

In this section we give a short description of the OAG algorithm using some notations of [4]. We modified this algorithm, so an attribute evaluation strategy is given for a larger subclass of noncircular AG's. The time needed for the modified algorithm does not significantly differ from the time needed for the original algorithm.

As opposed to ASE algorithm in the OAG algorithm there is not a predefined tree traversal strategy. For each AG$\in$OAG an attribute evaluation strategy is generated, and all derivation trees of the AG can be evaluated by this strategy. The OAG algorithm for each $X \in V_N$ constructs a partial order over the set $A(X)$, such that in any derivation tree containing X its attributes are evaluable in that order.

Denote by $\mathrm{DS}(X)$ the partial order over the $A(X)$, and let

$$\mathrm{DS} = \bigcup_{X \in V_N} \mathrm{DS}(X)$$

be the set of these partial orders.

We define dependency graphs over the attribute occurrences of syntactic rules and over the attributes of nonterminals, finally we construct DS using these graphs.

The dependency graph DP_p contains the direct dependencies between attribute occurrences associated to a syntactic rule p.

$$\mathrm{DP}_p = \{(X_i \cdot a \to X_j \cdot b) \mid \text{there is an } f \in F_p \text{ defining } X_j \cdot b \text{ depending on } X_i \cdot a\}$$

$$\mathrm{DP} = \bigcup_{p \in P} \mathrm{DP}_p$$

The dependency graph IDP can be constructed from the DP

$$\mathrm{IDP}_p = \mathrm{DP}_p \cup \{(X_i \cdot a \to X_i \cdot b) | X_i \text{ occurs in rules } p \text{ and } q,\ (X_i \cdot a \to X_i \cdot b) \in \mathrm{IDP}_q^+\},$$

where IDP_q^+ denotes the nonreflexive, transitive closure of IDP_q.

$$\mathrm{IDP} = \bigcup_{p \in P} \mathrm{IDP}_p$$

The graph IDP comprises the direct and indirect dependencies of attribute occurrences. For an $X \in V_N$ the dependency graph $\mathrm{IDS}(X)$ contains the induced dependencies between attributes of X

$$\mathrm{IDS}(X) = \{(X \cdot a \to X \cdot b) \mid \text{there is an } X_i = X \text{ in a rule } p \text{ and } (X_i \cdot a \to X_i \cdot b) \in \mathrm{IDP}_p\}$$

$$\mathrm{IDS} = \bigcup_{X \in V_N} \mathrm{IDS}(X).$$

The set DS can be constructed using IDS. For an $X \in V_N$ the set $A(X)$ is partitioned into disjoint subsets $A(X)_i$, and $\mathrm{DS}(X)$ defines a linear ordering over these subsets. The sets $A(X)_i$ are determined such that for an $a \in A(X)_i$ if $(X \cdot a \to X \cdot b) \in \mathrm{IDS}(X)$ and $b \in A(X)_k$, then $k \leqq i$. The sets $A(X)_i$ consist of either synthesized or inherited attributes only. The $\mathrm{DS}(X)$ defines an alternating sequence of the synthesized and inherited sets $A(X)_i$.

$$\mathrm{DS}(X) = \mathrm{IDS}(X) \cup \{(X \cdot a \to X \cdot b) | X \cdot a \in A(X)_k, X \cdot b \in A(X)_{k-1}, 2 \leqq k \leqq m_x\},$$

where m_x is the number of the sets $A(X)_i$. The extended dependency graph EDP is defined by IDP and DS.

$$\mathrm{EDP}_p = \mathrm{IDP}_p \cup \{(X_i \cdot a \to X_i \cdot b) | (X \cdot a \to X \cdot b) \in \mathrm{DS}(X), \quad X_i = X \text{ and } X_i \text{ occurs in rule } p\}$$

$$\mathrm{EDP} = \bigcup_{p \in P} \mathrm{EDP}_p.$$

A given AG is an OAG iff the EDP is noncircular. We implemented the OAG algorithm as a part of our compiler writing system. We have favourable experiences using the algorithm, but we have found simple attribute grammars (occurring in practical applications, see Fig. 1), where the IDP is noncircular but the EDP is circular. We modified the OAG algorithm so that in these cases we generate a new EDP.

The graphs DP, IDP, IDS, DS are computed using the original algorithm. In the next step for each $X \in V_N$ and $(X \cdot a \to X \cdot b) \in \mathrm{DS}(X)$—$\mathrm{IDS}(X)$ we add $(X \cdot a \to X \cdot b)$ to IDP_p, if X occurs in rule p, and construct IDP_p^+. If a $(Y \cdot c \to Y \cdot d)$ is induced in IDP_p^+, then

(a) if $(Y \cdot d \to Y \cdot c) \in \mathrm{DS}(Y)$—$\mathrm{IDS}(Y)$, then we add $(Y \cdot c \to Y \cdot d)$ to $\mathrm{IDS}(Y)$ and generate a new $\mathrm{DS}(Y)$ using the modified $\mathrm{IDS}(Y)$,

(b) if $(Y \cdot d \to Y \cdot c) \in \mathrm{IDS}(Y)$, then the algorithm is finished and the given AG is not an OAG,

(c) otherwise we have $(Y \cdot c \to Y \cdot d)$ out of consideration.

If each $(X \cdot a \to X \cdot b) \in \mathrm{DS}(X)$—$\mathrm{IDS}(X)$ is added for an $X \in V_N$, then the set $\mathrm{DS}(X)$ is not changed later on.

In Fig. 1 we show an AG which is neither ASE nor OAG but for which an attribute evaluation strategy can be generated using the modified OAG algorithm. We denote by ○ an inherited attribute and by ● a synthesized one.

The dependencies in rule 2 show that AG $\notin$ ASE. We construct the sets $A(Y)_i$ and $A(Z)_i$ using the rules 1, 3, 5. The sets $A(Y)_1 = \varnothing$, $A(Y)_2 = \{e, g\}$, $A(Y)_3 = \{f\}$ and $A(Z)_1 = \{f\}$, $A(Z)_2 = \{e\}$ imply that $(Y \cdot f \to Y \cdot e) \in \mathrm{DS}(Y)$ and $(Z \cdot e \to Z \cdot f) \in \mathrm{DS}(Z)$. If we construct EDP_3 by $\mathrm{DS}(Y)$ and $\mathrm{DS}(Z)$ it will be circular, so AG $\notin$ OAG. Using the modified algorithm, if we add $(Y \cdot f \to Y \cdot e)$ to IDP_3, then $(Z \cdot f \to Z \cdot e)$ is induced in $\mathrm{IDS}(Z)$. The new $\mathrm{DS}(Z)$ is constructed from the sets $A(Z)_1 = \varnothing$, $A(Z)_2 = \{e\}$, $A(Z)_3 = \{f\}$ and the EDP is generated by this $\mathrm{DS}(Z)$ will be noncircular. It is easy to prove that for an AG $\in$ OAG the modified algorithm does not change the set DS and graph EDP. The OAG algorithm for each $p \in P$ generates a visit-sequence VS_p using the graph EDP_p. Each VS_p is linear sequence of node visits and attribute evaluations and it is easy to generate an attribute evaluation strategy using the sets VS_p.

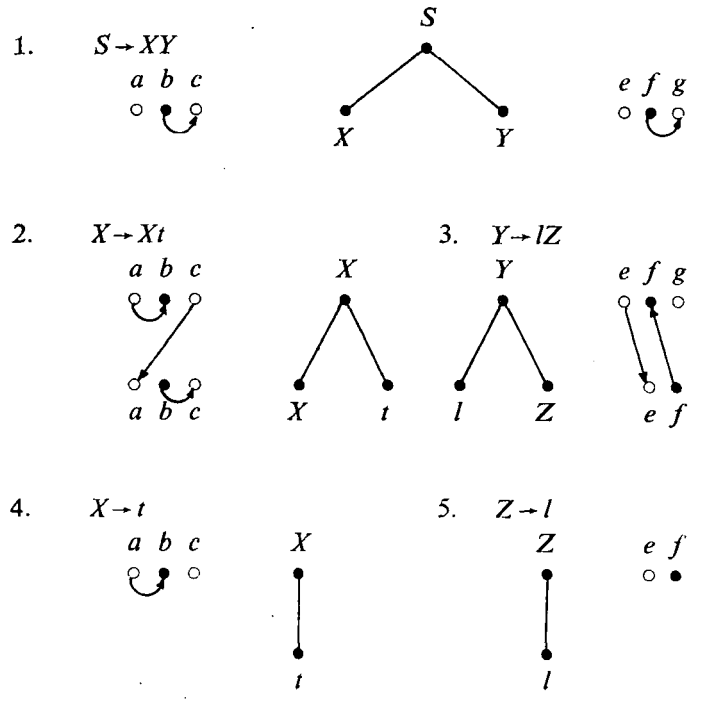


Fig. 1

Construction of the parser

In the present paragraph the logical description of parsing automata constructor modul is given. This modul serves to compute the state transitions for finite state and stack automaton too. The definition of the token classes by regular expressions and the description of an ECF grammar are coded uniform manner. Consequently the procedure which computes the parsing states can be controlled at the job control level to generate finite, ELR (1), ELALR (1), ESLR (1) or ELR (0) [2] states too. The states are represented by SIMULA objects based on the following declarations.

```
CLASS ITEM (NO, DOT, RSET);
INTEGER NO, DOT; REF (SET) RSET;
  BEGIN
  REF (ITEM) LINK;
  END ITEM;
CLASS SET (BOUND);
INTEGER BOUND;
  BEGIN INTEGER ARRAY TSET [0: BOUND];
  END SET;
```

It is easy to see that this representation has two advantages. The finite and LR (0) states which have no *follower set* can be stored uniform manner. Secondly, those

items which have the same follower sets store only one SIMULA reference to an object in which the followers have been written. If the computed state is equal to a state which has been computed earlier then the SIMULA run-time system releases the space by calling the garbage collector. For each computed state there is a table which contains a set of ordered pairs. The first element describes the state number from which this state has been derived. The second element contains the symbol code used to compute the considered state.

By applying the next theorem from [2] to our parser constructor we can obtaine an useful conclusion. An ELR (0) language can be parsed by a finite state automaton iff there is no state which can be derived by a nonterminal from more than one state. Hence, in order to generate a finite state automaton the ELR (0) states are computed first. It is followed by performing the *finite state test.*

After computing the selected type of states (ELR (1), ELALR (1), ESLR (1) and ELR (0)) a membership test will be performed together with parser code generation. If it produces true then the next type of states will be computed from the last states. The test are performed from ELR (1) to ELR (0). Some simple optimization procedure are executed during the tests. In the present version of our implementation there is no automatic error recovery procedure. Ordering of states on the base [8] an efficient error correcting algorithm is under development.

The states and the internal code of the lexical analyzer as a finite automaton are generated by the same modul. Of course we need additional service routines working in the lexical analyzers. These are written for metalanguage purposes, but they can be used in the generated compilers in the same form too.

Appendix

```
% LEXICAL DESCRIPTION FOR A SIMPLE BLOCK STRUCTURED
% LANGUAGE
% CALLED BLOCK HLP
LEXICAL DESCRIPTION BLOCK HLP

CHARACTER SETS
   LETTER_OR_DIGIT=LETTER/DIGIT;
END OF CHARACTER SETS
TOKEN CLASSES
   UNDERSCORE = ≠ ≠;
   IDENTIFIER=LETTER (LETTER_OR_DIGIT/UNDERSCORE)* [16];
   PROPERTY  =DIGIT+ [2];
   COMMENT   =≠%≠ ANY* ENDOFLINE;
   SPACES    =SPACE* ENDOFLINE;
   SPACES    =SPACE+;
END OF TOKEN CLASSES
   TRANSFORMATIONS ARE
   UNDERSCORE=>;
   END OF TRANSFORMATIONS
ACT_BLOCK: BEGIN
   IDENTIFIER=>IDENTIFIER / KEYSTRINGS;
   PROPERTY  =>PROPERTY;
   COMMENT   =>;
   SPACES    =>;
END OF ACT_BLOCK
```

```
END OF LEXICAL DESCRIPTION BLOCK HLP.
   FINIS
% SYNTACTIC-SEMANTIC DESCRIPTION OF BLOCK HLP
ATTRIBUTE GRAMMAR BLOCKHLP

SYNTHESIZED ATTRIBUTES ARE
    REF (SYMB) SYMREF; REF (SDECL) SEREF;
    INTEGER ID, TYPE, EXTYPE;
END OF SYNTHESIZED ATTRIBUTES
INHERITED ATTRIBUTES ARE
    REF (SBL) SYMT;
END OF INHERITED ATTRIBUTES
NONTERMINALS ARE
    PROGRAM;
    BLOCK HAS SYMT, SYMREF;
    STATLIST HAS SYMT, SYMREF;
    STAT HAS SYMT, SEREF;
    IDECL HAS ID, TYPE;
    EXDECL HAS SYMT, EXTYPE;
END OF NONTERMINALS

% PROCEDURES AND CLASSES

$$$$

    CLASS SBL (A, B);
    REF (SBL) A; REF (SYMB) B;
    BEGIN
    END OF SBL;

    CLASS SYMB (A, B);
    REF (SYMB)A; REF (SDECL)B;
    BEGIN
    END OF SYMB;
    CLASS SDECL (A, B);
    INTEGER A, B;
    BEGIN
    END OF SDECL;

    PROCEDURE FIND (A, B, C);
    NAME A;
    INTEGER A, B; REF (SBL) C;
    BEGIN

% The value of A will be the type of the variable B. This type is tried to find
% in the list of identifiers defined by C.B. If B is not found in it, then C is replaced
% by C.A. Repeating until having the type of B or being the list empty, the
% requested value is done or A is undeclared.

    END OF FIND

****

% END OF PROCEDURES AND CLASSES
   PRODUCTIONS ARE
%1%
```

```
PROGRAM = BLOCK;
DO
    BLOCK.SYMT :— NEW SBL (NONE, BLOCK.SYMREF);
END
%2%
BLOCK = STATLIST;
%3%
STATLIST = STATLIST STAT;
DO
    SYMREF :— IF STAT.SEREF=/= NONE THEN
                NEW SYMB (STATLIST.SYMREF, STAT.SEREF) ELSE
               STATLIST.SYMREF;
END
%4%
STATLIST = STAT;
DO
    SYMREF :— IF STAT.SEREF == NONE THEN NONE
                ELSE NEW SYMB (NONE, STAT.SEREF);
END
%5%
STAT = IDECL;
DO
    SEREF :— NEW SDECL (IDECL.ID,IDECL.TYPE);
END
%6%
STAT = EXDECL;
DO
    SEREF :— NONE;
END
%7%
STAT = BEGIN BLOCK END;
DO
    BLOCK.SYMT :— NEW SBL (SYMT,BLOCK.SYMREF);
    SEREF :— NONE;
END
%8%
IDECL = DECLARE IDENTIFIER PROPERTY;
DO
    ID := IDENTIFIER.VALUE;
    TYPE := PROPERTY.VALUE;
END
%9%
EXDECL = USE IDENTIFIER;
DO
```

```
    EXTYPE <=FIND (EXTYPE,IDENTIFIER.VALUE,EXDECL.SYMT);
END
END OF PRODUCTIONS
END OF ATTRIBUTE GRAMMAR

%  SIMULA classes associated with two nonterminals and a production in the
%  generated compiler
NODE        CLASS GRNODE 1;
            BEGIN COMMENT PROGRAM;
            END;
NODE        CLASS GRNODE 2;
            BEGIN COMMENT BLOCK;
            REF (SBL) SYMT;
            REF (SYMB) SYMREF;
            END ;
GRNODE 1    CLASS P 1;
            BEGIN COMMENT PROGRAM;
            REF (GRNODE 2) BLOCK;
           BLOCK: — POP QUA GRNODE 2;
           PUSH (GOTO (1), THIS NODE);
           DETACH;
           BLOCK.SYMT:— NEW SBL (NONE, BLOCK.SYMREF);
           CALL (BLOCK);
           DETACH;
            END ;
```

*RESEARCH GROUP ON THEORY OF AUTOMATA
HUNGARIAN ACADEMY OF SCIENCES
SOMOGYI U. 7
SZEGED, HUNGARY
H-6720

**DEPT. OF COMPUTER SCIENCE
A. JÓZSEF UNIVERSITY
ARADI VÉRTANÚK TERE 1
SZEGED, HUNGARY
H-6720

References

[1] DAHL, O. J., B. MYHRHAUG and K. NYGAARD, SIMULA 67 common base language, Norwegian Computing Center, Publication No. S-2, May 1968.
[2] HEILBRUNNER, S., A parsing automata approach to LR theory, *Theoret. Comput. Sci.*, v. 15, 1981, pp. 117—157.
[3] JAZAYERI, M. and K. G. WALTER, Alternating semantic evaluator, Proc. of the ACM 1975. Annual Conference, Oct. 1975, pp. 230—234.
[4] KASTENS, U., Ordered attribute grammars, *Acta Inform.*, v. 13, 1980, pp. 229—256.
[5] PURDOM, P. W., JR. BROWN and A. CYNTHIA, Parsing extended LR (k) grammars, *Acta Inform.* v. 15, 1981, pp. 115—127.
[6] RÄIHÄ, K. J., A space management technique for multi-pass attribute evaluators, University of Helsinki, Report A-1981-4, 1981.
[7] RÄIHÄ, K.J., M. SAARINEN, E. SOISALON—SOININEN and M. TIENARI, The compiler writing system HLP (Helsinki Language Processor). University of Helsinki. Report A—1978—2, 1978.
[8] RÖHRICH, J., Methods for the automatic construction for error correcting parsers, *Acta Inform.*, v. 13. 1980, pp. 115—139.

(Received Feb. 7, 1983)

INDEX — TARTALOM

ISSN 0324—721 X

Felelős szerkesztő és kiadó: Gécseg Ferenc
A kézirat a nyomdába érkezett: 1983. április 29
Megjelenés: 1983. december
Példányszám: 500. Terjedelem: 8,92 (A/5) ív
Készült monószedéssel, íves magasnyomással
az MSZ 5601 és az MSZ 5602—55 szabvány szerint
83-2054 — Szegedi Nyomda — F. v.: Dobó József igazgató