Tomus 8. Fasciculus 3.

ACTA CYBERNETICA

FORUM CENTRALE PUBLICATIONUM
CYBERNETICARUM HUNGARICUM

FUNDAVIT: L. KALMÁR

REDIGIT: F. GÉCSEG

COMMISSIO REDACTORUM

A. ÁDÁM
M. ARATÓ
S. CSIBI
B. DÖMÖLKI
B. KREKÓ
Á. MAKAY
D. MUSZKA
ZS. NÁRAY
F. OBÁL
F. PAPP
A. PRÉKOPA
J. SZELEZSÁN
J. SZENTÁGOTHAI
S. SZÉKELY
J. SZÉP
L. VARGA
T. VÁMOS

SECRETARIUS COMMISSIONIS
J. CSIRIK

Szeged, 1988
Curat: Universitas Szegediensis de Attila József nominata

8. kötet 3. füzet

ACTA CYBERNETICA

A HAZAI KIBERNETIKAI KUTATÁSOK
KÖZPONTI PUBLIKÁCIÓS FÓRUMA

ALAPÍTOTTA: KALMÁR LÁSZLÓ

FŐSZERKESZTŐ: GÉCSEG FERENC

A SZERKESZTŐ BIZOTTSÁG TAGJAI

ÁDÁM ANDRÁS
ARATÓ MÁTYÁS
CSIBI SÁNDOR
DÖMÖLKI BÁLINT
KREKÓ BÉLA
MAKAY ÁRPÁD
MUSZKA DÁNIEL
NÁRAY ZSOLT
OBÁL FERENC
PAPP FERENC
PRÉKOPA ANDRÁS
SZELEZSÁN JÁNOS
SZENTÁGOTHAI JÁNOS
SZÉKELY SÁNDOR
SZÉP JENŐ
VARGA LÁSZLÓ
VÁMOS TIBOR

A SZERKESZTŐ BIZOTTSÁG TITKÁRA
CSIRIK JÁNOS

Szeged, 1988
A Szegedi József Attila Tudományegyetem gondozásában

Pure languages of regulated rewriting and their codings[1)]

By J. DASSOW

Dedicated to Prof. Herbert Goering in the occasion of his 60th birthday

0. Introduction

Since context-free grammars are not able to cover all aspects which are of interest (e.g. in the theory of programming languages), a lot of regulating mechanisms for the derivation process have been introduced. However, mostly the generative capacity of these mechanisms has been studied with respect to the generated set of words over a terminal alphabet. Thus all the intermediate steps of the derivation are not contained in the language, and therefore it often happens that the differences between the mechanisms disappear. Hence — in order to contribute to a comparison of the mechanisms — we shall investigate languages which contain also the words of the intermediate steps, the so-called pure languages.

Let us also mention that these languages are of interest for themselves by the following two facts:

— they form a sequential counterpart to the L systems (with regulation),

— the intermediate steps are important for the syntax analysis.

The first results on pure versions of grammars with regulated rewriting are presented in (1), (6), (2). However, besides (6) (where a different definition of the pure language is used) the appearance checking mode is used in the derivation. One of the purposes of this paper is to complete the hierarchy by the relations concerning pure languages of regulated rewriting without appearance checking.

Usual languages (containing only terminal words) can be obtained from a pure language by intersecting with the set of all words over the terminal alphabet. By the results by Ehrenfeucht and Rozenberg it is known that for L systems there is a second way, namely the application of a coding (letter-to-letter homomorphisms), if one is

[1)] Paper presented at the 4th Hungarian Computer Science Conference, Győr, July, 8—10, 1985.

only interested in the generated family of languages (see (7)). This does not hold for sequential rewriting. But in (3) it is shown that the hierarchy of families of languages obtained by codings lies between that of pure languages and that of usual languages. Again, here we shall add some results by consideration of grammars without appearance checking.

In this paper we shall restrict to the following three types of regulated devices: matrix grammars, programmed grammars, and random context grammars.

Throughout the paper we assume that the reader is familiar with the rudiments of formal language theory and has some information on regulated rewriting (e.g. see (8), (5)).

1. Definitions

For the sake of completeness we give the formal definitions for the pure versions of the above mentioned grammars.

In the following definitions, let V be an alphabet, and let S be a finite subset of V^+. (Usually in the theory of pure languages one uses a set of starting words, however, as one can see by our proofs our results do not change if S consists of only one word.)

A *pure random context* grammar is a triple $G=(V, P, S)$ where P is a finite set of productions of the form

$$(a \to w, R, Q),\ a \in V,\ w \in V^*,\ R \subseteq V,\ Q \subseteq V.$$

We say that $x \in V^+$ directly derives $y \in V^*$ (written as $x \Rightarrow y$) iff $x = z_1 a z_2$, $y = z_1 w z_2$, $(a \to w, R, Q) \in P$, $z_1 z_2$ contains all letters of R, and $z_1 z_2$ contains no letter of Q. The language $L(G)$ generated by G is defined as

$$L(G) = \{y:\ z \overset{*}{\Rightarrow} y \quad \text{for some} \quad z \in S\}$$

where $\overset{*}{\Rightarrow}$ denotes the reflexive and transitive closure of $\Rightarrow$.

A *pure programmed* grammar is a triple $G=(V, P, S)$ where P is a finite set of rules of the form

$$(b, a \to w, E(b), F(b))$$

where b is a label of the production, $a \in V$, $w \in V^*$, and $E(b)$ and $F(b)$ are subsets of the set of labels. The language $L(G)$ consists of all words y such that there is a derivation

$$z = y_0 \underset{b_1}{\Longrightarrow} y_1 \underset{b_2}{\Longrightarrow} y_2 \underset{b_3}{\Longrightarrow} \cdots \underset{b_{n-1}}{\Longrightarrow} y_{n-1} \underset{b_n}{\Longrightarrow} y_n = y$$

where $z \in S$, $(b_i, a_i \to w_i, E_i, F_i)$ are rules of P, $1 \leqq i \leqq n$, and, for $1 \leqq i \leqq n$,

$$y_{i-1} = z_{i1} a_i z_{i2},\ y_i = z_{i1} w_i z_{i2} \quad \text{for some} \quad z_{i1}, z_{i2} \in V^*,$$
$$\text{and} \quad b_{i+1} \in E_i \quad (\text{if} \quad i < n)$$

or

$$a_i \text{ does not occur in } y_{i-1},\ y_i = y_{i-1}, \text{ and } b_{i+1} \in F_i \text{ (again, if } i < n).$$

A *pure matrix* grammar is a quadruple $G=(V, M, S, F)$ where M is a finite set of finite sequences of productions,

$$M = \{m_1, m_2, \dots\ m_r\},$$

$$m_i = (a_{i1} \to w_{i1},\ a_{i2} \to w_{i2}, \dots, a_{ir(i)} \to w_{ir(i)}),$$

$a_{ij} \in V$, $w_{ij} \in V^*$, $1 \leqq i \leqq r$, $1 \leqq j \leqq r(i)$, and F is a subset of occurrences of rules in M. Then, for $1 \leqq i \leqq r$, we say that $x \underset{m_i}{\Rightarrow} y$ iff

$$x = y_0 \Rightarrow y_1 \Rightarrow y_2 \Rightarrow \dots \Rightarrow y_{r(i)} = y$$

where

$$y_{j-1} = z_{j1} a_{ij} z_{j2},\ y_j = z_{j1} w_{ij} z_{j2} \quad \text{for some} \quad z_{j1}, z_{j2} \in V^*$$

or

a_{ij} does not occur in y_{j-1}, $a_{ij} \to w_{ij} \in F$, and $y_j = y_{j-1}$.

The language $L(G)$ generated by G is defined as the set of all words y which are obtained by iterated applications of matrices (elements of M) to words of S and all intermediate words (y_j in the above notation) of these applications of matrices.

These definitions are the most general ones, i.e. rules of the form $a \to \lambda$ are allowed and the appearance checking mode is used in the derivation process.

By $\mathscr{L}(pRC_{ac}^{\lambda})$, $\mathscr{L}(pPR_{ac}^{\lambda})$, $\mathscr{L}(pM_{ac}^{\lambda})$ we denote the families of languages obtained by pure random context, pure programmed, and pure matrix grammars, respectively. We omit the upper or lower index or both indices if we consider the families of languages generated by grammars without erasing rules or without appearance checking (i.e. $Q=\emptyset$ and $F(b)=\emptyset$ for all productions, or $F=\emptyset$) or without both these features.

By $\mathscr{L}(pCF)$ and $\mathscr{L}(pCS)$ we denote the families of pure context-free and pure context-sensitive languages, respectively (the definitions can be given in an obvious way, e.g. see (4)), and we add the upper index λ if λ-rules are allowed.

$\#_a(w)$ denotes the number of occurrences of the letter a in the word w.

2. The hierarchy of pure language families

Let us consider the pure programmed grammar

$$G_1 = (\{a, b\},\ \{(1, a \to b^3, \{1\}, \{2\}), (2, b \to a^3, \{2\}, \{1\})\},\ \{a\}).$$

The language L_1 generated by G_1 contains only words of $\{a, b\}^+$ which satisfy one of the following conditions:

$$\begin{aligned} \#_a(w) + \frac{\#_b(w)}{3} &= 3^{2n}, \\ \frac{\#_a(w)}{3} + \#_b(w) &= 3^{2n+1}, \end{aligned} \tag{1}$$

where $n \in N$.

Lemma 1. $L_1 \in \mathscr{L}(pPR_{ac})$, $L_1 \notin \mathscr{L}(pPR^{\lambda})$.

Proof. By the construction of L_1 we have to prove only the second statement. Let us assume the contrary, i.e. $L_1 = L(G)$ for some pure programmed grammar G without appearance checking. First we note that, for $w,\ w' \in L_1,\ w \neq w'$, $|l(w) - l(w')| \geqq 2$ holds. Hence without loss of generality we can assume that G is λ-free. If there is a rule whose core production is of the form $a \to b$ or $b \to a$ then its application to a word of L_1 produces a word which do not belong to L_1.

For a production $p: (h, x \to w, E(h), \emptyset)$, we set

$$l(p) = l(w) - 1,$$

and we also define

$$l(G) = \max_{p \in P} l(p).$$

Obviously, $l(G) \geqq 2$. Further, let t_1 be the number of productions in G, let t_2 be the maximal length of a word in S, and let n be an integer such that $3^n > t_1 \cdot t_2 \cdot l(G)$. Now we consider a derivation D of $a^{3^{2n}} b^3 \in L_1$, especially the last (t_1+2) steps of this derivation which increases the length, i.e.

$$D:\ s \overset{*}{\Rightarrow} y_0 \overset{}{\underset{l_0}{\Rightarrow}} y_1 \overset{*}{\Rightarrow} y_1 \overset{}{\underset{l_1}{\Rightarrow}} y_2 \overset{*}{\Rightarrow} y_2 \Rightarrow \ldots \overset{}{\underset{l_{t_1}}{\Rightarrow}} y_{t_1+1} \overset{*}{\Rightarrow} y_{t_1+1} \overset{}{\underset{l_{t_1+1}}{\Rightarrow}} a^{3^{2n}} b^3$$

where the derivation steps $y_i \Rightarrow y_{i+1}$ are obtained by application of the production $(l_i, x_i \to w_i, E_i, \emptyset)$ of G with $l(w_i) \geqq 2$ and the phases $y_i \overset{*}{\Rightarrow} y_i$ contain only applications of rules with core productions $a \to a$ or $b \to b$. By the definition of n and t_1, the following facts are valid:

— $\dfrac{\#_a(y_i)}{3} + \#_b(y_i) = 3^{2n-1}$ for $1 \leqq i \leqq t_1$,

— there are two integers k, j with $0 \leqq k < j \leqq t_1$ such that $l_k = l_j$.

Let

$$\#_a(y_k) = 3^{2n} - 3r_1, \quad \#_b(y_k) = r_1,$$
$$\#_a(y_j) = 3^{2n} - 3r_2, \quad \#_b(y_j) = r_2.$$

Then $3^n > r_1 - r_2 > 0$. Further we have also the correct derivation

$$s \overset{*}{\Rightarrow} y_0 \overset{}{\underset{l_0}{\Rightarrow}} y_1 \overset{*}{\Rightarrow} \ldots \Rightarrow y_k \overset{}{\underset{l_j}{\Rightarrow}} z_j \overset{*}{\Rightarrow} z_j \overset{}{\underset{l_{j+1}}{\Rightarrow}} z_{j+1} \Rightarrow \ldots \overset{}{\underset{l_{t_1+1}}{\Rightarrow}} z_{t_1+1}$$

where $z_j, \ldots, z_{t_1+1}$ are appropriate strings with

$$\#_a(z_{t_1+1}) = 3^{2n} - 3(r_1 - r_2), \quad \#_b(z_{t_1+1}) = 3 + (r_1 - r_2).$$

This contradicts (1) and thus $z_{t_1+1} \notin L_1$. Since $z_{t_1+1} \in L(G)$ we obtain the desired contradiction to $L(G) = L_1$. □

Now we consider the pure matrix grammar

$$G_2 = (\{a, b, c, d\},\ \{(a \to a^3,\ b \to b^3,\ c \to c^3),\ (a \to a^3,\ b \to d^3,\ c \to c^3)\},$$
$$\{abc\},\ \{b \to b^3,\ b \to d^3\}).$$

Then the words of its generated language L_2 satisfy the following conditons:
If $\#_b(w) = 0$, then $w = a^{2n+1} d^{3m} c^{2n-1}$ or $w = a^{2n+1} d^{3m} c^{2n+1}$ with $2n+1 > 3m$, $n, m \in N$.

If $\#_b(w) \geqq 1$, then $w = a^{2n+1}w'c^{2n+1}$ or $w = a^{2n+1}w'c^{2n-1}$ where

$$w' \in \{b, d^3\}^+, \quad l(w') = 2n+1 \quad \text{or} \quad l(w') = 2n-1.$$

Lemma 2. $L_2 \in \mathscr{L}(pM_{ac})$, $L_2 \notin \mathscr{L}(pM^\lambda)$.

Proof. Again, we have to prove only the second statement. Let us assume that $L_2 = L(G)$ for some pure matrix grammar G without appearance checking. As in the preceding proof we can show that all core productions (besides $x \to x$) have the form $x \to w$ with $l(w) \geqq 3$. Let n be a sufficiently large number. We consider a derivation of $w = a^{2n+1}d^3c^{2n+1}$, say

$$D\colon\ s \overset{*}{\Longrightarrow} v_1 \Rightarrow v_2 \Rightarrow w$$

where (without loss of generality) $l(v_1) < l(v_2) < l(w)$. By the structure of the words in L_2 it is easy to prove that $v_1 = a^{2n-1}d^3c^{2n-1}$, $v_2 = a^{2n+1}d^3c^{2n-1}$. Iterating this argument and taking into consideration that we can omit length preserving matrices we obtain

$$D\colon\ s \overset{*}{\Longrightarrow} u_1 \underset{m}{\Longrightarrow} u_2 \overset{*}{\Longrightarrow} w$$

where the derivation $u_2 \overset{*}{\Longrightarrow} w$ corresponds to the application of a proper initial part of a matrix or $u_2 = w$, and the application of m increases the number of a's and/or c's only. If it increases only the number of a's ,then by its iterated application to u_2 we can generate a word y with $\#_a(y) - \#_c(y) > 2$ which contradicts the structure of the words in L_2. Analogously, the matrix m cannot only increase the number of c's. Hence it has to increase both numbers. Now we consider a derivation D' of $w' = a^{2n+1}b^{2n+1}c^{2n+1}$. Again,

$$D'\colon\ s' \overset{*}{\Longrightarrow} z \overset{*}{\Longrightarrow} w'$$

where $z \overset{*}{\Longrightarrow} w'$ is the initial part of a matrix application or $z = w'$ and z is generated by iterated applications of matrices. Clearly, $\#_b(z) \geqq 1$. Then it is easy to show that the correct derivation

$$s' \overset{*}{\Longrightarrow} z \underset{m}{\Longrightarrow} z_1 \underset{m}{\Longrightarrow} z_2 \underset{m}{\Longrightarrow} z_3$$

produces a word z_3 which is not in L_2.

Lemma 3. *Let*

$$L_3 = \{a^2b^2c,\ ab^5c,\ b^3ab^2c,\ b^8c,\ b^{11}\} \cup \{a^{2n}b^5\colon\ n \geqq 1\} \cup$$

$$\cup \{b^3a^{2n+1}b^2\colon\ n \geqq 1\} \cup \{a^{2n+1}b^8\colon\ n \geqq 1\}.$$

Then $L_3 \in \mathscr{L}(pRC_{ac})$, $L_3 \notin \mathscr{L}(pRC^\lambda)$.

Proof. The pure random context grammar $G_3 = (\{a, b, c\}, \{(c \to b^3, \emptyset, \emptyset), (a \to a^3, \emptyset, \{c\}), (a \to b^3, \{c\}, \emptyset)\}, \{a^2b^2c\})$ generates L_3.

Assume that $L_3 = L(G)$ for some pure random context grammar without appearance checking. We consider $a^{2n}b^5$ where n is sufficiently large. This word can be derived only from a word $a^{2m}b^5$ (without loss of generality we can assume that $m < n$), and thus we have a production $(a \to a^{2(n-m)+1}, R, \emptyset)$ or $(b \to a^{2(n-m)}b, R, \emptyset)$

with $R \subseteq \{a, b, c\}$. This rule is applicable to a^2b^2c producing a word not contained in L_3.

Lemma 4. *Let*

$$L_4 = \{ca^n b^n : n \geq 2\} \cup \{ca^{n+1} b^n : n \geq 2\} \cup \{a^n b^n : n \geq 2\}.$$

Then $L_4 \in \mathcal{L}(pM^\lambda)$, $L_4 \in \mathcal{L}(pM_{ac})$.

Proof. Clearly, L_4 is generated by the pure matrix grammar

$$G_4 = (\{a, b, c\}, \{(c \to c, a \to a^2, b \to b^2), (c \to \lambda)\}, \{ca^2b^2\}).$$

Thus $L_4 \in \mathcal{L}(pM^\lambda)$.

Now assume $L_4 = L(G)$ for some pure matrix grammar G with appearance checking but without λ-rules. We consider the word $a^n b^n$ where n is chosen such that $a^n b^n$ is not an axiom. Then there is a word z with $z \Rightarrow a^n b^n$ and $z \neq a^n b^n$. If $z = a^r b^r$ for some $r < n$ ($r > n$ is impossible by the λ-freeness and $r = n$ is already excluded), then we have applied a rule of the form $a \to a^{s+1} b^s$ to the last a in z or $b \to a^s b^{s+1}$ to the first b in z where $s \geq 1$. Since $r \geq 2$ we can apply this rule to the first a or last b, too, and then we derive a word which is not in L_4. Hence z is of the form $ca^r b^r$ or $ca^{r+1} b^r$, and we have to apply a rule of the form $c \to z'$ which yields $z'a^r b^r$ or $z'a^{r+1} b^r$. In the first case $z' = \lambda$ and $r = n$ have to hold and this contradicts the λ-freeness of G. In the second case we do not obtain $a^n b^n$. Therefore $L_4 = L(G)$ do not hold for all λ-free pure matrix grammars G.

Lemma 5. *Let* $L_4' = \{d\} \cup L_4$. *Then* $L_4' \in \mathcal{L}(pPR^\lambda)$, $L_4' \in \mathcal{L}(pPR_{ac})$.

Proof. It is easy to see that $L_4' \in \mathcal{L}(pPR^\lambda)$, and $L_4' \notin \mathcal{L}(pPR_{ac})$ can be proved analogously to the proof of Lemma 4.

Lemma 6. *Let*

$$L_5 = \{ca^n b^n d : n \geq 2\} \cup \{c'a^{n+1} b^n d : n \geq 2\} \cup \{c'a^n b^n d' : n \geq 3\} \cup$$
$$\cup \{ca^n b^n d' : n \geq 3\} \cup \{a^n b^n d : n \geq 2\}.$$

Then $L_5 \in \mathcal{L}(pRC^\lambda)$, $L_5 \notin \mathcal{L}(pRC_{ac})$.

Proof. The pure random context grammar

$$G_5 = (\{a, b, c, d, c', d'\}, \{(c \to c'a, \{d\}, \emptyset), (d \to bd', \{c'\}, \emptyset)$$
$$(c' \to c, \{d'\}, \emptyset), (d' \to d, \{c\}, \emptyset), (c \to \lambda, \{d\}, \emptyset)\}, \{ca^2b^2d\})$$

generates L_5. Hence $L_5 \in \mathcal{L}(pRC^\lambda)$.

Now let $L_5 = L(G)$ for some pure random context grammar G (with appearance checking) without erasing rules. We consider $w = a^n b^n d$ with sufficiently large n. As in the proof of Lemma 4 it can be shown that w cannot be generated from a word of the form $a^r b^r d$. Hence z with $z \Rightarrow w$, $z \neq w$ has to be of the form $ca^r b^r d$ or $c'a^{r+1} b^r d$. It is easy to see that $a^n b^n d$ is obtained iff $c \to \lambda$ is applied to z and $r = n$ holds, i.e. we get a contradiction to the λ-freeness.

Lemma 7. *Let*

$$L_6 = \{b^2a\} \cup \{ba^{5+3n}\colon n \geqq 0\} \cup \{a^{3m+4}ba^{3n+1}\colon m, n \geqq 0\}.$$

Then $L_6 \in \mathscr{L}(pRC)$, $L_6 \notin \mathscr{L}(pPR_{ac}^{\lambda})$, $L_6 \notin \mathscr{L}(pM_{ac}^{\lambda})$.

Proof. i) L_6 is generated by the pure random context grammar

$$G_6 = (\{a, b\}, \{(b \to a^4, \{b\}, \emptyset), (a \to a^4, \{a\}, \emptyset)\}, \{b^2a\}).$$

Thus $L_6 \in \mathscr{L}(pRC)$.

ii) Assume that $L_6 = L(G)$ for some pure programmed grammar G. Again, G is λ-free, and hence b^2a has to be an axiom. Further, in order to generate ba^{5+3n} for a sufficiently large integer n we need a production with the core rule $a \to a^{3m+1}$ for some $m > 0$. Clearly, it can be applied to the axiom, and this gives $b^2a \Rightarrow b^2a^{3m+1}$. Therefore $L(G)$ contains the word b^2a^{3m+1} which is not in L_6.

iii) can be proved analogously to ii).

Fact 1. *Let*

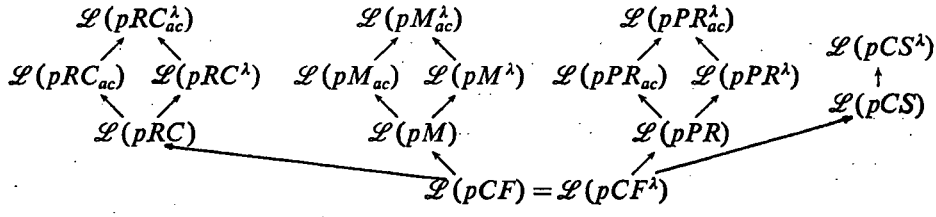
$$L_7 = \{a^nb^nc^n\colon n \geqq 1\} \cup \{a^{n+1}b^nc^n\colon n \geqq 1\} \cup \{a^{n+1}b^{n+1}c\colon n \geqq 1\}.$$

Then $L_7 \in \mathscr{L}(pM)$, $L_7 \notin \mathscr{L}(pPR_{ac}^{\lambda})$, $L_7 \notin \mathscr{L}(pRC_{ac}^{\lambda})$.

Fact 2. *If* $L_8 = \{a, a^5\} \cup \{a^{7+10n}\colon n \geqq 0\} \cup \{a^{11+10n}\colon n \geqq 0\}$, then $L_8 \in \mathscr{L}(pPR)$, $L_8 \notin \mathscr{L}(pRC_{ac}^{\lambda})$, $L_8 \notin \mathscr{L}(pM_{ac}^{\lambda})$.

Summarizing all these results and taking into consideration the relations to pure versions of the grammars of the Chomsky hierarchy which are already given in (1) we obtain the following diagram. Instead of $A \subsetneq B$ we write $A \to B$, and if two families are not connected then they are incomparable.

Theorem 8.



3. Codings of pure languages

Let X be a family of grammars. Then we set $\mathscr{C}(X) = \{L\colon L = h(L')$ for some $L' \in \mathscr{L}(X)$ and some coding $h\}$ (i.e. $h(a)$ has the length 1 for all letters a).

Lemma 9. $\mathscr{C}(pM) = \mathscr{C}(pPR)$.

Proof. $\mathscr{C}(pM) \subseteq \mathscr{C}(pPR)$ can be proved analogously to (3). Now let $G = (V, P, S)$ be a pure programmed grammar. We set $V' = \{(a, b)\colon a \in V, b$ is a label of a production in $P\}$. With the production $(b, a \to w, E(b), \emptyset)$ we associate the

matrices

$$((a, b) \to w,\ x \to (x, b'))$$

where $x \in V$, $b' \in E(b)$. The set S' is defined as the set of all words $w_1(x, b)w_2$ where $w_1 x w_2 \in S$, $(x, b) \in V'$. We consider the pure matrix grammar $G' = (V \cup V', M, S', \emptyset)$ where M is the set of all matrices of the above introduced type. Then the correct derivations have the following form (in the second row we note the applied rule):

$$s = w_{11}(x_1, b_1)w_{21} \overset{}{\underset{(x_1,b_1)\to z_1}{\Longrightarrow}} w_{11}z_1w_{21} \overset{}{\underset{x_2\to(x_2,b_2)}{\Longrightarrow}} w_{12}(x_2, b_2)w_{22} \ldots \Rightarrow$$

$$\ldots \Rightarrow w_{1n}(x_n, b_n)w_{2n} \overset{}{\underset{(x_n,b_n)\to z_n}{\Longrightarrow}} w_{1n}z_nw_{2n}$$

$$\overset{}{\underset{x_{n+1}\to(x_{n+1},b_{n+1})}{\Longrightarrow}} w_{1,n+1}(x_{n+1}, b_{n+1})w_{2,n+1} \Rightarrow \ldots .$$

Further we consider the coding h given by $h(x) = x$ for $x \in V$ and $h((x, b)) = x$ for $(x, b) \in V'$. Then the image of the above derivation is

$$w_{11}x_1w_{21} \overset{}{\underset{b_1}{\Longrightarrow}} w_{11}z_1w_{21} = w_{12}x_2w_{22} \overset{}{\underset{b_2}{\Longrightarrow}} \ldots$$

$$\ldots \Rightarrow w_{n1}x_nw_{2n} \overset{}{\underset{b_n}{\Longrightarrow}} w_{1n}z_nw_{2n} = w_{1,n+1}x_{n+1}w_{2,n+1} \overset{}{\underset{b_{n+1}}{\Longrightarrow}} \cdots$$

where, by the definition of the matrices, b_{i+1} is in the success field $E(b_i)$ of b_i. Thus we have proved that

$$L(G) = h(L(G')).$$

Since the composition of codings is a coding again,

$$h'(L(G)) = (h \circ h')(L(G')),$$

and thus

$$\mathscr{C}(pPR) \subseteq \mathscr{C}(pM).$$

Combining Lemma 9 with the results of (3) we obtain

Theorem 10.

$$\mathscr{C}(pRC) \subseteq \mathscr{C}(pPR) = \mathscr{C}(\mathrm{pm}) \subseteq \mathscr{C}(pM_{ac}) \subsetneq \mathscr{C}(pPR_{ac}) = \mathscr{C}(pRC_{ac}) \subsetneq \mathscr{C}(pCS).$$

It is known (see (8), (5)) that for usual languages (i.e. sets of words over a terminal alphabet) the following hierarchy holds:

$$\mathscr{L}(RC) \subseteq \mathscr{L}(M) = \mathscr{L}(PR) \subseteq \mathscr{L}(M_{ac}) = \mathscr{L}(PR_{ac}) = \mathscr{L}(RC_{ac}) \subsetneq \mathscr{L}(CS)$$

(and that for $X \in \{M, PR, RC\}$, $\alpha = ac$ or empty, all the families $\mathscr{L}(X_\alpha^\lambda)$ coincide with the family of all recursively enumerable languages). Thus one sees that the hierarchy of families obtained by codings of pure languages is situated between the hierarchy of language families obtained by the use of nonterminals and terminals and that of pure language families.

Acknowledgement. The author is grateful to the referee and Henning Bordihn for their comments which improved the earlier version.

TECHNISCHE UNIVERSITÄT
OTTO VON GUERICKE
POSTFACH 124
MAGDEBURG 3010/DDR

References

[1] DASSOW, J., Pure grammars with regulated rewriting. Revue Roumaine Math. Pures Appl. 31 (1986) 657—666.
[2] DASSOW, J. & GH. PAUN, Further remarks on pure grammars with regulated rewriting. Revue Roumaine Math. Pures Appl. 31 (1986) 855—864.
[3] DASSOW, J. & GH. PAUN, Codings of pure languages obtained by regulated rewriting. Foundat. Control Engineer. 9 (1984) 3—14.
[4] MAURER, H., A. SALOMAA & D. WOOD, Pure grammars. Inform. Control 49 (1980) 47—72.
[5] MAYER, O., Some restrictive devices for context free grammars. Inform. Control 20 (1972) 69—92.
[6] PAUN, GH. A note on pure matrix grammars. To appear in Revue Roumaine Math. Pures Appl.
[7] ROZENBERG, G. & A. SALOMAA, The Mathematical Theory of *L* Systems. Academic Press, 1980.
[8] SALOMAA, A. Formal Languages. Academic Press, 1973.

(Received Oct. 15, 1985)

On state grammars

A. MEDUNA and GY. HORVÁTH

In this paper we study some properties of state grammars. Among others, it is shown that for every recursively enumerable language there exists a state grammar with erasing rules that generates it. Some problems concerning the descriptive complexity of state grammars are discussed.

1. Introduction

In the last years several types of grammars have been defined which have context-free rules and an additional mechanism which regulates the derivation process (see e.g. [1]—[6]), since such grammars can describe such special aspects of programming languages that cannot be covered by context-free grammars. One of the typical grammars of this kind is the state grammar (see [3]) — the subject under investigation in the present paper.

Intuitively, a state grammar is a context-free grammar with an additional mechanism which consists of the following: at each derivation step the grammar is in a state which influences the choice of the next production to be applied, and the next state is determined by this production. Moreover, rewriting is done in a leftmost fashion.

2. Preliminaries

In this section we briefly review some of the basic notions and notations of formal language theory. Items not defined explicitly are standard in language theory, see e.g. [5].

For a finite set A we use $|A|$ to denote its cardinality. If w is a word over an alphabet Z, then $|w|$ denotes the length of w, alph (w) denotes the set of letters occuring in w and $N_Y(w)$ denotes the number of occurrences of letters from $Y(\subseteq Z)$ in w. The empty word is denoted by λ. For a grammar G, $\underset{G}{\Rightarrow}$ denotes its direct derivation relation and $\underset{G}{\Rightarrow}^*$ denotes the derivation relation of G. We also write $\Rightarrow$ and $\Rightarrow^*$ rather than $\underset{G}{\Rightarrow}$ and $\underset{G}{\Rightarrow}^*$ if no confusion arises. A production (p, q) of a grammar G will also be called a rule and written as $p \to q$. For context-free grammars we use

the following notation: $G=(\Sigma, \Delta, x_0, P)$, where Σ is the total alphabet of G, $\Delta \subseteq \Sigma$ is the terminal alphabet of G, $x_0 \in \Sigma \setminus \Delta$ is the initial letter and P is the set of productions of G.

Now we recall the definition of a queue grammar defined in [4].

A queue grammar is a 7-tuple $Q=(\Sigma, \Delta, S, F, x_0, s_0, P)$ where Σ and S are finite sets, the total alphabet and the state set of Q, respectively, $\Delta \subseteq \Sigma$ is the terminal alphabet, $F \subseteq S$ is the final state set, $x_0 \in \Sigma \setminus \Delta$ is the initial letter, $s_0 \in S$ is the initial state and finally, $P \subseteq \Sigma \times (S \setminus F) \times \Sigma^* \times S$ is a finite set, the production set of Q. A production (x, s, u, s') in P is also written $(x, s) \to (u, s')$.

For any $(u, s), (v, s') \in \Sigma^* \times S$ the direct derivation $(u, s) \underset{Q}{\Rightarrow} (v, s')$ holds iff $u=xu_1$ for some $x \in \Sigma$ and $u_1 \in \Sigma^*$, and there is a production $(x, s) \to (v_1, s')$ in P such that $v=u_1 v_1$. The derivation relation $\underset{G}{\Rightarrow}^*$ is the reflexive transitive closure of $\underset{Q}{\Rightarrow}$.

The language $L(Q)$ generated by Q is defined by $L(Q)=\{w \in \Delta^*: (x_0, s_0) \Rightarrow^* (w, s)$ for some $s \in F\}$.

The central notion of this paper is that of a state grammar that we recall now.

A state grammar is a construct $G=(\Sigma, \Delta, S, x_0, s_0, P)$ where Σ and S are finite sets, the total alphabet and the state set, respectively, $\Delta \subseteq \Sigma$ is the terminal alphabet, $x_0 \in \Sigma \setminus \Delta$ is the initial letter, $s_0 \in S$ is the initial state and P is a finite set of productions of the form $(x, s) \to (u, s')$ where $x \in \Sigma \setminus \Delta$, $s, s' \in S$ and $u \in \Sigma^*$. The direct derivation $(u,s) \underset{G}{\Rightarrow} (v, s')$ holds for $u, v \in \Sigma^*$ and $s, s' \in S$ iff there exist decompositon $u=u_1 x u_2$, $v=u_1 v_1 u_2$ and a production $(x, s) \to (v_1, s')$ in P such that for every nonterminal $y \in \Sigma \setminus \Delta$ occuring in the word u_1 there is no production with left side (y, s) in P.

The language $L(G)$ generated by G is defined by $L(G)=\{w \in \Delta^*: (x_0, s_0) \underset{G}{\Rightarrow}^* (w, s)$ for some $s \in S\}$, where $\underset{G}{\Rightarrow}^*$ is the reflexive transitive closure of $\underset{G}{\Rightarrow}$.

A state grammar G is called propagating if for every production $(x, s) \to (u, s')$ of G we have $u \neq \lambda$.

We point out that the above definition differs from Kasai's original definition in [3] since Kasai considers only propagating state grammars.

The family of languages generated by context-free, contex-sensitive, type 0, queue and state grammars is denoted by $\mathscr{L}(\mathrm{CF})$, $\mathscr{L}(\mathrm{CS})$, $\mathscr{L}(\mathrm{RE})$, $\mathscr{L}(\mathrm{Q})$ and $\mathscr{L}(\mathrm{S})$, respectively.

3. On the generative power of state grammars with erasing productions

In his original definition of state grammars, Kasai did not consider erasing productions. It is however a natural generalization (see e.g. [6]). But then, the first question we have to ask is: what is the generative power of such grammars? We are going to show in this section that $\mathscr{L}(\mathrm{S})$ and $\mathscr{L}(\mathrm{RE})$ coincide. Although the equality $\mathscr{L}(\mathrm{S})=\mathscr{L}(\mathrm{RE})$ can be obtained as a direct consequence of a theorem in section 4, we prove it here in order to demonstrate the close connection between queue and state grammars, moreover, this connection will be used in a subsequent section.

A state grammar $G=(\Sigma, \Delta, S, \#, s_0, P)$ is called front-end state grammar if the nonterminal letter $\#$ is an endmarker, which means that every production containing $\#$ is of the two forms $(\#, s) \to (u\#, s')$ or $(\#, s) \to (u, s')$, where $\#$ does

not occur in the word u, moreover, for every state s, if there is a production with left side (x, s) for some $x \neq \#$, then for each nonterminal letter $x' \neq \#$ there is a production with left side (x', s).

One can see that in any derivation step according to a front-end state grammar, the rewritten non-terminal letter is either the first nonterminal in the word or the endmarker.

Lemma 1. For any queue grammar Q one may construct an equivalent front-end state grammar G.

Proof. Let $Q=(\Sigma, \Delta, S, F, x_0, s_0, P)$ be an arbitrary queue grammar. We construct a state grammar $G=(\Sigma', \Delta, S, \#, q_0, P)$. For each terminal letter $a \in \Delta$ we introduce a new nonterminal letter x_a. We set

$$\Sigma' = \Sigma \cup \{x_a : a \in \Delta\} \cup \{\#\}$$

$$S' = \{q_0, q_1\} \cup S \cup (\Sigma \setminus \Delta \cup \{x_a : a \in \Delta\}) \times S.$$

We define a homomorphism h of Σ into Σ' by $h(x)=x$ for $x \in \Sigma \setminus \Delta$ and $h(a)=x_a$ for $a \in \Delta$. The homomorphism g of $\Sigma \cup h(\Delta)$ into Σ is defined by $g(x)=x$ for $x \in \Sigma$ and $g(x_a)=a$ for $a \in \Delta$. Clearly, $g(h(x))=x$ for $x \in \Sigma$ and $h(g(x))=x$ for $x \in \Sigma \cup h(\Delta)$. Now the production set $P'=P'_0 \cup P'_1 \cup P'_2 \cup P'_3 \cup P'_4$ is constructed in the following way:

$$P'_0 = \{(\#, q_0) \to (x_0 \#, s_0)\},$$

$$P'_1 = \{(x, s) \to (\lambda, [x, s]) : s \in S \setminus F, x \in h(\Sigma)\},$$

$$P'_2 = \{(\#, [x, s]) \to (h(u) \#, s') : (g(x), s) \to (u, s') \in P\},$$

$$P'_3 = \{(\#, s) \to (\lambda, q_1) : s \in F\},$$

$$P'_4 = \{(x, q_1) \to (g(x), q_1) : x \in h(\Sigma)\}.$$

It is evident that the state grammar G constructed above satisfies the front-end requirement. We are going to prove that a derivation

$$(x_0, s_0) \underset{Q}{\Rightarrow}^* (w, s) \quad \text{holds iff}$$

$$(x_0 \#, s_0) \underset{G}{\Rightarrow}^* (h(w) \#, s) \quad \text{holds.}$$

Indeed, for any production $(x, s) \to (u, s')$ in Q, the productions $(h(x), s) \to (\lambda, [h(x), s])$ and $(\#, [h(x), s]) \to (h(u) \#, s')$ are present in G, moreover, for every word $v \in h(\Sigma)^*$ the derivation

$$(h(x)v \#, s) \underset{G}{\Rightarrow} (v \#, [h(x), s]) \underset{G}{\Rightarrow} (vh(u) \#, s')$$

holds. Conversely, a derivation $(x_0 \#, s_0) \underset{G}{\Rightarrow}^* (w \#, s)$ can only be carried out by using productions from P'_1 and from P'_2. If a production $(h(x), s) \to (\lambda, [h(x), s])$, $x \in \Sigma$, is used in a derivation step, then after it a suitable production $(\#, [h(x), s]) \to (h(u) \#, s')$ must be applied, where $(x, s) \to (u, s')$ is in P. By the front-end property of G we obtain that $(x_0, s_0) \underset{Q}{\Rightarrow}^* (w, s)$ holds. Assume that $w \in L(Q)$. Then

$(x_0, s_0) \overset{}{\underset{Q}{\Rightarrow}}^*(w, s)$ holds for some $s \in F$, consequently we obtain the derivation

$$(\#, q_0) \overset{}{\underset{G}{\Rightarrow}} (x_0\#, s_0) \overset{}{\underset{G}{\Rightarrow}}^* (h(w)\#, s) \overset{}{\underset{G}{\Rightarrow}} (h(w), q_1).$$

The iterated application of productions from P_4' gives the derivation

$$(\#, q_0) \overset{}{\underset{G}{\Rightarrow}} (h(w), q_1) \overset{}{\underset{G}{\Rightarrow}}^* (g(h(w)), q_1) = (w, q_1).$$

Therefore $w \in L(G)$.

To establish the converse inclusion $L(G) \subseteq L(Q)$, assume that $w \in L(G)$. Thus $(\#, q_0) \overset{}{\underset{G}{\Rightarrow}}^* (w, q_1)$ must hold since every production of G not containing nonterminal letter on its right side is a member of P_3' or P_4'. The nonterminal letter $\#$ can only be eliminated by a production from P_3', thus we obtain the derivation

$$(\#, q_0) \overset{}{\underset{G}{\Rightarrow}} (x_0\#, s_0) \overset{}{\underset{G}{\Rightarrow}}^* (v\#, s) \overset{}{\underset{G}{\Rightarrow}} (v, q_1) \overset{}{\underset{G}{\Rightarrow}}^* (w, q_1)$$

for some $s \in F$ and $v \in h(\Sigma)^*$. Furthermore, $w = g(v)$ by virtue of definition of P_4', thus $h(w) = h(g(v)) = v$. We obtain the derivation $(x_0\#, s_0) \overset{}{\underset{G}{\Rightarrow}}^* (h(w), s)$, which implies $(x_0, s_0) \overset{}{\underset{Q}{\Rightarrow}}^* (w, s)$. Therefore $w \in L(Q)$. □

Lemma 2. Every recursively enumerable language can be generated by a front-end state grammar.

Proof. From Theorem 2.1 Chapter 2 in [4] we know that $\mathscr{L}(Q) = \mathscr{L}(RE)$, hence the lemma holds by Lemma 1. □

The following theorem is now an immediate consequence of Lemma 2.

Theorem 1. $\mathscr{L}(S) = \mathscr{L}(RE)$.

4. On the complexity of state grammars

It is a natural question whether or not the representation of languages by grammars of a certain type is "better" than by grammars of another type. In this section we study complexity measures of state grammars in terms of [2] as for example the number of nonterminals and the number of states sufficient to generate any language of a given type.

First we investigate the complexity of state grammars with regard to the measure of states.

It is an immediate consequence of the definition of the state grammar that any language L is context-free iff L can be generated by a state grammar with a single state.

Theorem 2. Let $L \in \mathscr{L}(RE)$. Then there exists a state grammar $G = (\Sigma, \Delta, S, x_0, s_0, P)$ such that $L = L(G)$ and $|S| = 3$.

Proof. Every type-0 language L can be obtained in the form $L = h(L_1 \cap L_2)$ where h is a homomorphism and L_1 and L_2 are context-free languages (Theorem 11.1, Part one in [6]). Assume that the languages L_1 and L_2 are generated by the context-free grammars $G_1 = (\Sigma_1, \Delta', x_0^1, P_1)$ and $G_2 = (\Sigma_2, \Delta', x_0^2, P_2)$, such that

$\Sigma_1 \cap \Sigma_2 = \emptyset$, $\Delta \cap \Delta' = \emptyset$. Let $h: \Delta' \to \Delta^*$ be the required homomorphism where L is a language over the alphabet Δ. Assume that $\Delta' = \{a_1, \ldots, a_n\}$ for some $n \geqq 1$ and consider an auxiliarly alphabet $\Delta'' = \{b_1, \ldots, b_n\}$. We define a function $\varphi: \Sigma_2 \to \to \Sigma_2 \setminus \Delta' \cup \Delta''$ by $\varphi(x) = x$ for $x \in \Sigma_2 \setminus \Delta$ and $\varphi(a_i) = b_i$ for $a_i \in \Delta'$. Furthermore, let $\psi: \Delta'' \to \Delta'$ be defined by $\psi(b_i) = a_i$ for $b_i \in \Delta''$.
Now we construct the state grammar G in the following way:

$$\Sigma = \{x_0, \#, \bot\} \cup \Sigma_1 \cup \Sigma_2 \cup \Delta'' \cup \{a_i^j, b_i^j \colon 1 \leqq j \leqq i \leqq n\} \cup \Delta,$$

$$S = \{s_0, s_1, s_2\},$$

$$P = P^0 \cup P^1 \cup P^2 \cup P^3 \cup P^4.$$

$$P^0 = \{(x_0, x_0) \to (x_0^1 x_0^2 \#, s_0), (\#, s_0) \to (\lambda, s_1)\},$$

$$P^1 = \{(x, s_0) \to (p, s_0) \colon x \to p \in P_1\} \cup \{(y, s_0) \to (\varphi(q), s_0) \colon y \to q \in P_2\},$$

$$P^2 = \{(a_i, s_1) \to (a_i^1, s_2), (a_i^j, s_1) \to (a_i^{j+1}, s_2) \colon 1 \leqq j < i \leqq n\},$$

$$P^3 = \{(b_i, s_2) \to (b_i^1, s_1), (b_i^j, s_2) \to (b_i^{j+1}, s_1) \colon 1 \leqq j < i \leqq n\},$$

$$P^4 = \{(a_i^i, s_1) \to (h(a_i), s_0), (b_i^i, s_0) \to (\lambda, s_1), (b_i^i, s_2) \to (\bot, s_2) \colon 1 \leqq i \leqq n\}.$$

We prove first that $L \subseteq L(G)$. Let $w = h(v)$ for some $v \in L_1 \cap L_2$. Then there exist (leftmost) derivations $x_0^1 \underset{G_1}{\Rightarrow}{}^* v$ and $x_0^2 \underset{G_2}{\Rightarrow}{}^* v$. By the construction of the production set P^1 we obtain the derivation

$$(x_0, s_0) \underset{G}{\Rightarrow} (x_0^1 x_0^2 \#, s_0) \underset{G}{\Rightarrow}{}^* (v x_0^2 \#, s_0) \underset{G}{\Rightarrow}{}^* (v\varphi(v) \#, s_0) \underset{G}{\Rightarrow} (v\varphi(v), s_1).$$

Using productions from P^2, P^3 and P^4 we have

$$(x_0, s_0) \underset{G}{\Rightarrow}{}^* (v\varphi(v), s_1) \underset{G}{\Rightarrow}{}^* (h(v), s_1).$$

Therefore $w \in L(G)$.

To establish the converse inclusion $L(G) \subseteq L$ consider a derivation $(x_0, s_0) \underset{G}{\Rightarrow}{}^* \underset{G}{\Rightarrow}{}^* (w, s)$ for some $w \in \Delta^*$ and $s \in S$. This derivation must start with the use of the production $(\#, s_0) \to (x_0 x_0^2 \#, s_0)$, and the nonterminal letter $\#$ can be eliminated by the production $(\#, s_0) \to (\lambda, s_1)$ only if

$$(x_0^1 x_0^2 \#, s_0) \underset{G}{\Rightarrow}{}^* (v_1 \varphi(v_2) \#, s_0) \underset{G}{\Rightarrow} (v_1 \varphi(v_2), s_1) \underset{G}{\Rightarrow}{}^* (w, s)$$

holds for some words $v_1, v_2 \in \Delta'^*$ such that $x_0^1 \underset{G_1}{\Rightarrow}{}^* v_1$ and $x_0^2 \underset{G_2}{\Rightarrow}{}^* v_2$. Suppose that $v_1 = a_{i_1} \ldots a_{i_k}$, $a_{i_1}, \ldots, a_{i_k} \in \Delta'$, and $\varphi(v_2) = b_{j_1} \ldots b_{j_l}$, $b_{j_1}, \ldots, b_{j_l} \in \Delta''$. Considering the production sets P^2 and P^3 one can see that the nonterminal a_{i_1} can be derived to a terminal word in the derivation $(v_1 \varphi(v_2), s_1) \underset{G}{\Rightarrow}{}^* (w, s)$ only if the subderivation

$$(a_{i_1} \ldots a_{i_k} b_{j_1} \ldots b_{j_l}, s_1) \underset{G}{\Rightarrow}{}^* (a_{i_1}^{i_1} \ldots a_{i_k} b_{j_1}^{i_1} \ldots b_{j_l}, s_1) \underset{G}{\Rightarrow}$$

$$\underset{G}{\Rightarrow} (h(a_{i_1}) a_{i_2} \ldots a_{i_k} b_{j_1} \ldots b_{j_l}, s_0)$$

holds. Moreover, this subderivation can be continued only if $i_1 = j_1$ and then the production $(b_{i_1}^{i_1}, s_0) \to (\lambda, s_1)$ must be used. Repeated application of the above proc-

ess yields the equalities $k=l$, $i_t=j_t$ $(t=1, \ldots, k)$, i.e. $\varphi(v_1)=\varphi(v_2)$ and $w=h(v_1)$. Since $v_1=\psi(\varphi(v_1))=\psi(\varphi(v_2)=v_2$ we conclude that $w\in h(L_1\cap L_2)=L$. □

Now we will study the complexity of state grammars with regard to the measure of nonterminals. First we shall give a uniform definition of the (uncontrolled) finite restriction for grammars.

We say that a grammar G is of index k (for some positive integer k) if, for every word in the language $L(G)$ there exists a derivation such that no word in this derivation contains more than k occurrences of nonterminal symbols. We say that G is of uncontrolled index k if every word in each derivation of a word in $L(G)$ contains no more than k occurrences of nonterminal symbols. Finally, we say that G is of (uncontrolled) finite index if it is of (uncontrolled) index k for some positive integer k.

Lemma 3. Let $G=(\Sigma, \Delta, S, x_0, s_0, P)$ be a state grammar such that $\Sigma\setminus\Delta=\{x_0\}$. Then there exists an equivalent context-free grammar G', moreover, if G is of uncontrolled index k then so is G'.

Proof. We construct the context-free grammar $G'=(\Sigma', \Delta, y_0, P')$

$$\Sigma' = \{y_0\}\cup S\times S\cup\Delta,$$

$$P' = \{y_0 \to [s_0, s]\colon s\in S\}\cup$$

$$\{[s, s_{k+1}] \to u_1[s_1, s_2]u_2[s_2, s_3]\ldots u_k[s_k, s_{k+1}]u_{k+1}\colon k\geqq 1$$

$$s, s_1, \ldots, s_{k+1}\in S,\ u_1, \ldots, u_{k+1}\in\Delta^*,$$

$$(x_0, s) \to (u_1x_0u_2\ldots u_kx_0u_{k+1}, s_1)\in P\}\cup$$

$$\{[s, s'] \to u\colon (x_0, s) \to (u, s')\in P,\ u\in\Delta^*\}.$$

We prove by induction on the length of derivation that for any $s, s'\in S$ and $w\in\Delta^*$, $(x_0, s)\underset{G}{\Rightarrow}{}^*(w, s')$ holds iff $[s, s']\underset{G'}{\Rightarrow}{}^*w$ holds. If $(x_0, s)\underset{G}{\Rightarrow}(w, s')$ is a direct derivation then $[s, s']\underset{G'}{\Rightarrow}w$ holds by the definition of P'. Assume that for every derivation $(x_0, s)\underset{G}{\Rightarrow}{}^*(w, s')$ of length less than a given integer l $(\geqq 2)$ the derivation $[s, s']\underset{G'}{\Rightarrow}{}^*w$ holds. Let

$$(x_0, s)\underset{G}{\Rightarrow}(u_1x_0u_2\ldots u_kx_0u_{k+1}, s_1)\underset{G}{\Rightarrow}{}^*(u_1v_1u_2\ldots u_kv_ku_{k+1}, s') = (w, s')$$

be a derivation of length l, where $k\geqq 1$ and $v_i\in\Delta^*$ $(i=1, \ldots, k+1)$. Since x_0 is the only nonterminal letter in G there are derivations

$$(x_0, s_1)\underset{G}{\Rightarrow}{}^*(v_1, s_2), \ldots, (x_0, s_k)\underset{G}{\Rightarrow}{}^*(v_k, s_{k+1})$$

for some states $s_2, \ldots, s_k\in S$ and $s_{k+1}=s'$ such that the length of each derivation is less than l. By the induction hypothesis we obtain derivations $[s_1, s_2]\underset{G'}{\Rightarrow}{}^*v_1, \ldots, \ldots, [s_k, s']\underset{G'}{\Rightarrow}{}^*v_k$. Furthermore, the rule $[s, s']\to u_1[s_1, s_2]u_2\ldots u_k[s_k, s']u_{k+1}$ is in P', thus $[s, s']\underset{G'}{\Rightarrow}{}^*w$ holds. The reverse implication can be proved similarly. The second statement of the Lemma follows immediately. □

Theorem 3. Any language L can be generated by a context-free grammar of uncontrolled index k iff there exists a state grammar G of uncontrolled index k such that the number of nonterminals of G is one and $L=L(G)$.

Proof. Let

$$G' = (\Sigma, \Delta, x_0, P')$$

be a context-free grammar of uncontrolled index k for some $k \geqq 1$. Obviously, for any $w \in L(G')$ there is a leftmost derivation such that no word in this derivation contains more than k occurrences of nonterminals. Consider a state grammar

$$G = (\Delta \cup \{y_0\}, \Delta, S, y_0, s_0, P)$$

where y_0 is a new nonterminal,

$$S = \{[\alpha]: \alpha \in (\Sigma \setminus \Delta)^*, 0 \leqq |\alpha| \leqq k\},$$

$$s_0 = [x_0],$$

and the set of productions P is defined as follows:

i) if $A \to u_0 B_1 u_1 \ldots u_{n-1} B_n u_n \in P'$

where $A, B_1, \ldots, B_n \in \Sigma \setminus \Delta$, $u_0, u_1, \ldots, u_n \in \Delta^*$ for some $n \geqq 1$

then $(y_0, [A\alpha]) \to (u_0 y_0 u_1 \ldots u_{n-1} y_0 u_n, [B_1 \ldots B_n \alpha]) \in P$ for every $[A\alpha] \in S$;

ii) if $A \to u \in P'$ where $u \in \Delta^*$

then $(y_0, [A\alpha]) \to (u, [\alpha]) \in P$ for every $[A\alpha] \in S$;

iii) each element of P is obtained by i) or ii).

It follows immediately that $L(G')=L(G)$ and that G is of uncontrolled index k. The reverse implication is true by Lemma 3. □

For the definition of metalinear languages we refer to [5].

Corollary 1. The family of metalinear languages is properly contained in the family of languages generated by state grammars of uncontrolled finite index with one nonterminal.

Proof. The statement follows from Theorem 3 and Theorem 3.14 in [6]. □

Now we recall the definition of forbidding context grammars.

A forbidding context grammar is a construct $G=(\Sigma, \Delta, x_0, P)$ which is very much like a context-free grammar except that each production p of P is of the form $A \to \alpha, F$ where F is a subset of the nonterminal alphabet $\Sigma \setminus \Delta$, called the forbidding field of p. Such a production p can be used to derive a word w in context-free fashion only if $F \cap \text{alph}(w) = \emptyset$. For detailed information we refer e.g. to Chapter 3 in [4].

Theorem 4. For any forbidding context grammar of uncontrolled index k one may construct an equivalent state grammar of uncontrolled index k with two nonterminals.

Proof. Let

$$G' = (\Sigma', \Delta, x_0, P)$$

be an arbitrary forbidding context grammar of uncontrolled index k for some $k \geqq 1$. We construct a state grammar

$$G = (\Sigma, \Delta, S, y_0, [\succ x_0], P)$$

in the following way:

$$\Sigma = \{y_0, y_0'\} \cup \Delta,$$

$$S = \{[\alpha \succ \beta], [\alpha \prec \beta] : \alpha, \beta \in (\Sigma' \setminus \Delta)^*, \quad 0 \leqq |\alpha\beta| \leqq k, \succ \text{ and } \prec \text{ are new symbols}\},$$

$$P = P_1 \cup P_2 \cup P_3 \cup P_4, \text{ where}$$

$$P_1 = \{(y_0, [\alpha_1 \succ A\alpha_2]) \to (y_0', [\alpha_1 A \succ \alpha_2]),$$
$$(y_0', [\alpha_1 A \prec \alpha_2]) \to (y_0, [\alpha_1 \prec A\alpha_2]) : A \in \Sigma' \setminus \Delta,$$
$$\alpha_1, \alpha_2 \in (\Sigma' \setminus \Delta)^* \text{ and } |\alpha_1\alpha_2| \prec k\},$$

$$P_2 = \{(y_0, [\prec \alpha]) \to (y_0, [\succ \alpha]) : \alpha \in (\Sigma' \setminus \Delta)^*, |\alpha| \leqq k\},$$

$$P_3 = \{(y_0, [\alpha_1 \succ A\alpha_2]) \to (u_0 y_0 u_1 \ldots u_{m-1} y_0 u_m, [\alpha_1 \prec B_1 \ldots B_m \alpha_2])$$
$$: m \geqq 1, \quad A \to u_0 B_1 u_1 \ldots u_{m-1} B_m u_m, F \in P',$$
$$A, B_1, \ldots, B_m \in \Sigma' \setminus \Delta, u_0, u_1, \ldots, u_m \in \Delta^*,$$
$$\alpha_1, \alpha_2 \in (\Sigma' \setminus \Delta)^* \text{ and } |\alpha_1\alpha_2| \prec k, \ F \cap \text{alph}(\alpha_1 A \alpha_2) = \emptyset\},$$

$$P_4 = \{(y_0, [\alpha_1 \succ A\alpha_2]) \to (u, [\alpha_1 \prec \alpha_2]) : A \to u, F \in P',$$
$$F \cap \text{alph}(\alpha_1 A \alpha_2) = \emptyset, \ u \in \Delta^*, \ \alpha_1, \alpha_2 \in (\Sigma' \setminus \Delta)^*, \ |\alpha_1\alpha_2| \prec k\}.$$

It is easy to verify that $L(G') = L(G)$ and that G is of uncontrolled index k. □

Corollary 2. Let L be a language generated by a state grammar of finite index. Then L can be generated by a state grammar of finite index with (at most) two nonterminals.

Proof. Immediately follows from Theorem 4. and Theorem 3.22 in [6]. □

Theorem 5. Any recursively enumerable language can be generated by a state grammar with at most three nonterminals.

Proof. Let $L \in \mathscr{L}(\text{RE})$. We may assume by Lemma 2 that L is generated by front-end state grammar $G = (\Sigma, \Delta, S, \#, s_0, P)$. Let us denote by X the set of nonterminals of G excluding $\#$. Assume that $X = \{x_1, \ldots, x_n\}$ for some $n \geqq 1$. We define a coding function $\varphi : X \to \{0, 1\}^*$ by

$$\varphi(x_i) = 0^i 1 \quad \text{for } i = 1, \ldots, n.$$

Extend φ to a homomorphism of Σ^* into $\{0, 1, \#\}^*$ by $\varphi(y) = y$ for $y \in \Delta \cup \{\#\}$.

From the front-end property of the state grammar G it follows that the state set S can be partitioned into subsets S_1 and S_2. Let S_1 be the set of all states $s \in S$ such that for every $x \in X$ there is a production in P with left side (x, s). Now the state grammar

$$G' = (\Sigma', \Delta, S', \#, [\lambda, s_0], P')$$

is constructed in the following way.

$$\Sigma' = \{0, 1, \#\} \cup \Delta,$$

$$S' = \{0^i : 0 \leqq i \leqq n\} \times S,$$

$$P' = P_1' \cup P_2' \cup P_3', \quad \text{where}$$

$$P_1' = \{(0, [0^i, s]) \to (\lambda, [0^{i+1}, s]) : s \in S_1, 0 \leqq i < n\},$$

$$P_2' = \{(1, [0^i, s]) \to (\varphi(u), [\lambda, s']) : (x_i, s) \to (u, s') \in P\},$$

$$P_3' = \{(\#, [\lambda, s]) \to (\varphi(u), [\lambda, s']) : (\#, s) \to (u, s') \in P\}.$$

One can see that for every $s \in S$ and every $w \in \Sigma^*$ a derivation $(\#, s_0) \underset{G}{\Rightarrow}^* (w, s)$ holds iff $(\#, [\lambda, s_0]) \underset{G'}{\Rightarrow}^* (\varphi(w), [\lambda, s])$ holds. Since $\varphi(w) = w$ if $w \in \Delta^*$, we obtain the desired equality $L(G) = L(G')$. □

To conclude this section let us remark that it remains an open question whether the number of nonterminals (three) in Theorem 5 is minimal.

COMPUTING CENTRE
TECHNICAL UNIVERSITY
OBRÁNCŮ MIRU 21
BRNO
CZECHOSLOVAKIA

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

References

[1] DASSOW, J., Comparison of some types of regulated rewriting. Technological University Magdeburg, Department of Mathematics and Physics, Technical Report SMA 58/83 (1983).
[2] DASSOW, J. and PAUN, G., Further remarks on the complexity of regulated rewriting. Technological University Magdeburg, Department of Mathematics and Physics, Technical Report SMA 70/83 (1983).
[3] KASAI, T., A hierarchy between context-free and context sensitive languages. Journal of Computer and System Sciences 4 (1970), 492—508.
[4] KLEIJN, H. C. M., Selective substitution grammars based on context-free production. Ph. D. Thesis, University Of Leiden (1983).
[5] SALOMAA, A., Formal languages. Academic Press, New York 1973.
[6] VERMIER, D., On structural restrictions of ET0L Systems. Ph. D. Thesis, University of Antwerpen (1978).

(Received July 2, 1987)

A note on the generalized v_1-product

B. IMREH

A hierarchy of products was introduced in [1]. This hierarchy contains one kind of product, the v_i-product, for every positive integer i, and the work [1] deals with the isomorphic completeness with respect to the v_i-products. As regards another representations, the metric representation was studied in [6], [8]. The work [6] contains the characterization of the metrically complete systems with respect to the v_i-products. In [8] it is shown that the v_1-product is metrically equivalent to the general product. The works [2], [3], [4], [5] are devoted to the investigation of the homomorphic representation. In [3] and [4] some special compositions of the α_0-product and v_i-products was studied and it is proved that these compositions are just as strong as the general product with respect to the homomorphic representation. The work [5] deals with the commutative automata. It is shown that there are finite homomorphically complete systems with respect to the v_1-product for this class. In [2] the hierarchy of the v_i-products was investigated. It is proved that this hierarchy is proper as regards the homomorphic representations. Finally, the work [7] compares the isomorphic and homomorphic representation powers of α_i-products and v_j-products.

In this paper, connecting with the work [1], we give a sufficient condition for a system of automata to be isomorphically S-complete with respect to the generalized v_1-product. This condition is a special case of condition (2) of Theorem 2 in [1], but the construction of the automata from these systems is simpler than the general construction given in [1]. Since our work is closely related to [1], we shall use its notions and notations.

Our result is the following statement.

Theorem. A system Σ of automata is isomorphically S-complete with respect to the generalized v_1-product if Σ contains an automaton which has a state a and input word q such that the states $a,\ aq,\ \ldots,\ aq^{s-1}$ are pairwise different and $aq^s=a$ for some integer $s>1$.

Proof. Let us assume that Σ satisfies the condition. Then without loss of generality we may suppose that Σ contains an automaton $\mathbf{A}$ which has a state a and input word q such that $a,\ aq,\ \ldots,\ aq^{p-1}$ are pairwise different, $aq^p=a$, and p is a prime number. Let us denote by $0, 1, \ldots, p-1$ the states $a,\ aq,\ \ldots,\ aq^{p-1}$, respectively. Depending on p, we shall distinguish two cases.

Case 1. Let us suppose that $p=2$. By the proof of Theorem 2 in [1], it is enough to prove that for any $n\geqq 3$ the automaton $\mathbf{T}'_n$ can be simulated isomorphically by a generalized v_1-product of automata from Σ, where $\mathbf{T}'_n=(\{t_1, t_2, t_3\}, \{0, \ldots, n-1\}, \delta'_n)$ and

$$t_1(k) = k+1 \pmod n \quad (k = 0, \ldots, n-1),$$

$$t_2(0) = 1,\ t_2(1) = 0,\ t_2(k) = k \ (k = 2, \ldots, n-1),$$

$$t_3(0) = t_3(1) = 0,\ t_3(k) = k \ (k = 2, \ldots, n-1).$$

Now let $n\geqq 3$ be an arbitrary fixed integer. Let us take an integer k for which $2^k+1\geqq n$ holds and denote by m the number 2^k+1. Form the generalized v_1-product $\mathbf{A}^m(X, \varphi, \gamma)$ where

$$X = \{x_1, x_2, x_3\}\cup\{y_t: 0 \leqq t \leqq m-1\}$$

and the mappings γ and φ are defined in the following way:

$$\gamma(t) = \{t-1 \pmod m\} \quad (t = 0, \ldots, m-1),$$

$$\varphi_t(0, x_1) = q,\ \varphi_t(1, x_1) = q^2 \quad (t = 0, \ldots, m-1),$$

$$\varphi_t(0, x_2) = \varphi_t(1, x_2) = q^2 \quad \text{if} \quad 0 \leqq t \leqq m-3,$$

$$\varphi_t(0, x_2) = \varphi_t(1, x_2) = q \quad \text{if} \quad m-3 < t \leqq m-1,$$

$$\varphi_t(0, x_3) = q^2,\ \varphi_t(1, x_3) = q \quad \text{if} \quad t \neq m-2,$$

$$\varphi_{m-2}(0, x_3) = q, \quad \varphi_{m-2}(1, x_3) = q^2,$$

$$\varphi_t(0, y_j) = q^2,\ \varphi_t(1, y_j) = \begin{cases} q & \text{if} \quad t = j, \\ q^2 & \text{otherwise,} \end{cases} \quad (j = 0, \ldots, m-1;\ t = 0, \ldots, m-1)$$

Take the mappings:

$$\mu: \begin{array}{l} 0 \to (0, 0, \ldots, 1), \\ \vdots \\ m-1 \to (1, 0, \ldots, 0), \end{array}$$

$$\tau: \begin{array}{l} t_1 \to x_1^{m-2}, \\ t_2 \to x_2 y_1 \ldots y_{m-3} y_{m-1} y_{m-2} \ldots y_1 y_{m-1}, \\ t_3 \to y_0 y_1 \ldots y_{m-3} x_3. \end{array}$$

Now we show that $\mathbf{T}'_m$ can be simulated isomorphically by $\mathbf{A}^m(X, \varphi, \gamma)$ under μ and τ. Indeed, the validity of the equations $\mu(\delta'_m(j, t_l))=\delta_{\mathbf{A}^m}(\mu(j), \tau(t_l))$ $(l=2, 3; j=0, \ldots, m-1)$ follows from the definitions. To prove the validity of the equations $\mu(\delta'_m(j, t_1))=\delta_{\mathbf{A}^m}(\mu(j), \tau(t_1))$ $(j=0, \ldots, m-1)$ let us observe the following connection. If

$$(u_0, \ldots, u_{m-1})\in\{0, 1\}^m \quad \text{and}$$

$$\delta_{\mathbf{A}^m}((u_0, \ldots, u_{m-1}), x_1) = (v_0, \ldots, v_{m-1})$$

then

$$v_t = \delta_A(u_t, \varphi_t(u_{t-1 \pmod m}, x_1)) =$$

$$= u_t q^{u_{t-1 \pmod m}+1 \pmod 2} = u_t + u_{t-1 \pmod m} + 1 \pmod 2$$

holds for any $0 \leqq t \leqq m-1$. Now let us denote by $(v_0^{(s)}, \ldots, v_{m-1}^{(s)})$ the state $\delta_{A^m}((u_0, \ldots, u_{m-1}), x_1^s)$. Then using the above connection, by induction on s, it can be proved that

$$v_t^{(s)} = 1 + \sum_{j=0}^{s} \binom{s}{j} u_{t-j \pmod m} \pmod 2 \quad (t = 0, \ldots, m-1).$$

On the other hand, it is known that $\binom{p^k}{j} \equiv 0 \pmod p$ $(j = 1, \ldots, p^k - 1)$ holds for any prime $p > 1$ and integer $k \geqq 1$. Using this, by induction on j, one can show that

$$\binom{p^k - 1}{j} (-1)^j \equiv 1 \pmod p \quad (j = 0, \ldots, p^k - 1).$$

From this, by $p = 2$, we obtain

$$\binom{2^k - 1}{j} \equiv 1 \pmod 2 \quad (j = 0, \ldots, 2^k - 1).$$

Now let $0 \leqq i \leqq m-1$ be an arbitrary integer and let us denote by $(c_0, \ldots, c_{m-1})$ the state $\mu(i)$. Then

$$c_t = \begin{cases} 1 & \text{if } t = m-i-1, \\ 0 & \text{otherwise,} \end{cases} \quad (t = 0, \ldots, m-1).$$

Let $(c_0', \ldots, c_{m-1}')$ denote the state $\delta_{A^m}((c_0, \ldots, c_{m-1}), x_1^{m-2})$. Then, by the above equality for $v_t^{(s)}$, we obtain that

$$c_t' = 1 + \sum_{j=0}^{2^k-1} \binom{2^k - 1}{j} c_{t-j \pmod m} \pmod 2 \quad (t = 0, \ldots, m-1).$$

If $t = m-i-2 \pmod m$ then from the definition of c_t it follows that $c_{t-j \pmod m} = 0$ $(j = 0, \ldots, 2^k - 1)$, and so, $c'_{m-i-2 \pmod m} = 1$. If $t \neq m-i-2 \pmod m$ then among the elements $c_{t-j \pmod m}$ $(j = 0, \ldots, 2^k - 1)$ one and only one is different from 0, and so, $c_t' = 1 + \binom{2^k-1}{j} \pmod 2$ for some $0 \leqq j \leqq 2^k - 1$. Since $\binom{2^k-1}{j} \equiv 1 \pmod 2$ this implies the equality $c_t' = 0$. Summarizing, we obtained that

$$c_t' = \begin{cases} 1 & \text{if } t = m-i-2 \pmod m, \\ 0 & \text{otherwise,} \end{cases} \quad (t = 0, \ldots, m-1).$$

Now let us observe that $(c_0', \ldots, c_{m-1}') = \mu(i+1 \pmod m)$ and so,

$$\mu(\delta_m'(i, t_1)) = \mu(i+1 \pmod m) = (c_0', \ldots, c_{m-1}') = \delta_{A^m}((c_0, \ldots, c_{m-1}), x_1^{m-2}) =$$

$$= \delta_{A^m}((\mu(i), \tau(t_1))$$

which completes the proof of the Case 1.

Case 2. Let us suppose that $p>2$ and let $n\geqq 3$ be an arbitrary fixed integer again. Let k be an integer such that $p^k+1\geqq 2n$ and, let $s=p^k+1$ and $m=s/2$. Form the generalized v_1-product $\mathbf{A}^s(X, \varphi, \gamma)$ where

$$X=\{x_1, \ldots, x_s\}\cup\{y_r\colon 0\leqq r\leqq s-2\}\cup\{v_r, z_r\colon 0\leqq r\leqq s-4\}\cup\{w_r\colon 0\leqq r\leqq s-1\}$$

and the mappings γ and φ are defined in the following way: for any $t\in\{0, \ldots, s-1\}$, $j\in\{0, \ldots, p-1\}$, $r\in\{0, \ldots, s-1\}$

$$\gamma(t)=\{t-1 \pmod s\},$$

$$\varphi_t(j, x_1)=q^{p-1-j} \text{ if } 0\leqq j<p-1, \quad \varphi_t(p-1, x_1)=q^p,$$

$$\varphi_t(j, x_2)=q^{p-1} \text{ if } t\in\{s-3, s-2, s-1\}, \varphi_t(j, x_2)=q^p \text{ if } 0\leqq t<s-3,$$

$$\varphi_{s-3}(0, x_3)=q^2, \ \varphi_t(j, x_3)=q^p \text{ otherwise},$$

$$\varphi_{s-2}(0, x_4)=q, \quad \varphi_t(j, x_4)=q^p \text{ otherwise},$$

$$\varphi_t(p-1, x_5)=q \text{ if } t\neq 0 \text{ and } \varphi_t(j, x_5)=q^p \text{ otherwise},$$

$$\varphi_t(p-1, x_6)=q \text{ if } t\in\{s-2, s-1\} \text{ and } \varphi_t(j, x_6)=q^p \text{ otherwise},$$

$$\varphi_t(0, x_7)=q^{p-1} \text{ if } t=s-3 \text{ and } \varphi_t(j, x_7)=q^p \text{ otherwise},$$

$$\varphi_t(p-2, x_8)=\begin{cases} q & \text{if } t=s-1, \\ q^2 & \text{if } t\neq s-1, \end{cases} \text{ and } \varphi_t(j, x_8)=q^p \text{ otherwise},$$

$$\varphi_t(j, y_r)=\begin{cases} q^{p-1} & \text{if } t=r \text{ and } j=p-1, \\ q^p & \text{otherwise}, \end{cases} \quad (r=0, \ldots, s-2),$$

$$\varphi_t(j, v_r)=\begin{cases} q^2 & \text{if } t=r \text{ and } j=p-2, \\ q^p & \text{otherwise}, \end{cases} \quad (r=0, \ldots, s-4),$$

$$\varphi_t(j, z_r)=\begin{cases} q & \text{if } t=r \text{ and } j=p-1, \\ q^p & \text{otherwise}, \end{cases} \quad (r=0, \ldots, s-4),$$

$$\varphi_t(j, w_r)=\begin{cases} q^{p-2} & \text{if } t=r \text{ and } j=p-2, \\ q^p & \text{otherwise}, \end{cases} \quad (r=0, \ldots, s-1).$$

Take the mappings:

$$\mu\colon \begin{array}{l} 0 \to (0, 0, 0, \ldots, 0, 0, p-1), \\ 1 \to (0, 0, 0, \ldots, p-1, 0, 0), \\ \vdots \\ m-1 \to (0, p-1, 0, \ldots, 0, 0, 0), \end{array}$$

$$\tau\colon \begin{array}{l} t_1 \to x_1^{2(p^k-1)}, \\ t_2 \to x_2 y_2 \ldots y_{s-4} w_0 \ldots w_{s-4} x_3 x_4 v_{s-4} \ldots v_0 x_5 y_{s-2} x_6, \\ t_3 \to y_0 \ldots y_{s-4} x_7 w_{s-2} w_{s-1} w_0 \ldots w_{s-4} z_{s-4} \ldots z_0 x_8. \end{array}$$

Now we shall show that the automaton $\mathbf{T}'_m$ can be simulated isomorphically by $\mathbf{A}^s(X, \varphi, \gamma)$ under μ and τ.

The validity of $\mu(\delta'_m(j, t_l))=\delta_{\mathbf{A}^s}(\mu(j), \tau(t_l))$ $(l=2, 3; j=0, \ldots, m-1)$ can be checked by a simple computation. To prove the validity of $\mu(\delta'_m(j, t_1))=\delta_{\mathbf{A}^s}(\mu(j), \tau(t_1))$ let $(u_0, \ldots, u_{s-1})\in\{0, \ldots, p-1\}^s$ be arbitrary and let us denote by $(u_0^{(r)}, \ldots, u_{s-1}^{(r)})$ the state $\delta_{\mathbf{A}^s}((u_0, \ldots, u_{s-1}), x_1^r)$ for arbitrary integer $r\geqq 1$. Then

$$u_t^{(1)} = \delta_{\mathbf{A}}(u_t, \varphi_t(u_{t-1(\mathrm{mod}\, s)}, x_1)) = u_t q^{(p-1-u_{t-1(\mathrm{mod}\, s)})(\mathrm{mod}\, p)} = u_t - u_{t-1(\mathrm{mod}\, s)} - 1 \,(\mathrm{mod}\, p).$$

Using this, by induction on r, it can be proved that

$$u_t^{(r)} = -1 + \sum_{j=0}^{r} \binom{r}{j} (-1)^j u_{t-j(\mathrm{mod}\, s)} \,(\mathrm{mod}\, p) \quad (t = 0, \ldots, s-1)$$

Now let $i\in\{0, \ldots, m-1\}$ be arbitrary and let us denote by $(c_0, \ldots, c_{s-1})$, $(c'_0, \ldots, c'_{s-1})$, $(\bar{c}_0, \ldots, \bar{c}_{s-1})$ the states $\mu(i)$, $\delta_{\mathbf{A}^s}(\mu(i), x_1^{\{p^k-1\}})$, $\delta_{\mathbf{A}^s}(\mu(i), x_1^{2(p^k-1)})$, respectively. Then from the definition of μ,

$$c_t = \begin{cases} p-1 & \text{if} \quad t = s-2i-1, \\ 0 & \text{otherwise.} \end{cases} \quad (t = 0, \ldots, s-1).$$

Consider the state $(c'_0, \ldots, c'_{s-1})$. By the above equality for $u_t^{(r)}$, we obtain that

$$c'_t = -1 + \sum_{j=0}^{p^k-1} \binom{p^k-1}{j} (-1)^j c_{t-j(\mathrm{mod}\, s)} \,(\mathrm{mod}\, p) \quad (t = 0, \ldots, s-1).$$

If $t=s-2i-2 \,(\mathrm{mod}\, s)$ then from the definition of c_t it follows that $c_{t-j(\mathrm{mod}\, s)}=0$ $(j=0, \ldots, p^k-1)$, and so, $c'_{s-2i-2(\mathrm{mod}\, s)}=p-1$. If $t\neq s-2i-2 \,(\mathrm{mod}\, s)$ then among the elements $c_{t-j(\mathrm{mod}\, s)}$ $(j=0, \ldots, p^k-1)$ exactly one element is different from 0 and this element is equal to $p-1$, and so, $c'_t=-1+\binom{p^k-1}{j}(-1)^j(p-1)$ for some $0\leqq j\leqq p^k-1$. From this, by $\binom{p^k-1}{j}(-1)^j\equiv 1 \,(\mathrm{mod}\, p)$ $(j=0, \ldots, p^k-1)$, we obtain that $c'_t=p-2$. Therefore

$$c'_t = \begin{cases} p-1 & \text{if} \quad t = s-2i-2, \\ p-2 & \text{otherwise,} \end{cases} \quad (t = 0, \ldots, s-1).$$

Now consider the state $(\bar{c}_0, \ldots, \bar{c}_{s-1})$.

$$\bar{c}_t = -1 + \sum_{j=0}^{p^k-1} \binom{p^k-1}{j} (-1)^j c'_{t-j(\mathrm{mod}\, s)} \,(\mathrm{mod}\, p) \;\; (t = 0, \ldots, s-1).$$

If $t=s-2(i+1)-1 \,(\mathrm{mod}\, s)$, then $c'_{t-j(\mathrm{mod}\, s)}=p-2$ $(j=0, \ldots, p^k-1)$. On the other hand $\binom{p^k-1}{j}(-1)^j\equiv 1 \,(\mathrm{mod}\, p)$, and so, we obtain that $\bar{c}_{s-2(i+1)-1(\mathrm{mod}\, s)}=p-1$. If $t\neq s-2(i+1)-1 \,(\mathrm{mod}\, s)$ then among the elements $c'_{t-j(\mathrm{mod}\, s)}$ $(j=0, \ldots, p^k-1)$ exactly one element is different from $p-2$ and this element is equal to $p-1$. From

this, by $\binom{p^k-1}{j}(-1)^j \equiv 1 \pmod p$ $(j=0, \ldots, p^k-1)$, we get that $\bar{c}_t=0$. Therefore,

$$\bar{c}_t = \begin{cases} p-1 & \text{if } t = s-2(i+1)-1 \pmod s, \\ 0 & \text{otherwise,} \end{cases} \qquad (t = 0, \ldots, s-1).$$

Observe that $(\bar{c}_0, \ldots, \bar{c}_{s-1}) = \mu(i+1 \pmod m)$, and so,

$$\mu(\delta'_m(i, t_1)) = \mu(i+1 \pmod m) = (\bar{c}_0, \ldots, \bar{c}_{s-1}) = \delta_{A^s}(\mu(i), x_1^{2(p^k-1)}) =$$
$$= \delta_{A^s}(\mu(i), \tau(t_1))$$

which completes the proof of Case 2. This ends the proof of our Theorem.

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

References

[1] DÖMÖSI, P. and IMREH, B., On ν_t-products of automata, Acta Cybernet., 6 (1983), 149—162.
[2] DÖMÖSI, P. and ÉSIK Z., On the hierarchy of ν_t-products of automata, Acta Cybernet., 8 (1988), 253—258.
[3] ÉSIK, Z., Loop products and loop-free products, Acta Cybernet., 8 (1987), 45—48.
[4] ÉSIK, Z. and GÉCSEG, F., On α_0-products and α_2-products, Theoret. Comput. Sci., 48 (1986), 1—8.
[5] GÉCSEG, F., On ν_t-products of commutative automata, Acta Cybernet., 7 (1985), 55—59.
[6] GÉCSEG, F. Metric representations by ν_t-products, Acta Cybernet., 7 (1985), 203—209.
[7] GÉCSEG, F. and IMREH, B., A comparison of α_t-products and ν_t-products, Foundations of Control Engineering, 12 (1987), 3—9.
[8] GÉCSEG, F. and IMREH, B., On metric equivalence of ν_t-products, Acta Cybernet., 8 (1987), 129—134.

(Received July 10, 1987)

On the hierarchy of v_i-products of automata

P. DÖMÖSI and Z. ÉSIK

In order to decrease the feedback complexity of the Gluškov-type product of automata, a hierarchy of products was introduced by F. Gécseg in [6]. This hierarchy, referred to as the α_i-hierarchy, contains one product concept for each nonnegative integer i. The α_0-product is also known as the loop-free product, the series-parallel composition or the cascade composition [11, 1, 13]. Another hierarchy, the v_i-hierarchy appears in [2], where i is any positive integer. Using the main result of [3] it has been shown in [5] that for homomorphic realization the α_i-hierarchy collapses at $i=2$. One of the aims of the present paper is to show that the v_i-hierarchy is strict. For some classes of automata even the v_1-product has a surprising power. This has been demonstrated in [2] for the first time and then in [7, 4]. In fact there are classes of automata for which the v_1-product is much stronger than the α_0-product. In this paper we prove that the opposite can also be true for some classes.

An automaton is a system $\mathbf{A}=(A, X, \delta)$ with finite nonempty sets A and X, the state set and the input set, and transition $\delta: A\times X\to A$. The transition is also used in the extended sense, i.e. as a map $\delta: A\times X^*\to A$ where X^* is the free monoid of all words over X. Let $\mathbf{A}_j=(A_j, X_j, \delta_j)$ $(j=1, \ldots, n, n\geqq 0)$ be automata, and take a family of feedback functions

$$\varphi_j: A_1\times\ldots\times A_n\times X\to X_j$$

$(j=1, \ldots, n)$, where X is a new finite nonempty set of input letters. The Gluškov-type product (cf. [10]) of the automata $\mathbf{A}_j$ with respect to the feedback functions φ_j is defined to be the automaton

$$\mathbf{A}_1\times\ldots\times\mathbf{A}_n(X, \varphi)$$

with state set $A=A_1\times\ldots\times A_n$, input set X and transition δ given by

$$\mathrm{pr}_j(\delta(a, x)) = \delta_j(\mathrm{pr}_j(a), \varphi_j(a, x)),$$

for all $a\in A$, $x\in X$ and $1\leqq j\leqq n$. The Gluškov-type product is also called the general product, or g-product, for short. Let $i\geqq 1$ be any integer. Following [2], the above defined g-product is called a v_i-product if for every integer $j=1, \ldots, n$ there is a set

$v(j) \subseteq \{1, \ldots, n\}$ with cardinality not exceeding i such that each feedback function

$$\varphi_j(a_1, \ldots, a_n, x)$$

is independent of any state variable a_k with $k \notin v(j)$. For the definition of the α_i-products see [6, 8].

Let $\mathscr{K}$ be a class of automata. We shall use the following notations:

$\mathbf{P}_g(\mathscr{K})$:=all g-products of automata from $\mathscr{K}$;
$\mathbf{P}_{\alpha_i}(\mathscr{K})$:=all α_i-products of automata from $\mathscr{K}$;
$\mathbf{P}_{v_i}(\mathscr{K})$:=all v_i-products of automata from $\mathscr{K}$;
$\mathbf{S}(\mathscr{K})$:=all subautomata of automata from $\mathscr{K}$;
$\mathbf{H}(\mathscr{K})$:=all homomorphic images of automata from $\mathscr{K}$.

In the sequel we shall also make use of a few simple facts.

Lemma 1. For every class $\mathscr{K}$, $\mathbf{HSP}_{\alpha_0}(\mathscr{K})$ is the smallest class containing $\mathscr{K}$ and closed under the operators $\mathbf{H}$, $\mathbf{S}$ and $\mathbf{P}_{\alpha_0}$.

The proof of Lemma 1 can be found in [8]. We note that a similar statement is true for the g-product.

Lemma 2. Let $\mathbf{A}=\mathbf{A}_1\times\ldots\times\mathbf{A}_n(X, \varphi)$ be a v_i-product of automata $\mathbf{A}_j=(A_j, X_j, \delta_j)$. Let π be a permutation of the set $\{1, \ldots, n\}$. There exists a v_i-product $\mathbf{A}'=\mathbf{A}_{\pi(1)}\times\ldots\times\mathbf{A}_{\pi(n)}(X, \varphi')$ which is isomorphic to $\mathbf{A}$, an isomorphism $\mathbf{A}\to\mathbf{A}'$ is the map $(a_1, \ldots, a_n)\mapsto(a_{\pi(1)}, \ldots, a_{\pi(n)})$ $((a_1, \ldots, a_n)\in A_1\times\ldots\times A_n)$.

Lemma 3. Let $\mathbf{A}=\mathbf{A}_1\times\ldots\times\mathbf{A}_n(X, \varphi)$ be a v_i-product with $n\geqq 1$ and components $\mathbf{A}_j=(A_j, X_j, \delta_j)$. Let $\mathbf{B}=(B, X, \delta)$ be a subautomaton of $\mathbf{A}$, $j_0\in\{1, \ldots, n\}$ a fixed integer and $a\in A_{j_0}$. If $\mathrm{pr}_{j_0}(b)=a$ for all $b\in B$ then there is a v_i-product $\mathbf{A}'=\mathbf{A}_1\times\ldots\times\mathbf{A}_{j_0-1}\times\mathbf{A}_{j_0+1}\times\ldots\times\mathbf{A}_n(X, \varphi')$ such that $\mathbf{A}'$ contains a subautomaton $\mathbf{B}'$ isomorphic to $\mathbf{B}$, an isomorphism $\mathbf{B}\to\mathbf{B}'$ is the map $(a_1, \ldots, a_{j_0-1}, a, a_{j_0+1}, \ldots, a_n)\mapsto$ $\mapsto(a_1, \ldots, a_{j_0-1}, a_{j_0+1}, \ldots, a_n)$.

We are now ready to state our main result.

Theorem. There exists a class $\mathscr{K}$ of automata such that $\mathbf{HSP}_{v_i}(\mathscr{K})\subset$ $\subset\mathbf{HSP}_{v_{i+1}}(\mathscr{K})\subset\mathbf{HSP}_{\alpha_0}(\mathscr{K})$ holds for all $i\geqq 1$.

Proof. Let p be a prime number. We define an automaton $\mathbf{D}_p=(D_p, \{x, y\}, \delta)$ as follows:

$$D_p = \{0, \ldots, p\},$$

$$\delta(j, x) = \begin{cases} j+1 \bmod p & \text{if } j < p, \\ p & \text{if } j = p, \end{cases}$$

$$\delta(j, y) = p, \quad j\in D_p.$$

Let $\mathscr{K}=\{D_p | p \text{ is a prime}\}$. We set out to prove the following properties of $\mathscr{K}$.

(1) $\mathbf{HSP}_g(\mathscr{K})\subseteq\mathbf{HSP}_{\alpha_0}(\mathscr{K})$,

(2) $\mathbf{HSP}_{v_i}(\mathscr{K})\subset\mathbf{HSP}_{v_{i+1}}(\mathscr{K})$ for all $i\geqq 1$.

Supposing (1) and (2) have been shown, the proof is easily completed. Since $\mathbf{HSP}_{\nu_{i+1}}(\mathscr{K}) \subseteq \mathbf{HSP}_g(\mathscr{K})$ holds obviously, from (1) we have $\mathbf{HSP}_{\nu_{i+1}}(\mathscr{K}) \subseteq \subseteq \mathbf{HSP}_{\alpha_0}(\mathscr{K})$, which in turn implies $\mathbf{HSP}_{\nu_i}(\mathscr{K}) \subset \mathbf{HSP}_{\alpha_0}(\mathscr{K})$ by (2). Thus $\mathbf{HSP}_{\nu_i}(\mathscr{K}) \subset \mathbf{HSP}_{\alpha_0}(\mathscr{K})$ for all $i \geqq 1$.

Proof of (1). For every prime number p, define $\mathbf{C}_p = (C_p, \{x\}, \delta)$ by

$$C_p = \{0, \ldots, p-1\},$$

$$\delta(j, x) = j+1 \bmod p, \; j \in C_p.$$

Moreover, let $\mathbf{E} = (E, \{x, y\}, \delta)$ with $E = \{0, 1\}$, $\delta(0, x) = 0$, $\delta(0, y) = \delta(1, x) = \delta(1, y) = 1$. Thus $\mathbf{C}_p$ is the counter with length p and $\mathbf{E}$ is the elevator. Set

$$\mathscr{K}' = \{\mathbf{C}_p \mid p \text{ is a prime}\} \cup \{\mathbf{E}\}.$$

From the proof of the main result of [5] we have $\mathbf{HSP}_g(\mathscr{K}) = \mathbf{HSP}_{\alpha_0}(\mathscr{K}')$. To end the proof, by Lemma 1, it suffices to show that $\mathscr{K}' \subseteq \mathbf{HSP}_{\alpha_0}(\mathscr{K})$. That is however obvious for we have $\mathbf{C}_p \in \mathbf{S}(\{\mathbf{D}_p\})$ and $\mathbf{E} \in \mathbf{H}(\{\mathbf{D}_p\})$, each prime number p.

Proof of (2). Let $i \geqq 1$ be any integer and $m = \prod (p_j \mid j = 1, \ldots, i+1)$, where p_j is the j-th prime. Define $\mathbf{M} = (M, \{x, y\}, \delta)$ to be the automaton with

$$M = \{0, \ldots, m\},$$

$$\delta(j, x) = \begin{cases} j+1 \bmod m & \text{if } j < m, \\ m & \text{if } j = m, \end{cases}$$

$$\delta(j, y) = \begin{cases} j+1 \bmod m & \text{if } 0 < j < m, \\ m & \text{if } j = 0 \text{ or } j = m, \end{cases}$$

for all $j \in M$. We prove that $\mathbf{M} \notin \mathbf{HSP}_{\nu_i}(\mathscr{K})$ while $\mathbf{M} \in \mathbf{HSP}_{\nu_{i+1}}(\mathscr{K})$.

Assume that, on the contrary, $\mathbf{M} \in \mathbf{HSP}_{\nu_i}(\mathscr{K})$. Let

$$\mathbf{D}_{q_1} \times \ldots \times \mathbf{D}_{q_n}(\{x, y\}, \varphi)$$

be a ν_i-product of automata from $\mathscr{K}$ that contains a subautomaton $\mathbf{A} = (A, \{x, y\}, \delta)$ which is mapped onto $\mathbf{M}$ under a suitable homomorphism h. We may choose n to be the least (positive) integer with the above property, i.e. if a ν_i-product of automata from $\mathscr{K}$ contains a subautomaton that can be mapped homomorphically onto $\mathbf{M}$ then the number of factors of that product is at least n. Also, the subautomaton $\mathbf{A}$ can be chosen such that none of its proper subautomata is mapped homomorphically onto $\mathbf{M}$.

Let us write A as the disjoint union $A = A_0 \cup A_1$ where $A_0 = h^{-1}(M - \{m\})$ and $A_1 = h^{-1}(\{m\})$. Let $a \in A_0$ be a state. Since a is a generator of A, if $\mathrm{pr}_j(a) = q_j$ for an integer $j = 1, \ldots, n$, then $\mathrm{pr}_j(b) = q_j$ for all $b \in A$. By Lemma 3, there exists a ν_i-product

$$\mathbf{D}_{q_1} \times \ldots \times \mathbf{D}_{q_{j-1}} \times \mathbf{D}_{q_{j+1}} \times \ldots \times \mathbf{D}_{q_n}(\{x, y\}, \varphi')$$

that contains a subautomaton isomorphic to $\mathbf{A}$. This contradicts the minimality of n. Thus $\mathrm{pr}_j(a) \neq q_j$ for all $a \in A_0$ and $j = 1, \ldots, n$. Suppose now that there is an $a \in A_1$ such that for all $j = 1, \ldots, n$ we have $\mathrm{pr}_j(a) \neq q_j$. Let $b \in A_0$ be a state and $u \in \{x, y\}^*$

a word with $\delta(b, u)=a$. Let $v=x^k$ where k denotes the length of u. We have $c=\delta(b, v)\in A_0$, henceforth $\mathrm{pr}_j(c)\neq q_j$ for all $j=1, \ldots, n$. The special structure of the automata $\mathbf{D}_{q_j}$ guarantees that $a=c$. This contradiction yields that for every $a\in A_1$ there is an integer $j(1\leqq j\leqq n)$ with $\mathrm{pr}_j(a)=q_j$.

Let $a_0=(a_{0,1}, \ldots, a_{0,n}), \ldots, a_{q-1}=(a_{q-1,1}, \ldots, a_{q-1,n})$ be all the states in A_0, so that $a_{t,j}\neq q_j$, $0\leqq t\leqq q-1$, $1\leqq j\leqq n$. By the minimality of $\mathbf{A}$ and the special structure of the automata $\mathbf{D}_{q_j}$ it follows that the letter x induces a cyclic permutation of the states a_t, say $\delta(a_t, x)=a_{t+1 \bmod q}$. Also q is the l.c.m. of the primes $q_1, \ldots, q_n$. Since h is a homomorphism of $\mathbf{A}$ onto $\mathbf{M}$, we have $q\equiv 0 \bmod m$. Without loss of generality we may suppose $\delta(a_0, y)=a\in A_1$. Thus $\mathrm{pr}_j(a)=q_j$ for some j. By Lemma 2, we may take $j=1$. Since $\mathrm{pr}_1(a)=q_1$ we must have $\varphi_1(a_0, y)=y$. Let $v(1)=\{j_1, \ldots, j_k\}$, so that $k\leqq i$. Define $\bar{q}$ to be the l.c.m. of the primes on the list $q_{j_1}, \ldots, q_{j_k}$. Obviously then $q\equiv 0 \bmod \bar{q}$. Since m is the product of $i+1$ distinct primes and $\bar{q}$ is the product of at most i distinct primes, from $q\equiv 0 \bmod m$ and $q\equiv 0 \bmod \bar{q}$ we obtain $\bar{q}<q$. Let us now consider the state $a_{\bar{q}}=(a_{\bar{q},1}, \ldots, a_{\bar{q},n})$. For every $l=1, \ldots, k$ we have $\delta(a_{0,j_l}, x^{\bar{q}})=a_{\bar{q},j_l}\neq q_{j_l}$. Since $\bar{q}\equiv 0 \bmod q_{j_l}$ we see that $a_{\bar{q},j_l}=a_{0,j_l}$. Since we have a v_i-product it follows that $\varphi_1(a_{\bar{q}}, y)=\varphi_1(a_0, y)=y$. We conclude $\delta(a_{\bar{q}}, y)\in A_1$. Since h is a homomorphism of $\mathbf{A}$ onto $\mathbf{M}$ we see that $\bar{q}\equiv 0 \bmod m$. This is however clearly impossible for m is the product of $i+1$ distinct primes and $\bar{q}$ is the product of at most i distinct primes.

We still have to show that $\mathbf{M}\in\mathbf{HSP}_{v_{i+1}}(\mathscr{K})$. For this define the g-product

$$\mathbf{A}=(A, X, \delta)=\mathbf{D}_{p_1}\times\ldots\times\mathbf{D}_{p_{i+1}}(\{x, y\}, \varphi)$$

by

$$\varphi_j(a_1, \ldots, a_{i+1}, x)=x,$$

$$\varphi_j(a_1, \ldots, a_{i+1}, y)=\begin{cases} y & \text{if } a_1=\ldots=a_{i+1}=0, \\ x & \text{otherwise.}\end{cases}$$

Since the number of factors is $i+1$, this g-product is also a v_{i+1}-product. Define

$$A_0=\{a\in A \mid \mathrm{pr}_j(a)\neq p_j \quad \text{for all} \quad j=1, \ldots, i+1\},$$

$$A_1=A-A_0.$$

For an $a=(a_1, \ldots, a_{i+1})\in A_0$ let $h(a)=t$ be that integer $0\leqq t<m$ with $t\equiv a_j \bmod p_j$, $j=1, \ldots, i+1$. If $a\in A_1$ put $h(a)=m$. The mapping h is easily seen to be a homomorphism of $\mathbf{A}$ onto $\mathbf{M}$. □

Remark. It is said that an automaton $\mathbf{A}=(A, X, \delta)$ satisfies the Letičevskiĭ criterion if there exist a state $a\in A$, input letters $x_1, x_2\in X$ and words $u_1, u_2\in X^*$ with $\delta(a, x_1)\neq\delta(a, x_2)$ and $\delta(a, x_1u_1)=\delta(a, x_2u_2)=a$. If only $\delta(a, x_1)\neq\delta(a, x_2)$ and $\delta(a, x_1u)=a$ hold for some $a\in A$, $x_1, x_2\in X$ and $u\in X_1^*$, we say that $\mathbf{A}$ satisfies the semi-Letičevskiĭ criterion. The above definitions extend to classes of automata: a class $\mathscr{K}$ satisfies the Letičevskiĭ criterion or the semi-Letičevskiĭ criterion if one of its members satisfies it. By a classical result in [12], $\mathbf{HSP}_g(\mathscr{K})$ is the class of all automata if and only if $\mathscr{K}$ satisfies the Letičevskiĭ criterion. It has been shown in [3] that the same is true for the α_2-product. If $\mathscr{K}$ does not satisfy the semi-Letičevskiĭ criterion then, by the proof of the main result in [5], $\mathbf{HSP}_g(\mathscr{K})=\mathbf{HSP}_{\alpha_0}(\mathscr{K})$. Also $\mathbf{HSP}_g(\mathscr{K})=\mathbf{HSP}_{v_1}(\mathscr{K})$ in this case as shown in [9]. Suppose now that $\mathscr{K}$

satisfies the semi-Letičevskiĭ criterion but does not satisfy the Letičevskiĭ criterion. In [5] it is proved that for every such $\mathscr{K}$ we have $\mathbf{HSP}_g(\mathscr{K})=\mathbf{HSP}_{\alpha_1}(\mathscr{K})$. The ν_i-products behave quite differently. The class $\mathscr{K}$ given in the proof of our Theorem satisfies the semi-Letičevskiĭ criterion but does not satisfy the Letičevskiĭ criterion, moreover, there exists no integer $i\geqq 1$ with $\mathbf{HSP}_g(\mathscr{K})=\mathbf{HSP}_{\nu_i}(\mathscr{K})$.

Open problems. (1) Suppose that $\mathscr{K}$ satisfies the Letičevskiĭ criterion. Does there exist an integer $i\geqq 1$ with $\mathbf{HSP}_g(\mathscr{K})=\mathbf{HSP}_{\nu_i}(\mathscr{K})$? (2) Does there exist an integer $i\geqq 1$ such that $\mathbf{HSP}_g(\mathscr{K})=\mathbf{HSP}_{\nu_i}(\mathscr{K})$ whenever $\mathscr{K}$ satisfies the Letičevskiĭ criterion? What is the least such i, if it exists?

INSTITUTE OF MATHEMATICS
L. KOSSUTH UNIVERSITY
EGYETEM TÉR 1
4010 DEBRECEN
HUNGARY

BOLYAI INSTITUTE
A. JÓZSEF UNIVERSITY
ARADI V. TERE 1
6720 SZEGED
HUNGARY

References

[1] Arbib, M. A. (ed), Machines, languages and semigroups, with a major contribution by K. Krohn and J. L. Rhodes, Academic Press, 1968.
[2] Dömösi, P. and Imreh, B., On ν_i-products of automata, Acta Cybernetica, 6 (1983), 149—162.
[3] Ésik, Z., Homomorphically complete classes of automata with respect to the α_2-product, Acta Sci. Math., 48 (1985), 135—141.
[4] Ésik, Z., Loop products and loop-free products, Acta Cybernetica, 8 (1978), 45—48.
[5] Ésik, Z. and Horváth, Gy., The α_2-product is homomorphically general, Papers on Automata Theory, V (1983), 49—62.
[6] Gécseg, F., Composition of automata, 2nd Colloq. Automata, Languages and Programming, 1974, LNCS 14 (1974), 351—363.
[7] Gécseg, F., On ν_1-products of commutative automata, Acta Cybernetica, 7 (1984), 55—59.
[8] Gécseg, F., Products of automata, Springer Verlag, 1986.
[9] Gécseg, F. and Imreh, B., On metric equivalence of ν_i-products, Acta Cybernetica, 8 (1987), 129—134.
[10] Glushkov, V. M. [Глушков, В. М.], Абстрактная теория автоматов, Успехи Матем. Наук, 16:5 (101), (1961), 3—62.
[11] Hartmanis, J. and Stearns, R. E., Algebraic structure theory of sequential machines, Prentice-Hall, 1966.
[12] Letičevskiĭ, A. A. [Летичевский, А. А.] Условия полноты для конечных автоматов, Журнал Выч. Мат. и Мат. Физ., 1 (1961), 702—710.
[13] Zeiger, H. B., Cascade synthesis of finite state machines, Information and Control, 10 (1967), 419—433.

(Received March 7, 1987)

On ranges of compositions of deterministic root-to-frontier tree transformations

Z. FÜLÖP and S. VÁGVÖLGYI

1. Introduction

In [3] we have proved that $\mathcal{DR}^2 = \mathcal{DR}^n$ for every $n \geqq 2$ where $\mathcal{DR}$ is the class of all deterministic root-to-frontier tree transformations. This result motivated us for examining whether the set $S = \{\mathcal{DR}, \mathcal{NDR}, \mathcal{LDR}, \mathcal{LNDR}, \mathcal{H}, \mathcal{NH}, \mathcal{LH}\}$ generates, with composition $\circ$, a finite or infinite set of tree transformation classes. Here $\mathcal{H}$ is the class of all homomorphism tree transformations, moreover the linear, nondeleting and linear-nondeleting subclasses of a class are denoted by prefixing the class by $\mathcal{L}$, $\mathcal{N}$ and $\mathcal{LN}$, respectively. We note that the enlargement of S by $\mathcal{LNH}$ has no effect on the generated set $[S] = \{\mathcal{H}_1 \circ \ldots \circ \mathcal{H}_n | n \geqq 1, \ \mathcal{H}_i \in S \text{ for } 1 \leqq i \leqq n\}$ since, for each $\mathcal{C} \in S$, $\mathcal{C} \circ \mathcal{LNH} = \mathcal{LNH} \circ \mathcal{C} = \mathcal{C}$.

In Theorem 12 of [3] we obtained a characterization for the set $[S]$, by means of which we proved that $[S]$ is infinite if and only if the hierarchy $\{(\mathcal{LNDR} \circ \mathcal{NH})^n\}$ is proper, which was shown in [6].

In this paper we examine the set of surface set classes $[S](\mathcal{R}ec) = \{\mathcal{C}(\mathcal{R}ec) | \mathcal{C} \in [S]\}$ as well as the set of classes of tree transformation languages $\mathrm{yd}([S](\mathcal{R}ec)) = \{\mathrm{yd}(\mathcal{T}) | \mathcal{T} \in [S](\mathcal{R}ec)\}$. ($\mathcal{R}ec$ is the class of all recognizable forests and yd is the operation "taking the string formed by the leaves" for trees.) We show that, although $\langle [S], \subseteqq \rangle$, as a poset, contains unrelated classes, $[S](\mathcal{R}ec)$ forms a chain with respect to inclusion with least element $\mathcal{R}ec$ and greatest element $\mathcal{DR}(\mathcal{R}ec)$. We also prove that, in this chain, $\mathcal{NDR}(\mathcal{R}ec)$ is properly contained in $\mathcal{DR}(\mathcal{R}ec)$ while the problem whether $[S](\mathcal{R}ec)$ is finite or infinite remains open. However, we show that the chain $\langle \mathrm{yd}([S](\mathcal{R}ec)), \subseteqq \rangle$ consists of exactly three elements.

2. Preliminaries

This paper is sequel to [3] and [6]. For notions and notations the reader is advised to consult with these works. Here we recall only the main results of [3] and [6] and introduce the terminology used exclusively in this paper.

We specify a special function symbol ε of arity 0 which either belongs to a ranked alphabet F or not.

If $p \in T_F$ is a tree then the yield $\mathrm{yd}(p) \subseteq F_0^*$ of p is defined inductively as follows:

(a) for $p \in F_0$, $\mathrm{yd}(p) = \lambda$ if $p = \varepsilon$ and $\mathrm{yd}(p) = p$ otherwise;

(b) for $p = f(p_1, \ldots, p_m)$, with $f \in F_m$ and $p_1, \ldots, p_m \in T_F$, $\mathrm{yd}(p) = \mathrm{yd}(p_1) \ldots \mathrm{yd}(p_m)$.

We call the attention of the reader not to confuse $\mathrm{yd}(p)$ with $\mathrm{fr}(p)$ defined in [3] and [6] and called the frontier of a tree p.

Subsets of T_F are called forests. If $T \subseteq T_F$ is a forest then $\mathrm{yd}(T) = \{\mathrm{yd}(p) | p \in T\}$ and, for a class $\mathscr{T}$ of forests we put $\mathrm{yd}(\mathscr{T}) = \{\mathrm{yd}(T) | T \in \mathscr{T}\}$.

In [6] we defined the set of paths $\mathrm{path}(p) \subseteq N^*$ for a tree $p \in T_F(Y)$. Here we shall consider two distinguished elements, the longest leftmost path $\mathrm{llp}(p)$ and the longest rightmost path $\mathrm{lrp}(p)$ of path (p) which are defined in the following way:

(a) if $p \in Y \cup F_0$ then $\mathrm{llp}(p) = \mathrm{lrp}(p) = \lambda$,

(b) if $p = f(p_1, \ldots, p_m)$ for some $m \geqq 1$, $f \in F_m$ and $p_1, \ldots, p_m \in T_F(Y)$ then $\mathrm{llp}(p) = 1\, \mathrm{llp}(p_1)$ and $\mathrm{lrp}(p) = n\, \mathrm{lrp}(p_n)$.

Let $\tau \subseteq T_F \times T_G$ be a tree transformation. The range of τ, defined as usual, is denoted by $\mathrm{ran}(\tau)$. Let $T \subseteq T_F$ be a forest. The image $\tau(T)$ of T under τ is the set $\{q \in T_G | (p, q) \in \tau \text{ for some } p \in T\}$.

For a class $\mathscr{C}$ of tree transformations and a class $\mathscr{T}$ of forests we set $\mathrm{ran}(\mathscr{C}) = \{\mathrm{ran}(\tau) | \tau \in \mathscr{C}\}$ and $\mathscr{C}(\mathscr{T}) = \{\tau(T) | \tau \in \mathscr{C} \text{ and } T \in \mathscr{T}\}$.

We denote by $\mathscr{R}ec$ the class of all recognizable forests (c.f. [4]).

Again, let $\mathscr{C}$ be a class of tree transformations.

The class of surface sets of $\mathscr{C}$ is the class $\mathscr{C}(\mathscr{R}ec)$ of forests, moreover, the class of tree transformation languages of $\mathscr{C}$ is the class $\mathrm{yd}(\mathscr{C}(\mathscr{R}ec))$ of languages.

If $\tau \subseteq T_F \times T_G$ is a tree transformation then the tree-to-string transformation τ_{tts} underlying τ is $\tau_{\mathrm{tts}} = \{(p, \mathrm{yd}(q)) | (p, q) \in \tau\}$. Thus $\tau_{\mathrm{tts}} \subseteq T_F \times G_0^*$. Analogously, for a class $\mathscr{C}$ of tree transformations we define $\mathscr{C}_{\mathrm{tts}} = \{\tau_{\mathrm{tts}} | \tau \in \mathscr{C}\}$.

We recall that the composition $\mathscr{C}_1 \circ \mathscr{C}_2$ of two tree transformation classes was defined in the order "first $\mathscr{C}_1$ and then $\mathscr{C}_2$" (c.f. [3], [6]). Thus we have $(\mathscr{C}_1 \circ \mathscr{C}_2)_{\mathrm{tts}} = \mathscr{C}_1 \circ \mathscr{C}_{2\mathrm{tts}}$ and, for any class $\mathscr{T}$ of forests $\mathrm{yd}(\mathscr{C}_1(\mathscr{T})) = \mathscr{C}_{1\mathrm{tts}}(\mathscr{T})$.

Let $\{\mathscr{C}_n | n = 1, 2, \ldots\}$ be a set of classes. We say that $\{\mathscr{C}_n | n = 1, 2, \ldots\}$, or $\{\mathscr{C}_n\}$ for short, is a hierarchy if $\mathscr{C}_n \subseteq \mathscr{C}_{n+1}$ for each $n \geqq 1$. This hierarchy is proper if $\mathscr{C}_n \subset \mathscr{C}_{n+1}$.

Now we introduce some technical details which, hopefully, make easier to understand the proofs in this paper.

Consider a DR transducer $\mathfrak{A} = (F, A, G, P, a_0)$ and a rule $af(x_1, \ldots, x_m) \to q$ in P. In this paper q is considered as an element of $T_G(A \times X_m)$ rather than $T_G(A(X_m))$. This is important when speaking about the height $h(q)$ of the right-hand side of a rule. (For the definition of height, see [3] or [6].) Moreover, we extend yd for the elements $T_G(A \times X_m)$ as follows: $\mathrm{yd}(q) = q$ if $q \in A \times X_m$ and otherwise $\mathrm{yd}(q)$ is defined in the same way as if q were in T_G, see above. Thus if q is the right-hand side of the above rule then $\mathrm{yd}(q)$ can be written in the form $w_0(a_1, x_{i_1}) w_1 \ldots (a_n, x_{i_n}) w_n$ for some $n \geqq 0$, $w_0, \ldots, w_n \in G_0^*$, $a_1, \ldots, a_n \in A$ and $x_{i_1}, \ldots, x_{i_n} \in X_m$.

The length of a string w will be denoted by $|w|$. The following abbreviated notation will also be used. Let F and G be disjoint ranked alphabets, let $f \in F_m$ with $m \geqq 0$ and $w \in G_0^*$ with $w = a_1 \ldots a_m$ for some $a_1, \ldots, a_m \in G_0$. For any partition $w = w_1 \ldots w_n$ $(n \geqq 0)$ of w the notation $f(w_1, \ldots, w_n)$ stands for the tree $f(a_1, \ldots, a_m) \in T_{F \cup G}$.

Finally we restate the main results of [3] and [6].

Denote the set $\{\mathscr{D}\mathscr{R}, \mathscr{N}\mathscr{D}\mathscr{R}, \mathscr{L}\mathscr{D}\mathscr{R}, \mathscr{L}\mathscr{N}\mathscr{D}\mathscr{R}, \mathscr{H}, \mathscr{N}\mathscr{H}, \mathscr{L}\mathscr{H}\}$ of tree transformation classes by S. The set of all tree transformation classes generated by S with composition $\circ$ is $[S]=\{\mathscr{K}_1\circ\ldots\circ\mathscr{K}_n | n\geqq 1,\ \mathscr{K}_i\in S \text{ for } 1\leqq i\leqq n\}$.

Let us introduce, for each integer $k\geqq 0$, the class $\mathscr{C}_k$ of tree transformations as follows:

(a) $\mathscr{C}_0=\mathscr{L}\mathscr{N}\mathscr{D}\mathscr{R}$,

(b) $\mathscr{C}_{k+1}=\mathscr{C}_k\circ\mathscr{N}\mathscr{H}$ if k is even and $\mathscr{C}_{k+1}=\mathscr{C}_k\circ\mathscr{L}\mathscr{N}\mathscr{D}\mathscr{R}$ if k is odd.

Moreover, consider the two finite subsets S_1 and S_2 of $[S]$ defined by

$$S_1 = S\cup\{\mathscr{D}\mathscr{R}^2, \mathscr{L}\mathscr{D}\mathscr{R}\circ\mathscr{N}\mathscr{H}, \mathscr{L}\mathscr{D}\mathscr{R}^2, \mathscr{L}\mathscr{D}\mathscr{R}\circ\mathscr{N}\mathscr{D}\mathscr{R}, \mathscr{H}\circ\mathscr{N}\mathscr{D}\mathscr{R}, \mathscr{L}\mathscr{D}\mathscr{R}^2\circ\mathscr{N}\mathscr{D}\mathscr{R}, \mathscr{L}\mathscr{N}\mathscr{D}\mathscr{R}\circ\mathscr{H}\}$$

and

$$S_2 = \{\mathscr{H}, \mathscr{N}\mathscr{H}, \mathscr{L}\mathscr{H}, \mathscr{L}\mathscr{D}\mathscr{R}\circ\mathscr{N}\mathscr{H}, \mathscr{L}\mathscr{N}\mathscr{D}\mathscr{R}\circ\mathscr{H}\}.$$

Proposition 2.1. (Theorem 12 of [3].) For each $\mathscr{C}\in[S]$ one of the following three assertions holds:

(i) $\mathscr{C}\in S_1$,

(ii) $\mathscr{C}=\mathscr{C}_k$ for some $k\geqq 0$,

(iii) $\mathscr{C}=\mathscr{C}'\circ\mathscr{C}_k$ for some $\mathscr{C}'\in S_2$ and $k\geqq 0$.

By this proposition, $[S]$ is infinite if and only if the hierarchy $\{\mathscr{C}_k\}$ is proper. Then, in [6] we obtained the following result.

Proposition 2.2. (Theorem 3 of [6].) $\{\mathscr{C}_{2k+1} | k=0, 1, \ldots\}$ is a proper hierarchy.

Notice that it follows from Proposition 2.2 that $\{\mathscr{C}_k\}$ is also a proper hierarchy. This can easily be seen by using the identities $\mathscr{L}\mathscr{N}\mathscr{D}\mathscr{R}\circ\mathscr{L}\mathscr{N}\mathscr{D}\mathscr{R}=\mathscr{L}\mathscr{N}\mathscr{D}\mathscr{R}$ and $\mathscr{N}\mathscr{H}\circ\mathscr{N}\mathscr{H}=\mathscr{N}\mathscr{H}$.

3. The results

First we examine the set of surface set classes $[S](\mathscr{R}ec)=\{\mathscr{C}(\mathscr{R}ec) | \mathscr{C}\in[S]\}$. We have the following result.

Theorem 3.1. The poset $\langle[S](\mathscr{R}ec), \subseteqq\rangle$ is a chain which can be written in the following form:

$$\mathscr{R}ec\subseteqq\mathscr{N}\mathscr{H}(\mathscr{R}ec)\subseteqq\mathscr{N}\mathscr{H}\circ\mathscr{C}_0(\mathscr{R}ec)\subseteqq\mathscr{N}\mathscr{H}\circ\mathscr{C}_1(\mathscr{R}ec)\ldots\subseteqq\mathscr{N}\mathscr{D}\mathscr{R}(\mathscr{R}ec)\subseteqq\mathscr{D}\mathscr{R}(\mathscr{R}ec).$$

Proof. By Proposition 2.1, we have $[S](\mathscr{R}ec)=\{\mathscr{C}(\mathscr{R}ec) | \mathscr{C}\in S_1\}\cup\{\mathscr{C}_k(\mathscr{R}ec) | k\geqq 0\}\cup\{\mathscr{C}'\circ\mathscr{C}_k(\mathscr{R}ec) | \mathscr{C}'\in S_2 \text{ and } k\geqq 0\}$. Then, using the results $\mathscr{D}\mathscr{R}^2(\mathscr{R}ec)=\mathscr{D}\mathscr{R}(\mathscr{R}ec)$ (Theorem I. 3. in [5]) and $\mathscr{L}\mathscr{D}\mathscr{R}(\mathscr{R}ec)=\mathscr{L}\mathscr{N}\mathscr{D}\mathscr{R}(\mathscr{R}ec)=\mathscr{L}\mathscr{H}(\mathscr{R}ec)=\mathscr{R}ec$ (Corollary IV.6.6. in [4]) as well as the identities $\mathscr{L}\mathscr{H}\circ\mathscr{N}\mathscr{H}=\mathscr{H}$ and $\mathscr{N}\mathscr{H}\circ\mathscr{N}\mathscr{D}\mathscr{R}=\mathscr{N}\mathscr{D}\mathscr{R}$ ([3]) we can write

$$\{\mathscr{C}(\mathscr{R}ec) | \mathscr{C}\in S_1\} = \{\mathscr{R}ec, \mathscr{N}\mathscr{H}(\mathscr{R}ec), \mathscr{N}\mathscr{D}\mathscr{R}(\mathscr{R}ec), \mathscr{D}\mathscr{R}(\mathscr{R}ec)\},$$

$$\{\mathscr{C}_k(\mathscr{R}ec) | k \geqq 0\} = \{\mathscr{R}ec, \mathscr{N}\mathscr{H}(\mathscr{R}ec), \mathscr{N}\mathscr{H}\circ\mathscr{C}_0(\mathscr{R}ec), \mathscr{N}\mathscr{H}\circ\mathscr{C}_1(\mathscr{R}ec), \ldots\}$$

and

$$\{\mathscr{C}'\circ\mathscr{C}_k(\mathscr{R}ec) | \mathscr{C}'\in S_2 \text{ and } k \geqq 0\} = \{\mathscr{N}\mathscr{H}(\mathscr{R}ec), \mathscr{N}\mathscr{H}\circ\mathscr{C}_0(\mathscr{R}ec), \mathscr{N}\mathscr{H}\circ\mathscr{C}_1(\mathscr{R}ec), \ldots\}$$

obtaining all the elements of $[S](\mathscr{R}ec)$. For proving the inclusions stated in our theorem we only have to observe that, since $\mathscr{N}\mathscr{D}\mathscr{R}$ is closed under composition,

$\mathscr{C}_k \subseteq \mathscr{NDR}$ and thus $\mathscr{NH} \circ \mathscr{C}_k \subseteq \mathscr{NDR}$ for each $k \geqq 0$. All the other inclusions follow by definition. □

We can raise the question that which of the inclusion relations appearing in Theorem 3.1 are proper. It is a folkloric result that $\mathscr{Rec} \subset \mathscr{NH}(\mathscr{Rec})$, moreover, it is also not difficult to see that $\mathscr{NH}(\mathscr{Rec}) \subset \mathscr{NH} \circ \mathscr{C}_0(\mathscr{Rec})$ which, in our paper, will be a consequence of Theorem 3.6. The questions that whether the hierarchy $\{\mathscr{NH} \circ \mathscr{C}_k(\mathscr{Rec})\}$ of classes of surface sets is proper or not and that whether $\bigcup_{k=0}^{\infty} \mathscr{NH} \circ \mathscr{C}_k(\mathscr{Rec}) \subset \mathscr{NDR}(\mathscr{Rec})$ are much more interesting and, at the same time, difficult. These problems are still open. However, we obtained the following result:

Lemma 3.2. $\mathscr{NDR}(\mathscr{Rec}) \subset \mathscr{DR}(\mathscr{Rec})$.

Proof. We observe that, by Theorem 3.2.1 of [2], $\text{ran}(\mathscr{DR}) = \mathscr{DR}(\mathscr{Rec})$ and $\text{ran}(\mathscr{NDR}) = \mathscr{NDR}(\mathscr{Rec})$. Therefore it is sufficient to give a forest in $\text{ran}(\mathscr{DR})$ which is not in $\text{ran}(\mathscr{NDR})$.

Let us introduce the ranked alphabet $F = F_0 \cup F_1 \cup F_2$ where $F_0 = \{\#\}$, $F_1 = \{f_1, f_2\}$ and $F_2 = \{g\}$. Denote the balanced tree of type $\{g, \#\}$ with height n by t'_n. Then construct the tree t_n from t'_n in the following manner: for each $w \in \text{path}(t'_n)$ with $|w| = n$ substitute the tree $f_{i_1}(\ldots f_{i_n}(\#)\ldots)$ for $\text{str}(t'_n, w)$ in t'_n where $w = i_1 \ldots i_n$. (We know that, for such a w, $\text{str}(t'_n, w) = \#$ and that $1 \leqq i_1, \ldots, i_n \leqq 2$.) An example for the case $n = 2$ of this construction can be seen in Fig. 1.

With this we achieved that each subtree of t_n with root g has exactly one occurrence in t_n.

Next we take a function symbol f with arity 2 and two function symbols e and h with arity 1. Let $G = F \cup \{e\}$ and $H = F \cup \{f, h\}$.

There exists a DR transducer $\mathfrak{A}$ such that $\tau_{\mathfrak{A}} = \{(e(p), f(p, h^n(\#)))\,|\, p \in T_F$ and $n = |\text{llp}(p)|\}$, where $h^n(\#) = \#$ if $n = 0$ and $h^n(\#) = h(h^{n-1}(\#))$ if $n \geqq 1$. (Notice that $\tau_{\mathfrak{A}} \subseteq T_G \times T_H$, moreover that $|\text{llp}(q)| = |\text{lrp}(q)|$ holds whenever $q \in \text{ran}(\tau_{\mathfrak{A}})$.)

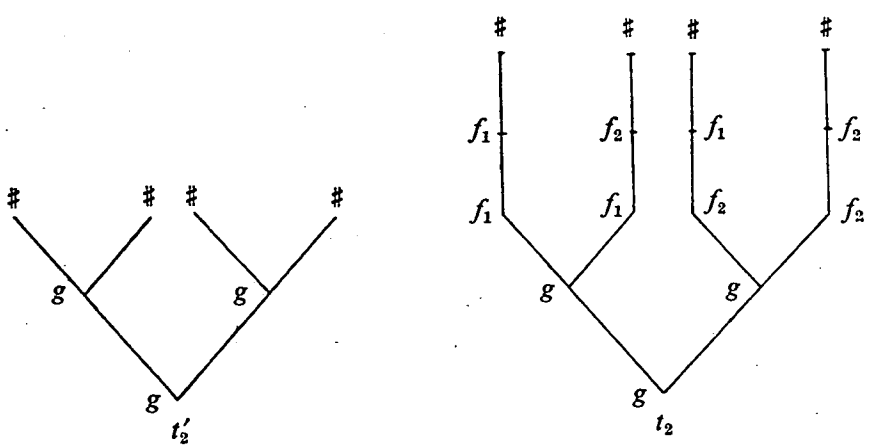


Figure 1.

In fact, the DR transducer the rules of which are listed below can be taken as $\mathfrak{A}$. The initial state is a.

$$ae(x_1) \to f(bx_1, cx_1),$$

$$bg(x_1, x_2) \to g(bx_1, bx_2),$$

$$bf_i(x_1) \to f_i(bx_1), \quad i = 1, 2, \quad b\# \to \#,$$

$$cg(x_1, x_2) \to h(cx_1), cf_i(x_1) \to h(cx_1), \quad i = 1, 2, \quad c\# \to \#.$$

We show that ran $(\tau_{\mathfrak{A}}) \notin$ ran $(\mathcal{NDR})$. For this, let us introduce first the abbreviation $q_n = g(t_n, h^{2n}(\#))$, for $n \geqq 1$. Then, since $\tau_{\mathfrak{A}}$ sends $e(t_n)$ to q_n we have that $\{q_n | n = 1, 2, \ldots\} \subseteq$ ran $(\tau_{\mathfrak{A}})$.

Now suppose indirectly that there exists an NDR transducer $\mathfrak{B} = (E, B, H, P, b_0)$ such that ran $(\tau_{\mathfrak{A}}) =$ ran $(\tau_{\mathfrak{B}})$. Then also $\{q_n | n = 1, 2, \ldots\} \subseteq$ ran $(\tau_{\mathfrak{B}})$ therefore, for each $n = 1, 2, \ldots$ there exists a $p'_n \in T_E$ so that $b_0 p'_n \overset{*}{\underset{\mathfrak{B}}{\Rightarrow}} q_n$. We note that some of these derivations may start with such a sequence of rules in which the height of the right-hand side of each rule is 0. But, after dropping this sequence of rules from each derivation we have that for each $n = 1, 2, \ldots$ there exists a $b_n \in B$ and a $p_n \in \mathrm{sub}(p'_n)$ with $b_n p_n \overset{*}{\underset{\mathfrak{B}}{\Rightarrow}} q_n$ such that each derivation starts with a rule, the height of the right-hand side of which is greater than 0. Then we can choose an infinite subsequence $n_1, n_2, \ldots, n_k, \ldots$ of $1, 2, \ldots, n, \ldots$ such that the same rule, let us say $b\sigma(x_1, \ldots, x_u) \to q(b_1 x_{i_1}, \ldots, b_v x_{i_v})$ is applied in the first step of the derivations $b_{n_k} p_{n_k} \overset{*}{\underset{\mathfrak{B}}{\Rightarrow}} q_{n_k}$ for $k = 1, 2, \ldots$. (This, of course, entails that $b = b_{n_k}$ for each $k = 1, 2, \ldots$.) Moreover, without loss of generality, we may suppose that $q \in \hat{T}_{H,v}$ and $\mathrm{fr}(q) = x_1, \ldots x_v$. (For notations, see [3] or [6].)

We observe that the longest leftmost path (resp. longest rightmost path) of q ends in x_1 (resp. x_v) or, formally, $\mathrm{str}(\mathrm{llp}(q), q) = x_1$ (resp. $\mathrm{str}(\mathrm{lrp}(q), q) = x_v$). For, if this were not the case then $|\mathrm{llp}(q_{n_k})|$ (resp. $|\mathrm{lrp}(q_{n_k})|$) would be a constant for each $k = 1, 2, \ldots$.

Next we show that $x_{i_1} = x_{i_v}$ or, equivalently, $i_1 = i_v$. On the contrary, assume that $i_1 < i_v$. Choose two integers k and l such that $k < l$ and write the derivations $bp_{n_k} \overset{*}{\underset{\mathfrak{B}}{\Rightarrow}} q_{n_k}$ and $bp_{n_l} \overset{*}{\underset{\mathfrak{B}}{\Rightarrow}} q_{n_l}$ in more detailed form as

$$\begin{aligned} bp_{n_k} &= b\sigma(p_1^{(k)}, \ldots, p_{i_1}^{(k)}, \ldots, p_{i_v}^{(k)}, \ldots, p_u^{(k)}) \overset{*}{\underset{\mathfrak{B}}{\Rightarrow}} \\ & q(b_1 p_{i_1}^{(k)}, \ldots, b_v p_{i_v}^{(k)}) \overset{*}{\underset{\mathfrak{B}}{\Rightarrow}} q(q_1^{(k)}, \ldots, q_v^{(k)}) = q_{n_k} \end{aligned} \tag{1}$$

and similarly

$$\begin{aligned} bp_{n_l} &= b\sigma(p_1^{(l)}, \ldots, p_{i_1}^{(l)}, \ldots, p_{i_v}^{(l)}, \ldots, p_u^{(l)}) \overset{*}{\underset{\mathfrak{B}}{\Rightarrow}} \\ & q(b_1 p_{i_1}^{(l)}, \ldots, b_v p_{i_v}^{(l)}) \overset{*}{\underset{\mathfrak{B}}{\Rightarrow}} q(q_1^{(l)}, \ldots, q_v^{(l)}) = q_{n_l}. \end{aligned}$$

These two derivations entail that

$$b\sigma(p_1^{(k)}, \ldots, p_{i_1}^{(k)}, \ldots, p_{i_v}^{(l)}, \ldots, p_u^{(k)}) \overset{*}{\underset{\mathfrak{B}}{\Rightarrow}} q(q_1^{(k)}, \ldots, q_v^{(l)})$$

from where we see that $q(q_1^{(k)}, \ldots, q_v^{(l)}) \in$ ran $(\tau_{\mathfrak{B}})$ and thus, by ran $(\tau_{\mathfrak{A}}) =$ ran $(\tau_{\mathfrak{B}})$,

$q(q_1^{(k)}, \ldots, q_v^{(l)}) \in \mathrm{ran}(\tau_{\mathfrak{A}})$. Then, by the note we made after the definition of $\tau_{\mathfrak{A}}$, $|\mathrm{llp}(q(q_1^{(k)}, \ldots, q_v^{(l)}))| = |\mathrm{lrp}(q(q_1^{(k)}, \ldots, q_v^{(l)}))|$. On the other hand

$$|\mathrm{llp}(q(q_1^{(k)}, \ldots, q_v^{(l)}))| = |\mathrm{llp}(q)| + |\mathrm{llp}(q_1^{(k)})| = |\mathrm{llp}(q_{n_k})| = 2n_k + 1 \quad \text{and}$$

$$|\mathrm{lrp}(q(q_1^{(k)}, \ldots, q_v^{(l)}))| = |\mathrm{lrp}(q)| + |\mathrm{lrp}(q_v^{(l)})| = |\mathrm{lrp}(q_{n_l})| = 2n_l + 1, \quad \text{that is,} \quad n_k = n_l.$$

This is a contradiction, since $k < l$.

Let us suppose that $i_1 = i_v = 1$.

Denote the number of states in B by $|B|$ and let $K = \max\{h(q) | q$ is the right-hand side of some rule in $P\}$. Let the integer k be chosen and fixed such that $n_k > K(|B|+1)$.

Consider, from (1), the derivation $b_v p_1^{(k)} \overset{*}{\underset{\mathfrak{B}}{\Rightarrow}} q_v^{(k)}$. Since $\mathrm{lrp}(q)$ ends in x_v, by the definition of q_{n_k}, $q_v^{(k)}$ contains only the function symbols h and $\sharp$ of H. But then, since $\mathfrak{B}$ is an NDR transducer and the arity of h is 1, the arity of the function symbols occuring in $p_1^{(k)}$ is either 1 or 0.

Consider now the derivation $b_1 p_1^{(k)} \overset{*}{\underset{\mathfrak{B}}{\Rightarrow}} q_1^{(k)}$. We state three properties of $q_1^{(k)}$. Namely, by the choice of k, we have

(P1) $h(q_1^{(k)}) \geqq 2n_k + 1 - K > 2 \cdot |B| \cdot K$

moreover, by the position of $q_1^{(k)}$ in q_{n_k},

(P2) if $w \in \mathrm{path}(q_1^{(k)})$ is such that $\mathrm{lab}(q_1^{(k)}, w)$ is f_1, f_2 or $\sharp$ then $|w| > |B| \cdot K$ and, since $q_1^{(k)}$ is a subtree of t_{n_k},

(P3) each subtree of $q_1^{(k)}$ with root g has exactly one occurrence in $q_1^{(k)}$.

Further on, we analyse the derivation $b_1 p_1^{(k)} \overset{*}{\underset{\mathfrak{B}}{\Rightarrow}} q_1^{(k)}$. Therefore, consider the following algorithm.

let $i=0$, $r_0 = x_1$, $b_1^{(0)} = b_1$, $s_0 = p_1^{(k)}$, $m_0 = 1$;
while $r_i \neq q_1^{(k)}$ **do**
begin
search for the smallest integer j for which $r_i(b_1^{(i)} s_i, \ldots, b_{m_i}^{(i)} s_i) \overset{j}{\underset{\mathfrak{B}}{\Rightarrow}} r(b_1' s, \ldots, b_m' s)$ holds for some $m \geqq 0$, $r \in \hat{T}_{H,m}$, $s \in T_E$ and $b_1', \ldots, b_m' \in B$ such that $\mathrm{rn}(r_i) < \mathrm{rn}(r)$;
let $i = i+1$;
let $r_i = r$, $s_i = s$, $m_i = m$, $j_i = j$
and $b_l^{(i)} = b_l'$ for $1 \leqq l \leqq m$
end

(Here $\overset{j}{\underset{\mathfrak{B}}{\Rightarrow}}$ stands for the j-fold composition of the relation $\underset{\mathfrak{B}}{\Rightarrow}$.)

We note that the smallest integer j in the above algorithm can be found by rewriting simultaneously the subtrees $b_1^{(i)} s_i, \ldots, b_{m_i}^{(i)} s_i$. (This simultaneous rewriting was called parallel derivation in [2].)

Since each derivation of $\mathfrak{B}$ starting from a state and an input tree terminates after a finite number of steps our algorithm also terminates after, let us say, N steps. Moreover, since $b_1 p_1^{(k)} \overset{*}{\underset{\mathfrak{B}}{\Rightarrow}} q_1^{(k)}$, it holds that $m_N = 0$ and $r_N = q_1^{(k)}$. Thus we can write

$$r_0(b_1^{(0)} s_0) \overset{j_1}{\underset{\mathfrak{B}}{\Rightarrow}} r_1(b_1^{(1)} s_1, \ldots, b_{m_1}^{(1)} s_1) \overset{j_2}{\underset{\mathfrak{B}}{\Rightarrow}} \ldots \overset{j_N}{\underset{\mathfrak{B}}{\Rightarrow}} r_N(b_1^{(N)} s_N, \ldots, b_{m_N}^{(N)} s_N) = q_1^{(k)}.$$

We make the following observations.

Since we choose the smallest integer j in the while loop it holds that $h(r_i) \leqq \leqq i \cdot K$, for $1 \leqq i \leqq N$, therefore, by property (P1) of $q_1^{(k)}$, we have that $N > 2 \cdot |B|$.

Let $i = |B|$. Then, by property (P2) of $q_1^{(k)}$ we obtain that each tree of $r_1, \ldots, r_i$ contains only the function symbol g of H. Thus the condition $\mathrm{rn}(r_0) < \mathrm{rn}(r_1) < \ldots < \mathrm{rn}(r_i)$ entails that $2 \leqq m_1 < \ldots < m_i$, hence, we get that $m_i > |B|$.

Then, for $i = |B|$, there is at least one state that appears at least twice in the sequence $b_1^{(i)}, \ldots, b_{m_i}^{(i)}$.

Since $r_i(b_1^{(i)} s_i, \ldots, b_{m_i}^{(i)} s_i) \overset{*}{\underset{\mathfrak{B}}{\Rightarrow}} q_1^{(k)}$ we obtain, by (P2) and $h(r_i) \leqq i \cdot K = B \cdot K$, that there is a subtree with root g of $q_1^{(k)}$ which appears at least twice in $q_1^{(k)}$. However, this contradicts property (P3) of $q_1^{(k)}$. With this we finished the proof of our lemma. □

We note that in the above proof we strongly used the fact that the output ranked alphabet H of our counter-example $\tau_{\mathfrak{A}}$ contains function symbols of arity 1. It is not clear how this lemma could be proved if we restricted ourselves to ranked alphabets that do not contain 1-ary function symbols.

Now we begin to deal with the poset $\langle \mathrm{yd}([S](\mathscr{R}ec)), \subseteq \rangle$ where $\mathrm{yd}([S](\mathscr{R}ec)) = \{\mathrm{yd}(\mathscr{T}) | \mathscr{T} \in [S](\mathscr{R}ec)\}$. We observe that, since $\langle [S](\mathscr{R}ec), \subseteq \rangle$ is a chain and yd preserves inclusion, $\langle \mathrm{yd}([S](\mathscr{R}ec)), \subseteq \rangle$ is also a chain. First we prove a technical lemma.

Lemma 3.3. $\mathscr{NDR}_{\mathrm{tts}} = \mathscr{DR}_{\mathrm{tts}}$.

Proof. It is sufficent to show that $\mathscr{DR}_{\mathrm{tts}} \subseteq \mathscr{NDR}_{\mathrm{tts}}$. To this end take a DR transducer $\mathfrak{A} = (F, A, G, P, a_0)$ and denote the number of rules in P by $|P|$. Suppose that the rules in P are numbered from 1 to $|P|$.

The following algorithm produces, for each $i = 1, \ldots, |P|$, a function symbol f_i and a rule ϱ_i for a DR transducer:

(a) Suppose that the i-th rule is of the form $af(x_1, \ldots, x_m) \to q$ where $q \in T_G(A \times X_m)$.

(b) Let $\mathrm{yd}(q) = w_0(a_1, x_{i_1}) w_1 \ldots (a_n, x_{i_n}) w_n$ where $n \geqq 0$, $1 \leqq x_{i_1}, \ldots, x_{i_n} \leqq m$, $w_0, w_1, \ldots, w_n \in G_0^*$.

(c) Let $\{x_{j_1}, \ldots, x_{j_k}\} \subseteq X_m$ be the set of all variables which do not occur in q (and so neither in $\mathrm{yd}(q)$).

(d) Let f_i be a new function symbol with arity $|w_0| + \ldots + |w_n| + n + k$.

(e) Let ϱ_i be the rule
$af(x_1, \ldots, x_m) \to f_i(w_0, a_1 x_{i_1}, \ldots, a_n x_{i_n}, w_n, cx_{j_1}, \ldots, cx_{j_k})$ where $c \notin A$ is a new state. (As usual, (a_k, x_{i_k}) is abbreviated by $a_k x_{i_k}$, for $1 \leqq k \leqq n$.)

Now we introduce the DR transducer $\mathfrak{B} = (F, A \cup \{c\}, F', P', a_0)$ where

$$F' = \{f_i | i = 1, \ldots, |P|\} \cup F \cup \{\varepsilon\} \quad \text{and}$$

$$P' = \{\varrho_i | i = 1, \ldots, |P|\} \cup \{cf(x_1, \ldots, x_m) \to$$

$$\to f(cx_1, \ldots, cx_m) | m \geqq 1, \quad f \in F_m\} \cup \{cf \to \varepsilon | f \in F_0\}.$$

It can be seen from the construction that $\mathfrak{B}$ is an NDR transducer. Moreover, it can be verified by an induction on p that for each $a \in A$, $p \in T_F$ and $w \in G_0^*$,

$$(\exists q \in T_G)(ap \overset{*}{\underset{\mathfrak{A}}{\Rightarrow}} q \wedge \mathrm{yd}(q) = w) \Leftrightarrow (\exists q' \in T_{F'})(ap \overset{*}{\underset{\mathfrak{B}}{\Rightarrow}} q' \wedge \mathrm{yd}(q') = w).$$

It then follows that $\tau_{\mathfrak{A}_{tts}}=\tau_{\mathfrak{B}_{tts}}$. Hence we have $\mathcal{DR}_{tts}\subseteq\mathcal{NDR}_{tts}$. □

Consequence 3.4. $\mathcal{LNDR}_{tts}=\mathcal{LDR}_{tts}$.

Proof. If $\mathfrak{A}$ in Lemma 3.3 is an LDR transducer then $\mathfrak{B}$ is an LNDR transducer. □

Consequence 3.5. $\mathcal{DR}_{tts}=(\mathcal{NH}\circ\mathcal{LNDR})_{tts}$.

Proof. It is well known that $\mathcal{DR}=\mathcal{NH}\circ\mathcal{LDR}$ (c.f. [1], [4]) thus we have $\mathcal{DR}_{tts}=(\mathcal{NH}\circ\mathcal{LDR})_{tts}=\mathcal{NH}\circ\mathcal{LDR}_{tts}=\mathcal{NH}\circ\mathcal{LNDR}_{tts}=(\mathcal{NH}\circ\mathcal{LNDR})_{tts}$. □

Now we are ready to state our last theorem.

Theorem 3.6. The poset $\langle \mathrm{yd}([S](\mathcal{R}ec)), \subseteq\rangle$ is a chain of three elements

$$\mathrm{yd}(\mathcal{R}ec)\subset\mathrm{yd}(\mathcal{NH}(\mathcal{R}ec))\subset\mathrm{yd}(\mathcal{DR}(\mathcal{R}ec)).$$

Proof. By Consequence 3.5, we can compute as follows:

$\mathrm{yd}(\mathcal{NH}\circ\mathcal{C}_0(\mathcal{R}ec))=\mathrm{yd}(\mathcal{NH}\circ\mathcal{LNDR}(\mathcal{R}ec))=(\mathcal{NH}\circ\mathcal{LNDR})_{tts}(\mathcal{R}ec)=\mathcal{DR}_{tts}(\mathcal{R}ec)=$ $=\mathrm{yd}(\mathcal{DR}(\mathcal{R}ec))$. Thus applying yd to each element of $\langle[S](\mathcal{R}ec), \subseteq\rangle$ we obtain the chain $\mathrm{yd}(\mathcal{R}ec)\subseteq\mathrm{yd}(\mathcal{NH}(\mathcal{R}ec))\subseteq\mathrm{yd}(\mathcal{DR}(\mathcal{R}ec))$. Here each inclusion is proper as it was shown in [2]. □

Finally we have the consequence mentioned before.

Consequence 3.7. $\mathcal{NH}(\mathcal{R}ec)\subset\mathcal{NH}\circ\mathcal{C}_0(\mathcal{R}ec)$.

Proof. It is obvious since, by the proof of Theorem 3.6, the same proper inclusion holds for the yields of these two classes. □

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

References

[1] ENGELFRIET, J., Bottom-up and top-down tree transformations — A comparison, Math. Syst. Theory, 9, 198—231, 1975.
[2] ENGELFRIET, J., G. ROZENBERG and G. SLUTZKI, Tree transducers, L systems and two-way machines, Journal of Comp. and Syst. Sciences, 20, 150—202, 1980.
[3] FÜLÖP Z. and VÁGVÖLGYI S., Results on compositions of deterministic root-to-frontier tree transformations, Acta Cybernetica, 8, 49—62, 1987.
[4] GÉCSEG F. and M. STEINBY, Tree Automata, Akadémia Kiadó, Budapest, 1984.
[5] ROUNDS, W. C., Mappings and grammars on trees, Math. Syst. Theory, 4, 257—287, 1970.
[6] VÁGVÖLGYI S. and FÜLÖP Z., An infinite hierarchy of tree transformations in the class $\mathcal{NDR}$, Acta Cybernetica, 8, 153—168, 1987.

(Received June 19, 1987)

On the numbers of shortest keys in relational databases on nonuniform domains

O. SELESNJEW, B. THALHEIM

The use of the relational model of data structures by E. F. Codd is a promising mathematical tool for handling data. In this model the user's data are expressed as relations where the rows denote the records and the columns represent domains or attributes. For the handling of relations the identification of sets of domains, called keys, is suggested. The keys uniquely determine the values of the rest of the domains. Delobel and Casey, Fadous and Forsyth, M. Fernandez, C. L. Lucchesi and S. L. Osborn, J. Demetrovics and V. Thi have given different algorithms for finding the set of all minimal keys in a relational database given by a set of functional dependencies on the database. For characterizing the complexity of this algorithms we need some combinatorial bounds.

In this paper we consider the maximal numbers of shortest keys in relational databases on weighted domains and extend the result of J. Demetrovics who solved the problem for relational databases on uniform domains. [1]

1. Basic notions

We recall briefly some definitions of the theory of relational databases. Given sets $D_1, D_2, \ldots, D_n$, called domains, not necessarily distinct, an n-ary relation R defined over $D_1, \ldots, D_n$ is a subset of the cartesian product $D_1 \times D_2 \times D_n$.

An attribute is a name assigned to a domain of a relation. Any value associated with an attribute is called attribute value. The attribute names must be distinct. The symbol U will be used to denote the set of all n attributes of R.

A set of attributes X, $X \subseteq U$, is called a *key* of R if, for every n-tuple of R, the values of the attributes in X uniquely determine the values of the attributes in U.

Now, suppose we are given some weight function (or complexity measure) $g: U \to \mathbf{N}'$ and a system S_R of keys of R. For $X \subseteq U$ let $g(X) = \sum_{A \in X} g(A)$ the complexity of X. An element K of S_R is called *g-shortest* if there does not exist an element K' of S_R with $g(K') < g(K)$. By $S_R(g)$ we denote the set of all g-shortest elements of S_R and by $s_R(g)$ its cardinality. For $g=1$ the set $S_R(g)$ is called the set of all shortest keys or the set of shortest keys in an unweighted database. It is obvious that

any set $S_R(g)$ is a subset of a set of minimal keys [1]. For any set S of minimal keys there exists a subset $S(g)$ of shortest keys. This is well-known for $g=1$.

Theorem 1. [1] The maximal size of a set of shortest keys in a database with n attributes is $\binom{n}{\left[\frac{n}{2}\right]}$. For any k, $1 \leqq k \leqq \binom{n}{\left[\frac{n}{2}\right]}$, there is an n-ary relation R with k shortest keys.

2. Maximal number of shortest keys in nonuniform databases

In practical cases, keys are of different meaning and complexity. Domains for attributes have very different complexity. This is well-known in practice but it is not taken into consideration in the theory of minimal keys. Therefore, shortest keys are introduced.

Lower and upper bounds for $s_R(g)$ are proved in [4]. The most interesting set of functions g is the set G^+ of functions g with $g(A_i) \neq g(A_j)$ for $i \neq j$. The other cases can be splitted in the case $g(A)=1$ for $A \in X$ and in this case for $A \in U \setminus X$. We introduce the following functions:

$s(g) = \max\limits_{R} s_R(g)$,

$s(G') = \max\limits_{g \in G'} s(g)$ for sets G' of weight functions.

Using the functions g_1, g_2, g_3 with $g_1(A_i)=2^i$, $g_2(A_i)=3^{[i/2]}$, $g_3(A_i)=i$, for i, $1 \leqq i \leqq n$, by the definitions and the recursion formulas for g_3, we get

Corollary 2. 1. For weight functions g it holds $1 \leqq s(g) \leqq \binom{n}{\left[\frac{n}{2}\right]}$ [1].

2. $s(g_1)=1$, $s(g_2)=2^{[n/2]}$, $s(g_3) \geqq \dfrac{2^n}{n^2}$ [4].

Our next aim is to prove

Theorem 3. $s(G^+) = \dfrac{2^n}{\sqrt{\frac{\pi}{6} n^{3/2}}} (1-o(1))$.

From number theory [2] we get that functions g with $s(g)=s(G^+)$ must be regulary. W.l.o.g. we consider a subclass G^* of G^+, the class of all equidistant functions g with the property $g(A_i)-g(A_{i-1})=c$ for some c and any i, $2 \leqq i \leqq n$.

Lemma 4. 1. Given two equidistant functions g, g' from G^+. Then $s(g)=s(g')$.

2. Let g be a function from G^+. There exists an equidistant function g' in G^* such that $s(g)=s(g')$.

Proof. 1. Is obvious.

2. W.l.o.g. we consider only functions g from G^+ with $g(A_i)<g(A_{i+1})$ for i, $1 \leqq i < n$. We prove the assertion by induction. For $n=2$ the assertion is obvious. Let n be a fixed number. Now we assume that for a fixed function g there is no equi-

distant function $g' \in G^*$ such that $s(g) \leqq s(g')$. Let be S_R a key system with $s(g) = s_R(g)$. Now we define $S_1 = \{K \in S_R / A_n \notin K\}$, $S_2 = \{K \setminus \{A_n\} / K \in S_R, A_n \in K\}$. By precondition of induction, we get for $g' = g|_{U \setminus \{A_n\}}$ an equidistant function g'' such that $s(g') \leqq s(g'')$. It follows that there is an equidistant function g^+ in G^* such that $g^+|_{U \setminus \{A_n\}} = g''$ and $s(g) \leqq s(g^+)$. That is a contradiction.

W.l.o.g. we can consider for $s(G^+)$ the function g_3 of Corollary 2. Now we define independent random variables ξ_k with two-point distribution for $k = 1, 2, \ldots, n$:

$$\xi_k = \begin{cases} k \\ 0 \end{cases}$$

and consider the distribution of $S_n = \sum_{i=1}^{n} \xi_i$.

Corollary 5. $P\left(S_n = \left[\frac{n(n+1)}{4}\right]\right) \cong \frac{1}{2^n} s(g_3)$ for probability $P(S_n = m)$.

For the expectation ES_n and the variance DS_n of S_n we get

$$M_n = ES_n = \sum_{k=1}^{n} E\xi_k = \sum_{k=1}^{n} \frac{k}{2} = \frac{n(n+1)}{4}$$

$$B_n^2 = DS_n = \sum_{k=1}^{n} D\xi_k = \frac{n(n+1)(2n+1)}{24} \sim \frac{n^3}{12} \quad (n \to \infty).$$

We shall say that the sequence $\{S_n\}$ satisfies a local limit theorem iff

$$\sup_m |B_n P(S_n = m) - \varphi(x_{nm})| \to 0 \quad (n \to \infty)$$

where $x_{nm} = \frac{m - M_n}{B_n}$, $z_n = \frac{S_n - M_n}{B_n}$, φ is the standard normal distribution density. We denote

$$\alpha(a, q, N) = \frac{1}{q^2} \sum_{-q/2 < r \leqq q/2} r^2 P(a\tilde{\xi}_k = r \pmod q, \quad |\xi_k| \leqq N) \qquad (+)$$

for $\tilde{\xi}_k = \xi_k - \xi_k'$ symmetrized random variable, where ξ_k' is a random variable independent of ξ_k and having the same distribution as ξ_k, relatively prime integers a, q with $a \leqq \frac{q}{2}$ and $1 < q \leqq 2N$.

In [3] is proved the following: If the distribution function of the sum of unboundedly increasing number of random variables converges to the standard normal distribution function,

$$\text{i.e.} \quad z_n \xrightarrow{D} N(0, 1), \qquad (1)$$

$$N_n \exp\left\{-\frac{1}{2} \min_{a,q} \sum_{k=1}^{n} \alpha_k(a, q, N_k)\right\} \to 0 \quad (n \to \infty), \qquad (2)$$

$$N_n \text{ is selected such that } \lim_{n \to \infty} \frac{1}{B_n^2} \sum_{k=1}^{n} \int_{|x| \leqq N_n} x^2 \, dF_{\tilde{\xi}_k}(x) = l > 0, \qquad (3)$$

then the sum satisfies a local limit theorem.

Let $N_n=n$. Then we get

$$l = \lim_{n\to\infty} \frac{1}{B_n^2} \sum_{k=1}^{n} D\xi_k^2 = 1 > 0,$$

$$P(\tilde{\xi}_k = k) = P(\tilde{\xi}_k = -k) = \frac{1}{4}, \quad P(\tilde{\xi}_k = 0) = \frac{1}{2}.$$

By summation of (+) over representatives of q we get $|\tilde{\xi}_k| \leqq n$ for $k \in \{1, \ldots, n\}$. Observe that if $\tilde{\xi}_k = 0$ then $r=0$ and this summand can be eliminated and that if $\tilde{\xi}_k = k$ then $ak = r_k + ql_k$ for the unique representative of q. We get

$$\alpha_k(a, q, n) = \frac{1}{q^2} \sum_{-q/2 < r \leqq q/2} r^2 P(a\tilde{\xi}_k = r \pmod q) =$$

$$= \frac{1}{4q^2}(r_k^2 + r_{-k}^2) \geqq \frac{1}{4q^2} r_k^2.$$

From number theory it is known that if $\{x\}$ form a full system of representatives of q then $\{ax\}$ form a full system of representatives. Now $\lambda_n \geqq \min\limits_{a,q} \sum\limits_{k=1}^{n} \alpha_k(a, q, n) \geqq$ $\geqq \min\limits_{q} \frac{1}{4q^2} \sum\limits_{k=1}^{n} r_k^2$. Assume that $q=2m$. (For odd q proof is analogous.) Let $0 < \alpha < \frac{1}{2}$. If $\alpha_n \leqq m \leqq n$ then

$$\sum_{k=1}^{n} r_k^2 \geqq \sum_{k=1}^{m} k^2 \geqq cm^3 \geqq c\alpha^3 n^3$$

for the full system of representatives $r_k = -(m-1), \ldots, 0, 1, \ldots, m$ and therefore

$$\lambda_n \geqq \min_q \frac{1}{4q^2} c\alpha^3 n^3 \geqq \frac{c\alpha^3}{4\alpha^2 4} \frac{n^3}{n^2} = \beta n, \quad \beta > 0.$$

If $1 < m < \alpha n$ then the full system of representatives $\{r\}_{-(m-1)}^{m}$ is contained in $\{1, \ldots, n\}$ at least $\left[\frac{n}{q}\right]$ times. Consequently we get

$$\lambda_n \geqq \min_{a,q} \frac{1}{4q^2} \sum_{k=1}^{n} r_k^2 \geqq \min_{a,q} \frac{\left[\frac{n}{q}\right] \sum\limits_{k=1}^{m} k^2}{4q^2} \geqq$$

$$\geqq \min_{a,q} \left(\frac{n}{q} - 1\right) \frac{\sum\limits_{k=1}^{m} k^2}{4q^2} \geqq \min_{q} (n-q) \frac{\sum\limits_{k=1}^{m} k^2}{4q^3} \geqq \min_{q} (n - 2\alpha n)c =$$

$$= c(1-2\alpha)n = \beta n > 0, \quad \beta > 0.$$

We get that (2) holds because $\Delta_n = n \exp\left\{-\frac{1}{2}\lambda_n\right\} \leqq n \exp\left\{-\frac{\beta}{2}n\right\} \to 0 (n \to \infty)$. Summarizing corollary 5, lemma 4 and the properties of S_n we get

$$s(g_3) = 2^n P\left(S_n = \left[\frac{n(n+1)}{4}\right]\right) = \frac{2^n}{B_n} \varphi\left(x_{n\left[\frac{n(n+1)}{4}\right]}\right) \underset{n\to\infty}{\sim}$$

$$\underset{n\to\infty}{\sim} \frac{2^n}{\sqrt{2\pi \frac{n(n+1)(2n+1)}{24}}} = \frac{2^n}{\sqrt{\frac{\pi}{6}} n^{3/2} \left(1 + \frac{3}{2n} + \frac{1}{2n^2}\right)^{1/2}} \sim \frac{2^n}{\sqrt{\frac{\pi}{6}} n^3}.$$

The proof of theorem 3 is complete.

It is of interest to compare this result with $s(g_4) \sim \frac{2^n}{\sqrt{\frac{\pi}{2}} \sqrt{n}}$ for $g_4(A) = 1$ for $A \in U$.

Using a central limit theorem we get further

$$\left| s(G^+) - \frac{2^n}{\sqrt{\frac{\pi}{6} n^3 \left(1 + \frac{3}{2n} + \frac{1}{2n^2}\right)}} \right| \leqq \frac{c}{\sqrt{n}}$$

for some constant c.

O. SELESNJEW
MOSCOW STATE UNIVERSITY
DEPT. OF MATHEMATICS AND
MECHANICS
117 234 MOSCOW
USSR

B. THALHEIM
DRESDEN UNIVERSITY OF TECHNOLOGY
DEPT. OF MATHEMATICS
COMPUTER SCIENCE DIVISION
8027 DRESDEN
GDR

References

[1] J. DEMETROVICS, On the equivalence of candidate keys with Sperner systems. Acta Cybernetica 4 (1979), 247—252.
[2] K. KNOPP, I. SCHUR, Elementarer Beweis einiger asymptotischer Formeln der additiven Zahlentheorie. Mathematische Zeitschrift 24, 1925, 559—574.
[3] А. А. Миталаускас, В. Л. Статулявичус, Локальные предельные теоремы и асимптотические разломение для сумм независимых решетчатых случайных величин. Литовский математический сборник 1966, Т. 6, N. 4, 569—583.
[4] B. THALHEIM, Abhängigkeiten in Relationen. Dissertation B, Technische Universität Dresden, Dresden 1985.

(Received March 4, 1987)

Some results about functional dependencies*

J. DEMETROVICS and V. D. THI

Abstract

§ 1. Introduction

The relational datamodel was defined by E. F. Codd [2]. In this datamodel a relation is a table (matrix) in which each column corresponds to a distinct attribute and each row to a distinct record. Relations are used to describe connections among data items. The functional dependency is one of the main concepts in relational datamodel. The mathematical structure of functional dependencies was thouroughly investigated by W. W. Armstrong [1]. The equivalence of sets of minimal keys with Sperner-systems was proved [4]. It is known [1] that for a given family F of functional dependencies there is a relation representing F in the sense that the full family of functional dependencies of this relation is exactly F. Also it is shown [4] that for an arbitrarily given Sperner-system there exists a relarion R representing this Sperner-system so that this Sperner-system is exactly the set of all minimal keys of R. In this paper we give necessary and sufficient conditions for a relation to represent a given family of functional dependencies or a Sperner-system.

The closure operation is a useful and interesting instrument for investigating the structure of functional dependencies. In this paper we investigate the connection between closure operations and sets of minimal keys, too. Now we give some necessary definitions.

Let $\Omega=\{a_1, \ldots, a_n\}$ be a finite non-empty set of attributes. For each attribute a_i there is a non-empty set $D(a_i)$ of all possible values of that attribute. An arbitrary finite subset of the Cartesian product $D(a_1)\times\ldots\times D(a_n)$ is called a relation over Ω. It can be seen that a relation over Ω is a set of mappings $h: \Omega \to \bigcup_{a\in\Omega} D(a)$, where $h(a)\in D(a)$ for all a.

*This paper was supported by grants from the Hungarian Academy of Sciences OTKA Nr. 1066 and 1812.

The main purpose of this paper is to give necessary and sufficient conditions for a relation to represent an arbitrarily given family of functional dependencies or a closure operation or a Sperner-system. The connection between closure operations and sets of minimal keys is investigated too.

Definition 1.1. [2] Let $R=\{h_1, \ldots, h_m\}$ be a relation over the finite set of attributes Ω. Let $A, B\subseteq\Omega$. We say that B functionally depends on A in R (denoted as $A\xrightarrow[R]{}B$) iff $(\forall h_i, h_j\in R)((\forall a\in A)(h_i(a)=h_j(a))\rightarrow(\forall b\in B)(h_i(b)=h_j(b)))$.

Let $F_R=\{(A, B): A\xrightarrow[R]{}B\}$. F_R is called the full family of functional dependencies of R.

Definition 1.2. [1] Let Ω be a finite set, and denote $P(\Omega)$ its power set. Let $F\subseteq P(\Omega)\times P(\Omega)$. We say that F is an f-family over Ω iff for all $A, B, C, D\subseteq\Omega$

(F1) $(A, A)\in F$;
(F2) $(A, B)\in F, (B, C)\in F\rightarrow(A, C)\in F$;
(F3) $(A, B)\in F, A\subseteq C, D\subseteq B\rightarrow(C, D)\in F$;
(F4) $(A, B)\in F, (C, D)\in F\rightarrow(A\cup C, B\cup D)\in F$.

By [1], F_R is an f-family over Ω. It is known [1] that if F is an f-family, then there is a relation R over Ω such that $F_R=F$.

Definition 1.3. The mapping $L: P(\Omega)\rightarrow P(\Omega)$ is called a closure operation over Ω iff for every $A, B\subseteq\Omega$:

(1) $A\subseteq L(A)$;

(2) $A\subseteq B\rightarrow L(A)\subseteq L(B)$;

(3) $L(L(A))=L(A)$.

Remark 1.1. It is easy to see that if F is an f-family and for all $A\subseteq\Omega$, we set $L_F(A)=\{a\in\Omega: (A, \{a\})\in F\}$ then L_F is a closure operation over Ω. Conversely, it is shown [1] that if L is a closure operation over Ω, then there is exactly one f-family such that $L_F=L$, where $F=\{(A, B): B\subseteq L(A)\}$. Thus, between closure operations and f-families over Ω there exists an one-to-one correspondence.

Definition 1.4. Let R be a relation, F an f-family and L a closure operation over Ω. We say that R represents F (L) iff $F_R=F$ ($L_{F_R}=L$).

Definition 1.5. Let R be a relation, L a closure operation over Ω, and $K\subseteq\Omega$. We say that K is a key of R (of L) if $K\rightarrow\Omega$ $(L(K)=\Omega)$. K is a minimal key of R (of L) if K is a key of R (of L) and for any proper subset B of K, $B\not\xrightarrow[R]{}\Omega$ $(L(B)\neq\Omega)$. Denote $\mathcal{K}_R$ the set of all minimal keys of R and $\mathcal{K}_L$ that of L. Clearly, $K_i, K_j\in\mathcal{K}_R$ implies $K_i\not\subseteq K_j$. Systems of subsets of Ω satisfying this condition are Sperner-systems. Consequently, $\mathcal{K}_R$, $\mathcal{K}_L$ are Sperner-systems.

For a Sperner-system $\mathcal{K}$ we can define the set of antikeys of $\mathcal{K}$ (denoted by $\mathcal{K}^{-1}$) as follows:

$$\mathcal{K}^{-1}=\{B\subset\Omega: (K\in\mathcal{K})\rightarrow(K\not\subseteq B) \text{ and } (B\subset C)\rightarrow(\exists K\in\mathcal{K})(K\subseteq C)\}.$$

It is easy to see that $\mathcal{K}^{-1}$ is also a Sperner-system. Clearly, the elements of $\mathcal{K}^{-1}$ do not contain the elements of $\mathcal{K}$ and they are maximal for this property.

Definition 1.6. Let $R=\{h_1, \ldots, h_m\}$ be a relation over Ω. For $1\leq i<j\leq m$ denote E_{ij} the set $\{a\in\Omega: h_i(a)=h_j(a)\}$. We set $E_R=\{E_{ij}: 1\leq i<j\leq m\}$. E_R is called the equality set of R.

§ 2. Results

Now we give a necessary and sufficient condition for a relation representing a given f-family. It is a precise characterization for relations represent f-families.

Theorem 2.1. Let $R=\{h_1, \ldots, h_m\}$ be a relation and F an f-family over Ω. Then R represents F iff for every $A \subseteq \Omega$

$$L_F(A) = \begin{cases} \bigcap_{A \subseteq E_{ij}} E_{ij} & \text{if } \exists E_{ij} \in E_R \colon A \subseteq E_{ij}, \\ \Omega & \text{otherwise,} \end{cases}$$

where $L_F(A)=\{a \in \Omega \colon (A, \{a\}) \in F\}$ and E_R is the equality set of R.

Proof. It is easy to see that F_R is an f-family over Ω, first we prove that in an arbitrary relation R for all $A \subseteq \Omega$

$$L_{F_R}(A) = \begin{cases} \bigcap_{A \subseteq E_{ij}} E_{ij} & \text{if } \exists E_{ij} \in E_R \colon A \subseteq E_{ij}, \\ \Omega & \text{otherwise.} \end{cases}$$

We suppose that A is a set such that there is not an $E_{ij} \in E_R$ so that $A \subseteq E_{ij}$. Then for all h_i, $h_j \in R$ $\exists a \in A \colon h_i(a) \neq h_j(a)$. According to the definition of functional dependency $A \xrightarrow{R} \Omega$ holds. By the definition of the mapping L_{F_R} we obtain $L_{F_R}(A)=\Omega$. It is obvious that $L_{F_R}(\emptyset)= \bigcap_{E_{ij} \in E_R} E_{ij}$ holds. If $A \neq \emptyset$ and there is an $E_{ij} \in E_R$ so that $A \subseteq E_{ij}$, then we set $V=\{E_{ij} \colon A \subseteq E_{ij}, E_{ij} \in E_R\}$ and $E= \bigcap_{E_{ij} \in V} E_{ij}$. Clearly, $A \subseteq E$. If $V=E_R$ holds, then $(A, E) \in F_R$ holds. If $V \neq E_R$ holds, then it can be seen that for all $E_{ij} \in V$ $(\forall a \in A)(h_i(a)=h_j(a)) \to (\forall b \in E)(h_i(b)=h_j(b))$ and for all $E_{ij} \notin V$ $\exists a \in A \colon h_i(a) \neq h_j(a)$. Thus, $(A, E) \in F_R$ holds. By the definition of L_{F_R}, $E \subseteq L_{F_R}(A)$ holds. Clearly, by the definition of relation we have $E \subset \Omega$. From $A \subseteq E \subseteq L_{F_R}(A)$ and according to the definition of closure operation we obtain $(E, L_{F_R}(A)) \in F_R$. Now we assume that c is an attribute such that $c \notin E$. Consequently, there is an $E_{ij} \in V$ so that $c \notin E_{ij}$. Thus, $\exists h_i, h_j \in R \colon \forall b \in E \colon h_i(b)=h_j(b)$ holds, but $h_i(c) \neq h_j(c)$. According to Definition 1.1, $(E \cup c)$ does not depend on E. Thus, for all attributes $c \notin E$ $(E, E \cup c) \notin F_R$ holds. By the definition of L_{F_R} we obtain $L_{F_R}(A)= \bigcap_{E_{ij} \in V} E$. By Remark 1.1 it is easy to see that $F_R=F$ holds iff $L_{F_R}=L_F$ holds. The proof is complete. □

The following corollary is obvious.

Corollary 2.1. Let R be a relation and L a closure operation over Ω. Then R represents L iff for all $A \subseteq \Omega$

$$L(A) = \begin{cases} \bigcap_{A \subseteq E_{ij}} E_{ij} & \text{if } \exists E_{ij} \in E_R \colon A \subseteq E_{ij}, \\ \Omega & \text{otherwise.} \end{cases}$$ □

Definition 2.1. Let L be a closure operation over Ω. Let $Z(L)=\{A \subseteq \Omega \colon L(A)=A\}$, and $M(L)=\{A \subset \Omega \colon A \in Z(L), A \subset B \to L(B)=\Omega\}$ The elements of $Z(L)$ are called closed sets. $M(L)$ is the family of maximal closed sets (except Ω).

Clearly, $Z(L)$ is closed under intersection.

Definition 2.2. Let $N \subseteq P(\Omega)$. Denote N^+ the set $\{\cap N' : N' \subseteq N\}$. By convention $\cap \emptyset = \Omega$, i.e. N^+ always contains Ω. It can be seen that for all $E_{ij} \in E_R$ we have $E_{ij} \in Z(L_{F_R})$, i.e. $E_R^+ \subseteq Z(L_{F_R})$. By Theorem 2.1, $Z(L_{F_R}) \subseteq E_R^+$ holds. Clearly, if L_1, L_2 are two closure operations over Ω then $L_1 = L_2$ holds iff $Z(L_1) = Z(L_2)$. Consequently, the next corollary is clear.

Corollary 2.2. Let R be a relation and L a closure operation over Ω. Then R represents L iff $Z(L) = E_R^+$ holds. □

Definition 2.3. Let F be an f-family over Ω and $(A, B) \in F$. We say that (A, B) is a maximal right side dependency of F iff

$$\forall B' \quad (B \subseteq B') : (A, B') \in F \rightarrow B' = B.$$

Denote by $M(F)$ the set of all maximal right side dependencies of F. We say that B is a maximal side of F iff there is an A so that $(A, B) \in M(F)$. Denote $I(F)$ the set of all maximal sides of F.

It can be seen that $I(F) = Z(L_F)$. Consequently, the following corollary is obvious.

Corollary 2.3. Let F be an f-family and R a relation over Ω. Then R represents F iff $I(F) = E_R^+$. □

It is known ([1], [4]) that for an arbitrary non-empty Sperner-system $\mathcal{H}$ there is a relation R so that $\mathcal{H}_R = \mathcal{H}$.

Definition 2.4. Let R be a relation and $\mathcal{H}$ a Sperner-system over Ω. We say that R represents $\mathcal{H}$ iff $\mathcal{H}_R = \mathcal{H}$.

The next theorem is a useful precise characterization of relations which represent a given Sperner-system. First we define the following concept.

Definition 2.5. Let R be a relation over Ω, and E_R the equality set of R, i.e. $E_R = \{E_{ij} : 1 \leq i < j \leq m\}$, where $E_{ij} = \{a \in \Omega : h_i(a) = h_j(a)\}$. Let $T_R = \{A \subset \Omega : \exists E_{ij} \in E_R : E_{ij} = A$ and $\overline{\exists} E_{st} \in E_R : A \subset E_{st}\}$. Then T_R is called the maximal equality system of R.

Theorem 2.2. Let $\mathcal{H}$ be a non-empty Sperner-system and R a relation over Ω. Then R represents $\mathcal{H}$ iff $\mathcal{H}^{-1} = T_R$, where T_R is the maximal equality system of R.

Proof. As $\mathcal{H}$ is a non-empty Sperner-system, $\mathcal{H}^{-1}$ exists. On the other hand, $\mathcal{H}$ and $\mathcal{H}^{-1}$ are uniquely determined by each other, we obtain $\mathcal{H}_R = \mathcal{H}$ holds iff $\mathcal{H}_R^{-1} = \mathcal{H}^{-1}$ does. Consequently, we must prove that $\mathcal{H}_R^{-1} = T_R$.

It is obvious that F_R is an f-family over Ω. Now we suppose that A is an antikey of $\mathcal{H}_R$. Clearly, $A \neq \Omega$. If there is a B such that $A \subset B$ and $A \xrightarrow[R]{} B$ then by definition of antikeys we obtain $B \xrightarrow[R]{} \Omega$. Hence $A \xrightarrow[R]{} \Omega$ holds. This contradicts to $K \in \mathcal{H}_R$: $K \not\subseteq A$. So $A \in I(F_R)$ holds. If there is a B' so that $B' \neq \Omega$, $B' \in I(F_R)$, and $A \subset B'$, then B' is a key of R. This contradicts to $B' \neq \Omega$. Thus, $A \in I(F_R) \setminus \Omega$ and $\overline{\exists} B' \ (B' \in I(F_R) \setminus \Omega) : A \subset B'$. On the other hand, $\Omega \notin T_R$ by definition of R. It is easy

to see that $E_{ij} \in I(F_R)$. Hence $T_R \subseteq I(F_R)$ holds. If D is a set such that $\forall C \in T_R: D \not\subseteq C$, then from Definition 1.1, D is a key of R. Consequently, T_R is the set of maximal distinct elements of $I(F_R)$. So we obtain $A \in T_R$.

Conversely, we assume that $A \in T_R$. According to the definition of a relation and T, we have $A \not\to_R \Omega$, i.e. $\forall K \in \mathscr{K}_R: K \not\subseteq A$. On the other hand, by definition of T_R for all D $(A \subset D)$ $D \to_R \Omega$ holds. Consequently, by the definition of antikeys $A \in \mathscr{K}_R^{-1}$. The proof is complete. □

Now we investigate the connection between closure operations.

Lemma 2.1. [6] Let L be a closure operation over Ω, and $\mathscr{K}_L$ the set of minimal keys of L. Then $\mathscr{K}_L^{-1} = M(L)$. □

Definition 2.6. [3] Let Q be a set of all closure operations over Ω. An ordering over Q is defined as follows:

For $L, L' \in Q$ let $L \leqq L'$ iff for all $A \subseteq \Omega$, $L'(A) \subseteq L(A)$. It can be seen that Q is a partially ordered set for this ordering. If $L \leqq L'$ but $L \neq L'$ then the notation $L < L'$ is used.

Theorem 2.3. [3] Let L, L' be two closure operations over Ω. Then $L \leqq L'$ iff $Z(L) \subseteq Z(L')$. □

Based on Theorem 2.3 it is easy to see that $L < L'$ iff $Z(L) \subset Z(L')$.

Theorem 2.4. Let $\mathscr{K}$ be a non-empty Sperner-system over Ω, and $\mathscr{K}^{-1}$ the set of all antikeys of $\mathscr{K}$. Let

$$L(A) = \begin{cases} \bigcap\limits_{\substack{A \subseteq B \\ B \in \mathscr{K}^{-1}}} B & \text{if there is a } B \in \mathscr{K}^{-1}: A \subseteq B, \\ \Omega & \text{otherwise.} \end{cases}$$

Then L is a closure operation over Ω and $\mathscr{K}_L = \mathscr{K}$. If L' is an arbitrary closure operation over Ω such that $\mathscr{K} = \mathscr{K}_{L'}$, than $L \leqq L'$ holds.

Proof. Clearly, L is a closure operation over Ω. Also it is obvious that for all $B \in \mathscr{K}^{-1}$ we have $L(B) = B$, i.e. $B \in Z(L)$. On the other hand, $\mathscr{K}^{-1}$ being a Sperner-system over Ω we obtain $M(L) = \mathscr{K}^{-1}$. By Lemma 2.1 $\mathscr{K}^{-1} = \mathscr{K}_L^{-1}$. Since $\mathscr{K}$ and $\mathscr{K}^{-1}$ are uniquely determined by each other $\mathscr{K}_L = \mathscr{K}$.

Suppose that L' is an arbitrary closure operation so that $\mathscr{K} = \mathscr{K}_{L'}$, it can be seen that $Z(L) = (\mathscr{K}^{-1})^+$. By Lemma 2.1, $M(L') = \mathscr{K}^{-1} = \mathscr{K}_{L'}^{-1}$. Consequently, $M(L') = M(L) = \mathscr{K}^{-1}$. Hence $Z(L) \subseteq Z(L')$ holds and by Theorem 2.3 we obtain $L \leqq L'$. Clearly, L is the closure operation for which $\mathscr{K} = \mathscr{K}_L$ and for any closure operation L' such that $\mathscr{K} = \mathscr{K}_{L'}$, and $L \neq L'$ we obtain $L < L'$. The theorem is proved. □

Corollary 2.4. Let $\mathscr{K}$ be a non-empty Sperner-system over Ω. Denote by V the set of all closure operations over Ω the minimal keys of which are exactly the elements of $\mathscr{K}$. Then L as constructed in Theorem 2.4 is the unique minimal element of the partially ordered set V for the ordering defined. □

Remark 2.1. In [6] we constructed an algorithm which computes the set of all antikeys of an arbitrary Sperner-system. By Theorem 2.4 and this algorithm we can explicitly construct the closure operation L for which $\mathcal{K}=\mathcal{K}_L$ to an arbitrarily given Sperner-system $\mathcal{K}$. □

The next remark shows that conversely, the set of all minimal keys of a given closure operation can be found.

Remark 2.2. In [5] we construct an algorithm which determines the set H such that $H^{-1}=\mathcal{K}$ for a given Sperner-system $\mathcal{K}$. Thus, if $\mathcal{K}$ is a set of antikeys then H is a set of minimal keys. Consequently, from a given closure operation L we can construct the family $M(L)$. By Lemma 2.1 $M(L)=\mathcal{K}_L^{-1}$ holds. From $M(L)$ we can determine the set of all minimal keys of L by this algorithm. □

Резюме

Одно из главных понятий теории реляционных баз данных является понятие функциональной зависимости. Статья изучает реляции которые представляют данную фамилию функциональных зависимостей, операции замыкания и системы Спернера. А также изучается связь между операциями замикания и минимальными ключами.

COMPUTER AND AUTOMATION INSTITUTE
HUNGARIAN ACADEMY OF SCIENCES
VICTOR HUGO U. 18—22
1123 BUDAPEST
HUNGARY

References

[1] W. W. Armstrong, Dependency structures of data base relationships. Information Processing 74, North-Holland (1974) 580—583.
[2] E. F. Codd, Relational model of data for large shared data banks. Communications of ACM, 13 (1970), 377—384.
[3] G. Burosch, J. Demetrovics, G. O. H. Katona, The poset of closures as a model of changing databases. Order, 4 (1987), 127—142.
[4] J. Demetrovics, On the equivalence of candidate keys with Sperner-systems. Acta Cybernetica 4 (1979), 3, 247—252.
[5] J. Demetrovics, V. D. Thi, Relations and minimal keys. Acta Cybernetica, Szeged 8 (1988), 3. 279—285.
[6] V. D. Thi, Minimal keys and antikeys. Acta Cybernetica, 7 (1986), 4, 361—371.

(Received Dec. 4, 1986)

Relations and minimal keys*

J. DEMETROVICS and V. D. THI

Abstract

The main purpose of this paper is to prove that the time complexity of finding a relation representing a given Sperner-system K is exactly exponential in the number of elements of K. Conversely, we show that if $NP \neq P$ then the time complexity of finding a set of all minimal keys of given relation R is also exactly exponential in the size of R.

§ 1. Introduction

The minimal keys play important roles for the logic and structural investigation of relational datamodel. In this datamodel the form of data storage is matrix (relation), rows of which represent records and columns represent attributes. A set of minimal keys of a relation forms a Sperner-system. Sets of minimal keys and Sperner-systems are equivalent in the sense that for an arbitrary Sperner-system K there exists a relation R such that the minimal keys of R are exactly the elements of K (cf. [3]), i.e. R represents K.

In this paper we prove that the time complexity of finding a relation representing a given Sperner-system K is exactly exponential in the number of elements of K, i.e. we shall show that there is an algorithm that determines a relation representing a given Sperner-system K in time exponential in the number of elements of K, and there is no algorithm which finds a relation representing K and the time complexity of which is polynomial in the number of elements of K. Let P denote the class of problems that can be solved in deterministic polynomial time and let NP denote the class of problems that can be solved in nondeterministic polynomial time. It is shown that if $NP \neq P$ then the time complexity of finding a set of all minimal keys of a given relation R is exactly exponential in the number of rows and columns of R.

We start with some necessary definitions, and in § 2 formulate our results.

*This paper was supported by a grant from the Hungarian Academy of Sciences OTKA Nr. 1066.

Definition 1.1. Let $R=\{h_1, \ldots, h_m\}$ be a relation over Ω, and $A, B\subseteq\Omega$. Then we say that B functionally depends on A in R (denoted $A\overset{f}{\underset{R}{\to}}B$) iff

$$(\forall h_i, h_j\in R)((\forall a\in A)(h_i(a)=h_j(a))\to((\forall b\in B)(h_i(b)=h_j(b))).$$

Let $F_R=\{(A, B): A\overset{f}{\underset{R}{\to}}B\}$. F_R is called the full family of functional dependencies of R.

Definition 1.2. Let Ω be a finite set, and denote $P(\Omega)$ its power set. Let $F\subseteq\subseteq P(\Omega)\times P(\Omega)$. We say that F is an f-family over Ω iff for all $A, B, C, D\subseteq\Omega$:

(F1) $(A, A)\in F$;
(F2) $(A, B)\in F, (B, C)\in F\to(A, C)\in F$;
(F3) $(A, B)\in F, A\subseteq C, D\subseteq B\to(C, D)\in F$;
(F4) $(A, B)\in F, (C, D)\in F\to(A\cup C, B\cup D)\in F$.

Clearly, F_R is an f-family over Ω.

It is known [2] that if F is an arbitrary f-family, then there is a relation R over Ω such that $F_R=F$.

Definition 1.3. The mapping $L: P(\Omega)\to P(\Omega)$ is called a closure operation over Ω iff for every $A, B\subseteq\Omega$

(1) $A\subseteq L(A)$;

(2) $A\subseteq B\to L(A)\subseteq L(B)$;

(3) $L(L(A))=L(A)$.

Definition 1.4. Let R be a relation, L be a closure operation over Ω, and $A\subseteq\Omega$. A is a key of R (a key of L) if $A\overset{f}{\underset{R}{\to}}\Omega$ $(L(A)=\Omega)$. A is a minimal key of R (a minimal key of L) if A is a key of R (a key of L), but $B\overset{f}{\underset{R}{\not\to}}\Omega$ $(L(B)\neq\Omega)$ for any proper subset B of A. Denote K_R (K_L) the set of all minimal keys of R (L). Clearly, K_R, K_L are Sperner-systems over Ω.

Definition 1.5. Let K be a Sperner-system over Ω. We define the set of antikeys of K, denoted by K^{-1}, as follows:

$$K^{-1}=\{A\subsetneq\Omega: (B\in K)\to(B\not\subseteq A) \quad \text{and} \quad (A\subsetneq C)\to(\exists B\in K)\ (B\subseteq C)\}.$$

It is easy to see that K^{-1} is also a Sperner-system over Ω.

Theorem 1.1. ([2], [3]) If K is an arbitrary Sperner-system, then there is a closure operation L for which $K_L=K$. □

In this paper we always assume that if a Sperner-system plays the role of the set of minimal keys (antikeys), then this Sperner-system is not empty (doesn't contain Ω).

Definition 1.6. ([2]) Let F be an f-family over Ω, and $(A, B)\in F$. We say that (A, B) is a maximal right side dependency of F iff $\forall B'$ $(B\subseteq B'): (A, B')\in F\to B=B'$. Denote by $M(F)$ the set of all maximal right side dependencies of F. We say that B

is a maximal side of F iff there is an A such that $(A, B)\in M(F)$. Denote by $I(F)$ the set of all maximal sides of F.

In this paper we regard the comparison of two attributes to be the elementary step of algorithms. Thus, if we assume that subsets of Ω are represented as sorted lists of attributes, then a Boolean operation on two subsets of Ω requires at most $|\Omega|$ elementary steps.

Definition 1.7. Let R be a relation, and K be a Sperner-system over Ω. We say that R represents K iff $K_R=K$.

§ 2. Results

Definition 2.1. Let L be a closure operation over Ω.
Denote $Z(L)=\{A\in P(\Omega)\colon L(A)=A\}$,

$$T(L)=\{A\in P(\Omega)\colon L(A)=A \text{ and } A\subsetneq B\to L(B)=\Omega\}.$$

The elements of $Z(L)$ are called closed sets. $T(L)$ is called a maximal family of L.

Lemma 2.1. ([5]) Let L be a closure operation over Ω. Then $K_L^{-1}=T(L)$. □

Theorem 2.1. ([4]) Let K be a Sperner-system over Ω. Let $s(K)=\min\{m\colon |R|=m, K_R=K, R \text{ is a relation over } \Omega\}$. Then $\sqrt{2|K^{-1}|}\leqq s(K)\leqq |K^{-1}|+1$. □

Theorem 2.2. The time complexity of finding a relation representing a given Sperner-system K is exactly exponential in the number of elements of K.

Proof. We have to prove that:
(1) There exists an algorithm that determines a relation representing a given Sperner-system K in time exponential in the number of elements of K.
(2) There is no algorithm that finds a relation representing K in time polynomial in the number of elements of K. Based on (1) and (2) it is clear that the time complexity of any algorithm that determines a relation representing a given Sperner-system is at least exponential in the number of elements of this Sperner-system.
For (1): First we construct an algorithm which finds the set of antikeys from a given Sperner-system, as follows:
Let us given an arbitrary Sperner-system $K=\{B_1, \ldots, B_m\}$ over Ω. We set $K_1=\{\Omega\setminus\{a\}\colon a\in B_1\}$. It is obvious that $K_1=\{B_1\}^{-1}$. Let us suppose that we have constructed $K_q=\{B_1, \ldots, B_q\}^{-1}$ for $q<m$. We assume that $X_1, \ldots, X_{t_q}$ are the elements of K containing B_{q+1}. So $K_q=F_q\cup\{X_1, \ldots, X_{t_q}\}$, where $F_q=\{A\in K_q\colon B_{q+1}\not\subseteq A\}$. For all i $(i=1, \ldots, t_q)$ we construct the antikeys of $\{B_{q+1}\}$ on X_i in an analogous way as K_1, which are the maximal subsets of X_i not containing B_{q+1}. We denote them by $A_1^i, \ldots, A_{r_i}^i$ $(i=1, \ldots, t_q)$. Let

$$K_{q+1}=F_q\cup\{A_p^i\colon A\in F_q\to A_p^i\not\subseteq A,\quad 1\leqq i\leqq t_q,\quad 1\leqq p\leqq r_i\}.$$

Clearly, because K and K^{-1} are uniquely determined by one another, the determination of K^{-1} based on our algorithm does not depend on the order of $B_1, \ldots, B_m$.
In [5] we proved that for every q $(1\leqq q\leqq m)$, $K_q=\{B_1, \ldots, B_q\}^{-1}$, i.e. $K_m=K^{-1}$, and the worst-case time of this algorithm is exponential not only in the number of

elements of K, but also in the number of attributes. Now we construct the following algorithm:

Step 1: Based on the above algorithm we construct K^{-1}.

Step 2: Let $K^{-1}=\{A_1, \ldots, A_t\}$ be a set of antikeys. Let $R=\{h_0, h_1, \ldots, h_t\}$ be a relation over Ω given as follows:

For all $a\in\Omega$, $h_0(a)=0$,

$$\text{for } i\ (1\leqq i\leqq t),\quad h_i(a)=\begin{cases}0 & \text{if } a\in A_i,\\ i & \text{otherwise.}\end{cases}$$

In [4], it has been proved that R represents K. It is clear that the complexity of this algorithm is the complexity of the algorithm that finds the set of antikeys.

For (2): Let us take a partition $\Omega=X_1\cup\ldots\cup X_m\cup W$, where $m=\left[\frac{n}{3}\right]$, and $|X_i|=3$ $(1\leqq i\leqq m)$. Let

$K=\{B\colon |B|=2, B\subseteq X_i \text{ for some } i\}$ if $|W|=0$,

$K=\{B\colon |B|=2, B\subseteq X_i \text{ for some } i\colon 1\leqq i\leqq m-1 \text{ or } B\subseteq X_m\cup W\}$ if $|W|=1$,

$K=\{B\colon |B|=2, B\subseteq X_i \text{ for some } i\colon 1\leqq i\leqq m \text{ or } B=W\}$ if $|W|=2$.

It is clear that

$K^{-1}=\{A\colon |A\cap X_i|=1, \forall i\}$ if $|W|=0$,

$K^{-1}=\{A\colon |A\cap X_i|=1\ (1\leqq i\leqq m-1) \text{ and } |A\cap(X_m\cup W)|=1\}$ if $|W|=1$,

$K^{-1}=\{A\colon |A\cap X_i|=1\ (1\leqq i\leqq m) \text{ and } |A\cap W|=1\}$ if $|W|=2$.

Let $f\colon N\to N$ (N is the set of natural numbers) be the function defined as follows:

$$f(n)=\begin{cases}3^{n/3} & \text{if } n\equiv 0 \pmod 3,\\ 3^{[n/3]}\cdot 4/3 & \text{if } n\equiv 1 \pmod 3,\\ 3^{[n/3]}\cdot 2 & \text{if } n\equiv 2 \pmod 3.\end{cases}$$

It can be seen that $f(n)=|K^{-1}|$. Clearly, $n-1\leqq|K|\leqq n+2$ and $3^{[n/4]}<f(n)$, where $n=|\Omega|$, i.e. $3^{[n/4]}<|K^{-1}|$. According to Theorem 2.1 we have $\sqrt{2}\cdot 3^{[n/8]}\leqq s(K)$, i.e. the number of rows of minimal relation representing K is greater than $\sqrt{2}\cdot 3^{[n/8]}$. Thus, for an arbitrary set of attributes we always can construct an example, in which the number of K is not greater than $|\Omega|+2$, but the number of rows of any relation representing K is exponential not only in the number of attributes, but also in the number of elements of K. Hence, there is no algorithm which finds a relation representing a given Sperner-system and the time complexity of which is polynomial in the number of elements of Sperner-system. The theorem is proved. □

Now we give a necessary and sufficient condition for a relation to represent a given Sperner-system. We define the following concept.

Definition 2.2. Let $R=\{h_1, \ldots, h_m\}$ be a relation over Ω. Let $E_R=\{E_{ij}: 1\leqq i<j\leqq m\}$, where $E_{ij}=\{a\in\Omega: h_i(a)=h_j(a)\}$. Let

$$M_R = \{A\in P(\Omega): \exists E_{ij}\in E_R: E_{ij}=A \text{ and } \overline{\exists} E_{st}\in E_R: A\subsetneqq E_{st}\}.$$

M_R is called the maximal equality system of R.

Theorem 2.3. Let K be a non-empty Sperner-system, and R be a relation over Ω. Then R represents K iff $K^{-1}=M_R$, where M_R is the maximal equality system of R.

Proof. Because K is a non-empty Sperner-system, K^{-1} exists. On the other hand, K and K^{-1} are uniquely determined by each other, hence $K_R=K$ holds iff $K_R^{-1}=K^{-1}$ holds. Consequently, we must prove that $K_R^{-1}=M_R$. It is obvious that F is a f-family. Now we suppose that A is an antikey of K. It can be seen that $A\neq\Omega$. If there exists a B such that $A\subsetneqq B$ and $A\overset{f}{\underset{R}{\rightarrow}}B$, then by the definition of antikey we have $B\overset{f}{\underset{R}{\rightarrow}}\Omega$. Hence $A\overset{f}{\underset{R}{\rightarrow}}\Omega$ holds. This contradicts $C\in K_R: C\nsubseteqq A$. So $A\in I(F_R)$ holds. If there is a B' so that $B'\neq\Omega$, $B'\in I(F_R)$, and $A\subsetneqq B'$ then B' is a key of R. This contradicts to $B'\neq\Omega$. Consequently, $A\in I(F_R)\setminus\Omega$ and $\overline{\exists} B'\,(B'\in I(F_R)\setminus\Omega)$: $A\subsetneqq B'$. On the other hand, according to the of relation $\Omega\sharp M_R$. It is easy to see that $E_{ij}\in I(F_R)$. Thus, $M_R\subseteqq I(F_R)$ holds. If D is a set such that $\forall C\in M_R: D\nsubseteqq C$, then by the definition of functional dependency, D is a key of R. Consequently, M_R is the set of maximal distinct elements of $I(F_R)$. So we have $A\in M_R$.

Conversely, we assume that $A\in M_R$. According to the definition of relation and M_R we obtain $A\overset{f}{\underset{R}{\nrightarrow}}\Omega$, i.e. $\forall B\in K_R: B\nsubseteqq A$. On the other hand, because A is a maximal equality set, for all D $(A\subsetneqq D)$ $D\overset{f}{\underset{R}{\rightarrow}}\Omega$ holds. Consequently, by the definition of antikey $A\in K_R^{-1}$. The theorem is proved. □

It can be seen that the time complexity of finding the set of antikeys of R is polynomial in the number of rows and columns of R. We construct the following algorithm for finding a minimal key. Let H be a Sperner-system. We take a B $(B\in H)$ and an $a\in\Omega\setminus B$. We suppose that $B=\{b_1, \ldots, b_m\}$. Let $G=\{B_j\in H: a\notin B_j\}$ and $T_0=B\cup\{a\}$. We define

$$T_{q+1}=\begin{cases} T_q\setminus\{b_{q+1}\} & \text{if } \forall B_i\in H\setminus G: T_q\setminus\{b_{q+1}\}\nsubseteqq B_i,\\ T_q & \text{otherwise.}\end{cases}$$

Lemma 2.2. ([5]) If H is a set of antikeys, then $T_0, T_1, \ldots, T_m$ are the keys and T_m is a minimal key. □

It is easy to see that the worst-case time of finding T_m is $O(|\Omega|^2\cdot|H|)$.

Lemma 2.3. Let H be a Sperner-system over Ω, and let $H^{-1}=\{B_1, \ldots, B_m\}$ be a set of antikeys of H, $T\subseteqq H$. Then $T\subsetneqq H$ and $T\neq\emptyset$ if and only if there is a $B\subseteqq\Omega$ such that $B\in T^{-1}$ and $B\nsubseteqq B_i$ $(\forall i: 1\leqq i\leqq m)$.

Proof. Suppose that there exists a B such that $B\in T^{-1}$ and $B\nsubseteqq B_i$ $(\forall i: 1\leqq i\leqq m)$. From the definition of the set of antikeys and by $T^{-1}\neq\emptyset$, we have $T\neq\emptyset$, and for all C $(C\in T)$, B does not contain C. If there is a B_i such that $B_i\in H^{-1}$ and $B_i\subset B$,

then it is obvious that B is a key. If $H^{-1} \cup B$ is a Sperner-system, then by Theorem 1.1 there exists a closure operation L such that $H = K_L$. It is clear that if $L(B) \neq \Omega$, then from Lemma 2.1 there is a B_i ($B_i \in H^{-1}$) such that $L(B) \subseteq B_i$. Consequently, $B \subseteq B_i$. This conflicts with the fact that $B \not\subseteq B_i$ ($\forall i$: $1 \leqq i \leqq m$). That is, B is a key. Hence there is an A ($A \subseteq \Omega$) such that $A \subseteq B$ and $A \in H \setminus T$. It is easy to see that $T \subset H$.

Conversely, we suppose that $T \subset H$ and $T \neq \emptyset$. It is obvious that there is an A such that $A \in H \setminus T$. From H is a Sperner-system we have $A \cup T$ is a Sperner-system. Denote B the biggest set such that $A \subseteq B$ and $B \cup T$ is also a Sperner-system. It is clear that, B always exists and from the definition of antikeys we have $B \in T^{-1}$. By $A \in H$ it can be seen that $A \not\subseteq B_i$ ($\forall i$: $1 \leqq i \leqq m$). By $A \subseteq B$ we have $B \not\subseteq B_i$ ($\forall i$: $1 \leqq i \leqq m$). The theorem is proved. □

Let $K = \{B_1, \ldots, B_m\}$ be a Sperner-system over Ω. We have to construct H, where $H^{-1} = K$. We construct H by induction.

Algorithm 2.1. *Step 1:* Using a minimal key algorithm we construct an A_1, ($A_1 \in H$). We set $K_1 = \{A_1\}$.

Step $i+1$: If there is a $B \in K_i^{-1}$ such that $B \not\subseteq B_j (\forall j$: $1 \leqq j \leqq m)$, then by algorithm which finds a minimal key we determine an A_{i+1} ($A_{i+1} \in H$) and $A_{i+1} \subseteq B$. After that, let $K_{i+1} = K_i \cup \{A_{i+1}\}$. In the converse case we set $H = K_i$. □

Based on Lemma 2.3 there is a natural number p so that $K_p = H$. It can be seen that the time complexity of Algorithm 2.1 is also exponential in the number of attributes.

Lemma 2.4. The following problem is NP-complete:

Given a Sperner-system $K = \{B_1, \ldots, B_m\}$ over $\Omega = \{a_1, \ldots, a_n\}$ and integer k ($k \leqq n$), decide whether there exists an $A \subseteq \Omega$ such that $|A| \leqq k$ and $A \not\subseteq B_i$ ($i = 1, \ldots, m$), i.e. decide whether there exists a key having cardinality not greater than k, if K is the set of antikeys.

Proof. We nondeterministically choose a subset A of Ω so that $|A| \leqq k$ and decide whether A is not a subset of $B_i (i = 1, \ldots, m)$. It is obvious that this algorithm is nondeterministic polynomial. Thus, the problem lies in NP. It is known [1] that the vertex cover problem is NP-complete:

Given integer k and non-directed graph $G = \langle V, E \rangle$, where V is a set of vertices and E is set of edges, decide whether or not G has a vertex cover having cardinality not greater than k.

We shall prove that the vertex cover problem is polynomially reducible to our problem.

Let $G = \langle V, E \rangle$ be a non-directed graph, $k \leqq |V|$. We set $\Omega = V$ and $K = \{\Omega \setminus \{a_i, a_j\}: (a_i, a_j) \in E\}$.

If $A \subseteq \Omega$, $|A| \leqq k$ and $A \not\subseteq B$ ($\forall i = 1, \ldots, m$), then according to definition of K we have $A \cap \{a_i, a_j\} \neq \emptyset (\forall (a_i, a_j) \in E)$. Consequently, A is a vertex cover of G.

Conversely, if A is a vertex cover of G, then by definition of K and definition of vertex cover we have $A \not\subseteq B_i$ ($\forall i = 1, \ldots, m$). Hence, $A \not\subseteq B_i$ ($\forall i = 1, \ldots, m$) holds if and only if A is a vertex cover of G. The Lemma is proved. □

Based on Lemma 2.4 and Step 2 of the algorithm which determines a relation representing a given Sperner-system in Theorem 2.2, the following corollary is obvious.

Corollary 2.1. The following problem is NP-complete: Given integer k and relation, decide whether or not there exists a key having cardinality not greater than k. □

Theorem 2.4. The time complexity of finding a set of all minimal keys of a given relation R is exactly exponential in the number of rows and columns of R.

Proof. For a given arbitrary relation R we construct the following algorithm which determines the set of all minimal keys of R.

Step 1: According to Theorem 2.3 we construct the set of antikeys of R.

Step 2: Based on Algorithm 2.1 we determine the set of all minimal keys of R.

By Lemma 2.2, Lemma 2.3, Theorem 2.3 and Algorithm 2.1, it is clear that the worst-case time of this algorithm is exponential in the number of rows and columns of R.

According to Lemma 2.4 and Corollary 2.1, it can be seen that there is no algorithm which finds a set of all minimal keys of a given relation and the time complexity of which is polynomial in the size of this relation. The theorem is proved. □

Based on Theorem 2.1 and Theorem 2.4 it can be seen that the problem of finding a relation representing a given Sperner-system and finding a set of all minimal keys of a relation are inherently difficult.

Резюме

В настоящей работе изучается связь между отношениями и минимальными ключами.

COMPUTER AND AUTOMATION INSTITUTE
HUNGARIAN ACADEMY OF SCIENCES
1132 BUDAPEST
VICTOR HUGO U. 18—22
HUNGARY

References

[1] A. V. Aho, J. E. Hopcroft, J. D. Ullman, The design and analysis of computer algoritms. Addison-Wesley, Reading, Mass. 1974.
[2] W. W. Armstrong, Dependency structures of data base relationships. Inf. Proc. 74 North-Holland Pub. Co. (1974) 580—583.
[3] J. Demetrovics, On the equivalence of candidate keys with Sperner-systems. Acta Cybernetica 4 (1979) 247—252.
[4] J. Demetrovics, Z. Füredi, G. O. H. Katona, Minimum matrix representation of closure operations. Discrete Applied Mathematics 11 (1985) 115—128.
[5] V. D. Thi, Minimal keys and antikeys. Acta Cybernetica 7 (1986), 361—371.

(Received March 12, 1987)

A new approach to defining software complexity measures

Z. DARÓCZY, L. VARGA

Abstract

A general method is given for defining software complexity measures. Properties of the complexity measure are given by functional equation. Three cases of functional equations are discussed. Many known software complexity measures are given as special cases of the solutions of functional equations and a new measure is also presented.

Introduction

It is a fact of common knowledge, that both simple and complicated programs can be developed for the solution of a given problem, whatever its inherent complexity. Therefore software complexity can be investigated independently of the complexity of a problem. During recent years many efforts have been made to create quantifiable measures of software complexity ([1], [3]), to use objective complexity measures in programming methodology and to validate these uses with empirical researches [2].

In spite of the importance of software complexity it is insufficiently known and defined. In this paper a new approach to defining abstract properties of software complexity over the class of structured programs is proposed by the help of using functional equations. Many known measures of software complexity can be obtained as special cases of the solutions of functional equations, and a new measure is also presented.

Software complexity measure

Given a system (X, F, A) where X is a set of data, F is a set of functions $(f: X \to X)$ and A is a set of predicates $(a: X \to \text{Bool})$.

Definition 1. Simple programs are:

1. null, with program function x,
2. assignment f, with program function $f(x)$, for all $f \in F$.

Let SP be the set of all simple programs.

Definition 2. Structured programs are:

1. Simple programs.
2. Sequence $B(u, v)$ with the program function $p_v(p_u(x))$;
3. Selection I—T—E $(a; u, v)$ with the program function **if** $a(x)$ **then** $p_u(x)$ **else** $p_v(x)$;
4. Repetition W—D$(a; u)$ with the program function $p(x)=$**if** $a(x)$ **then** $p(p_u(x))$ **else** x;

where $a \in A$, u and v are structured programs with program functions $p_u(x)$, $p_v(x)$ respectively.

Let S be the set of all structured programs.

Given the complexity measures

$$b\colon \mathrm{SP} \to N,$$

$$c\colon A \to N$$

and the homomorphisms

$$g\colon N^2 \to N,$$

$$h\colon N^3 \to N,$$

$$j\colon N^2 \to N.$$

Definition 3. Complexity measures of $B(s_1, s_2)$, I—T—E $(a; s_1, s_2)$, W—D $(a; s)$ are:

$$b(B(s_1, s_2)) = g(b(s_1), b(s_2)),$$

$$b(\text{I—T—E}(a;\ s_1, s_2)) = h(c(a), b(s_1), b(s_2)),$$

$$b(\text{W—D}(a;\ s)) = j(c(a),\ b(s).$$

The question is what kind of functions g, h, j characterize the software complexity measure of structured programs sufficiently? In order to find an appropriate complexity measure the properties of functions g, h, j will be given by functional equations.

First approximation. If the functional equations

$$g(x+x', y+y') = g(x, y)+g(x', y'),$$

$$h(x+x', y+y', z+z') = h(x, y, z)+h(x', y', z'),$$

$$j(x+x', y+y') = j(x, y)+j(x', y'),$$

hold, then the functions g, h, j give an *acceptable measure* for each structured program.

Theorem 1.

$$g(x, y) = c_1x + c_2y,$$
$$h(x, y, z) = d_1x + d_2y + d_3z,$$
$$j(x, y) = e_1x + e_2y$$

where $c_1, c_2, d_1, d_2, d_3, e_1, e_2$ are integer constants.

Proof.

$$x' = y = 0 \Rightarrow g(x, y') = g(x, 0) + g(0, y'),$$
$$x = x' = y = y' = 0 \Rightarrow \quad g(0, 0) = 0,$$
$$y = y' = 0 \Rightarrow g(x+x', 0) = g(x, 0) + g(x', 0).$$

This is the well known Chauchy equation, which has the following solution:

$$g(x, 0) = c_1x.$$

Similarly we have

$$x = x' = 0 \Rightarrow g(0, y+y') = g(0, y) + g(0, y') \Rightarrow g(0, y) = c_2y.$$

That is, $g(x, y) = c_1x + c_2y$.

Second approximation. If

$$g(x+x', y+y') = g(x, y) + g(x', y'),$$
$$h(x+x', y+y', z+z') = h(x, y, z) + h(x, y', z') + h(x', y, z) + h(x', y', z'),$$
$$j(x+x', y+y') = j(x, y) + j(x, y') + j(x', y) + j(x', y'),$$

then g, h, j give an *adequate measure* for each structured program.

Theorem 2.

$$g(x, y) = c_1x + c_2y,$$
$$h(x, y, z) = x(d_1y + d_2z)$$
$$j(x, y) = exy$$

Proof.

$$x = x' = y = y' = 0 \Rightarrow j(0, 0) = 0$$
$$x' = y = y' = 0 \Rightarrow \quad j(x, 0) = 0$$
$$x = x' = y' = 0 \Rightarrow \quad j(0, y) = 0$$
$$y' = 0 \Rightarrow j(x+x', y) = j(x, y) + j(x', y) \Rightarrow j(x, y) = e(y)x$$
$$x' = 0 \wedge x \neq 0 \Rightarrow e(y+y') = e(y) + e(y') \Rightarrow e \cdot y$$

That is

$$j(x, y) = exy.$$

Similarly we have

$$h(x, y, z) = x(d_1 y + d_2 z).$$

Third approximation. If

$$g(x+x', y+y') = g(x, y)+g(x', y')$$
$$h(x+x', y+y', z+z') = g(x+x', y, z)+g(x+x', y', z')$$
$$j(x, y) = h(x, y, l)$$

then g, h, j give a *correct measure* for each structured program.

Theorem 3.

$$g(x, y) = c_1 x_1 + x_2 y$$
$$h(x, y, z) = d_1(x)y + d_2(x)z$$
$$j(x, y) = d_1(x)y + d_2(x)$$

where $d_1(x)$, $d_2(x)$ are unknown functions.

Proof.

$$x' = 0 \Rightarrow g(x, y+y', z+z') = g(x, y, z)+g(x, y', z') \Rightarrow$$
$$\Rightarrow g(x, y, z) = d_1(x)y + d_2(x)z$$

Special cases

1. *First approximation*

$$g(x, y) = x+y$$
$$h(x, y, z) = x+y+z$$
$$j(x, y) = x+y$$

1.1. Let $b(f_i)=b_i, \quad f_i \in F;$

$$c(a_i) = c_i, \quad a_i \in A.$$

Then

$$b(s) = \sum_{i=1}^{\varphi} b_i + \sum_{i=1}^{\pi} c_i$$

where $s \in S$ and

π=number of predicates in s;
φ=number of functions in s.

1.2. Let $b_i=c_i=1$ for $i=1, 2, \ldots$, then

$$b(s) = \varphi+\pi.$$

1.3. Let $c_i=1$ and $b_i=0$ for $i=1, 2, \ldots$, then $b(s)=\pi$,

which gives the well known McCabe metric [1].

2. *Second approximation*

$$g(x, y) = x+y$$

$$h(x, y, z) = x(y+z)$$

$$j(x, y) = xy$$

2.1. Let $b(f_i)=b_i, \quad f_i \in F,$

$$c(a_i) = c_i, \quad a_i \in A,$$

then we get the Prather measure [3].

3. *Third approximation*

$$g(x, y) = x+y$$

$$h(x, y, z) = d_1(x)y+d_2(x)z$$

$$j(x, y) = d_1(x)y+d_2(x)$$

3.1. Let $d_i(c(a_j))=d_{ij}$; $i=1, 2$; $a_j \in A$, where
d_{1j}=number of "true" value in the operation table of predicate a_j;
d_{2j}=number of "false" value in the operation table of predicate a_j
and $b(f_i)=b_i$, $f_i \in F$ which produces a new measure.

An example

Let

$$s\text{: } \textbf{if } a_1 \textbf{ then } s_1\text{; } \textbf{while } a_2 \textbf{ do } s_2 \textbf{ od}$$
$$\textbf{else } s_2 \textbf{ fi; } s_4\text{;}$$

and

$$c(a_i) = c_i, \quad b(s_i) = b_i; \quad d_i(c(a_j)) = d_{ij}.$$

The complexity measures in the above special cases are:

1.1.: $c_1+c_2+c_2+b_1+b_2+b_3$

1.2.: 6

1.3.: 2 (McCabe)

2.1.: $c_1b_1+c_1c_2b_2+c_1b_3+b_4$ (Prather)

3.1.: $d_{11}b_1+d_{12}d_{21}b_2+d_{12}b_3+b_4+d_{11}d_{22}$

L. KOSSUTH UNIVERSITY
DEBRECEN
HUNGARY

L. EÖTVÖS UNIVERSITY
BUDAPEST
HUNGARY

References

[1] T. J. McCabe, A Complexity measure, IEEE Trans. Software Eng. 2. (1976) 308—320
[2] B. Curtis, In search of software complexity, Workshop on Quantitative Software Models for Reliability, Complexity, and Cost. New York, IEEE (1980).
[3] Ronald E. Prather, An axiomatic theory of software complexity measure, The Computer Journal 4 (1984) 340—347.

(Received Jan. 30, 1987)

A noninterleaving semantics for communicating sequential processes: a fixed-point approach

D. V. HUNG, E. KNUTH

Abstract

The paper presents a noninterleaving semantics for Communicating Sequential Processes introduced by Hoare and studied in many works. Concurrency is expressed explicity in the introduced model. Furthermore, semantics of CSP-programs can be obtained by equations in the model. By relating the model to labelled event structures and Petri nets the relationship between CPS and the mentioned models is pointed out.

Key words: CPS programs, concurrency, traces, semiwords, event structures, synchronization.

1. Introduction

In 1978 C. A. R. Hoare introduced in [6] a language for distributed programming called Communicating Sequential Processes — in short CSP. Subsequently, this language has received a great deal of attention. As mentioned in [4], both ADA and OCCAM are based upon CSP.

Many models of semantics for CSP have been proposed. Among them we should mention Hoare's interleaving of strings [7, 8], Gostz's and Reisig's Petri nets with individual tokens [4], Janicki's semitraces [11], Hennessy's (et al.) operational model [5]. In the model of interleaving semantics concurrency is represented by the possibility of shuffling sequences of operations, and thus is not expressed in an explicit form. Furthermore, concurrency is not distinguishable from nondeterminism in this model. Janicki [11] has used Mazuskiewicz's traces to give semantics for CSP, in which concurrency can be distinguished from nondeterminism. The possibility of handling with traces as words makes analyzing properties of CSP-programs in Janicki's model as easy as in Hoare's model. However, as shown by him, traces cannot be used to give semantics for all CPS-programs. The notion of so-called semitraces was introduced by Janicki in order to describe the behaviour of all CPS-programs. Although his semitraces are powerful enough to describe the behaviour of CSP, they have a disadvantage that they cannot be represented by single words.

In our paper [8] we have developed the notion of Mazurkiewicz's traces to a new one based upon the notion of Starke's semiwords. This is the notion of labelled

traces. As pointed out in [8], labelled traces have the same advantage as and are more powerful than traces. They can describe the behaviour of concurrent systems modelled by bounded Petri nets, e.g. producer-consumer systems, while traces cannot.

In the present paper, with the same point of view as in Gorts and Reisig's, Janiki's papers [4], [11], we construct a model of semantics for concurrent systems, more specifically, CSP by using labelled traces. The advantage of Mazurkiewicz's model [14] is taken to this model. By relating the model to Starke's one the relationship between CSP and finite event structures is pointed out. Combining with the results presented in [8], CSP are related to Petri nets as well.

In this paper the basic notion of Mazurkiewicz's traces is used and understood as follows.

For a finite alphabet X, X^* denotes the set of all finite strings over X, ε denotes the empty word, a subset of X^* is called a language over X; a reflexive and symmetric relation on X is called a dependency on X. For a given dependency D on X, let $\equiv_D$ be the least congruence on X^* w.r.t. the concatenation of strings such that $ab \equiv_D ba$ for all $a, b \in X$ with $(a, b) \notin D$. Each equivalence class of $\equiv_D$ is called a trace over D, and a set of traces over D is called a trace language over D.

The paper is organized as follows.

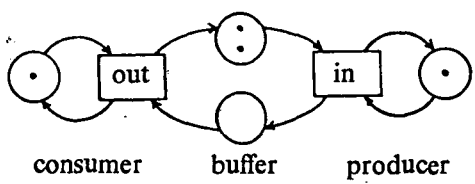
The second section presents a model of semantics for concurrent systems by introducing the notion of labelled traces. The third section is devoted to a study on labelled trace languages. The introduced model is related to other models in the fourth section. A noninterleaving semantics for CSP based upon labelled traces is presented in the fifth one.

2. Labelled trace languages

Let us consider the following problem (bounded buffer [6]):

Construct a buffering process X to smooth variations in the speed of output of portions by a producer process and input by a consumer process. The buffer should contain up to two portions.

A solution of the problem is represented by a 2-bounded Petri net as follows.



Suppose that, at the begining, the buffer is empty so that only the action "in" of the producer can be executed for the first time. After that, the action "out" can occur the first time and the action "in" can occur the second time concurrently. However the first occurrence of "out" depends causally on the first occurrence of "in". Thus, actions "in" and "out" have different cases, and we should take them for atomic actions. For the sake of simplicity sets of atomic actions is assumed to be finite. From the 2-boundedness of the buffer process, every occurrence of one action is concurrent

with not more than one occurrence of the other. Hence, we can use a dependency on a set of four elements to construct a set of labelled partial orderings for describing the behaviour of the above solution.

Basing on the above notice and the theory of Mazurkiewicz's traces we introduce a new description of concurrent systems presented bellow.

All the notions introduced in this section have been presented in our paper [8] in detail. Here, for the aim of the paper we present them in a different form for the sake of convenience.

Let X be a finite alphabet. A finite symmetric relation $D \subseteq (X \times \{1, 2, \ldots\}) \times (X \times \{1, 2, \ldots\})$ such that

a) $(a, i) \in \mathrm{dom}(D) \Rightarrow (a, j) \in \mathrm{dom}(D)$ for all $j \leqq i$, and

b) $((a, j), (a, i)) \in D$ for all $(a, i), (a, j) \in \mathrm{dom}(D)$ will be called a *labelled dependency* over X.

Let D be a labelled dependency over X. Define $\equiv_D$ as the least congruence over $(\mathrm{dom}(D))^*$ (w.r.t. the concatenation) such that $(a, j)(b, i) \equiv_D (b, i)(a, j)$ for all $(a, j), (b, i) \in \mathrm{dom}(D)$ and $((a, j), (b, i)) \notin D$. Each equivalence class of $\equiv_D$ will be called a *labelled trace* over the labelled dependency D, and a set of labelled traces over D will be called a *labelled trace language* over D. Like in the case of traces $[w]_D$ will denote the labelled trace generated by a string $w \in (\mathrm{dom}(D))^*$, and $[L]_D$ will denote the labelled trace language generated by a language $L \subseteq (\mathrm{dom}(D))^*$, i.e.

$$[w]_D := \{v\colon v \in (\mathrm{dom}(D))^* \,\&\, v \equiv_D w\},$$

$$[L]_D := \{[v]_D\colon v \in L\}.$$

For $(a, i) \in \mathrm{dom}(D)$, (a, i) will be called a case of a in D and we denote by $\#(a, D)$ the number $\max\{i\colon (a, i)$ is a case of a in $D\}$. Throughout this paper l always denotes the projection from cases to their first component.

Remark 1. If we identify X with $X \times \{1\}$, each dependency over X (defined by Mazurkiewicz) is a labelled dependency over X. On the other hand each labelled dependency is a special dependency on $X \times \{1, 2, \ldots\}$. Thus, all the notions and results obtained in the theory of traces [1], [13], [14] can be applied to labelled traces and labelled trace languages. This means that we can handle with labelled traces as traces, and the advantage of traces and trace languages is taken to labelled traces and labelled trace languages. The only difference between traces and labelled traces is how the atomic actions are considered.

To show the difference between our notion and Mazurkiewicz's one we consider the dependency graphs of labelled traces.

Definition 1. Let D be a labelled dependency over X, $w \in (\mathrm{dom}(D))^*$. A dependency graph of w (over D) abbreviated a dep-graph of w (over D) and denoted by $D(w)$, is a graph isomorphic to the node-labelled graph (V, E, X, β), defined by:

$$V = \{1, 2, \ldots, n\} \quad \text{if} \quad w = x_1 x_2 \ldots x_n, \quad \beta(i) = l(x_i), \quad \text{and}$$

$$E \subseteq V \times V \quad \text{is such that, for all } 1 \leqq i, j \leqq n, \quad (i, j) \in E$$

if and only if $i < j$ and $(x_i, x_j) \in D$.

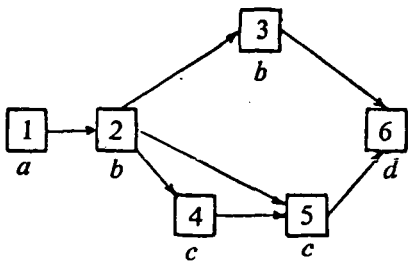
For the sake of convenience by syre (R) we denote the symmetrical and reflexive closure of a binary relation R on dom (R) throughout the paper.

Example 1. Let $X=\{a, b, c, d\}$,

$$D = \text{syre}\big(\{((a, 1), (b, 1)), ((b, 1), (b, 2))((b, 1), (c, 1)), ((b, 2), (d, 1)), ((c, 1), (d, 1))\}\big),$$

$$w = (a, 1)(b,1)(b, 2)(c, 1)(c, 1)(d, 1).$$

$D(w)$ has the following form:



Notice that this node-labelled graph can not be a dep-graph of any trace over any dependency on X. The reason is that the occurrences of c depend on the first occurrence of b, but are concurrent with the second occurrence of b.

It can be seen from the theory of traces that

$$w \equiv_D v \Rightarrow D(w) \cong D(v).$$

(Unlike in the case of dep-graphs of traces, the converse direction is not true!)

Hence, it is reasonable to define a *dep-graph of a labelled trace* t over D as $D(w)$ for any $w \in t$. Not distinguishing labelled traces, dep-graphs of which are isomorphic, we define that labelled traces t and t' over D are *isomorphic* iff $D(t) \cong D(t')$, where $D(u)$ denotes a dep-graph of a labelled trace u over D.

Clearly, each dep-graph of a labelled trace is acyclic, and its transitive closure is a labelled partial ordering over X, which will be called a *labelled partial ordering generated* or *induced* by a labelled trace, and in which any pair of nodes with the same label is ordered. Hence, our notion is related to Starke's one of semiwords (see the section 4).

As mentioned in Remark 1, all the notions of traces are applied to labelled traces. Here we remention some of them, which are needed in the sequel.

Let D be a labelled dependency over X. The *trace concatenation* and *trace iteration* of labelled traces and labelled trace languages are defined by

$[x]_D[y]_D := [xy]_D$ for $x, y \in (\text{dom}\,(D))^*$,

$UV := \{uv \mid u \in U\ v \in V\}$ for labelled trace languages U and V over D.

$U^* := \bigcup_{i=0}^{\infty} U^i$, $U^0 = [\varepsilon]_D$, $U^{i+1} = U^i U$, $i \geqq 0$, for a labelled trace language U over D.

The following lemma is needed for a purpose of constructing operations on labelled trace languages.

Lemma 1. Let D and D' be labelled dependencies on X, $h: (\mathrm{dom}\,(D'))^* \to (\mathrm{dom}\,(D))^*$ be a homomorphism satisfying:

a) $\forall x \in \mathrm{dom}\,(D')$: $h(x) \in \mathrm{dom}\,(D) \cup \{\varepsilon\}$,

b) $\forall x, y \in \mathrm{dom}\,(D')$: $((x, y) \notin D' \,\&\, h(x) \neq \varepsilon, h(y) \neq \varepsilon) \Rightarrow (h(x), h(y)) \notin D$.

Then, for any labelled trace u over D', $w \in u$ we have

$$h(u) \subseteq [h(w)]_D.$$

Proof. Let u be a labelled trace over D' and $w \in u$. Since h is a homomorphism, we have only to prove that if $w = w_1 xyw_2$, $w' = w_1 yxw_2$, $(x, y) \notin D'$, then $h(w) \equiv_D h(w')$. But this is obvious from the specified property of h.

Hence, each mapping $h: (\mathrm{dom}\,(D'))^* \to (\mathrm{dom}\,(D))^*$ satisfying the condition of Lemma 1 can be considered as a homomorphism from a labelled trace language over D' to a labelled trace language over D as well.

We shall adopt Mazurkiewicz's denotation. Let D be a labelled dependency over X.

$$A(D) := \{[a]_D : a \in \mathrm{dom}\,(D)\},$$

$$T(D) := \{[w]_D : w \in (\mathrm{dom}\,(D))^*\}, \quad \text{and}$$

$$P(D) := 2^{T(D)}.$$

Having in mind our intended interpretation, elements of $A(D)$ will be called *actioncases* over D, those of $T(D)$ *processes* over D and those of $P(D)$ *activities* over D. Actioncases a and b occur *concurrently* in a process t if $t = t'[ab]t''$ where $(a, b) \notin D$.

From Remark 1 and the definition of dep-graphs, if we identify X with $X \times \{1\}$, and a word over X with a trace over the dependency $D = (X \times \{1\}) \times (X \times \{1\})$ in the obvious way, we have that each word, each trace, and each labelled trace induce a labelled partial ordering over X. Let $W(X)$, $T(X)$ and $LT(X)$ denote classes of labelled partial orderings induced by words, traces, and labelled traces, respectively, on X. Clearly,

$$W(X) \subsetneq T(X) \subsetneq LT(X).$$

We have introduced cases of actions in order to expand the power of our model comparing to Mazurkiewicz's model of traces. Thus, from the intended meaning, we should not distinguish cases having the same effect in a labelled dependency. Therefore, only reduced labelled dependencies are considered. Formally, we introduce the following notions.

Definition 2. Let D and D' be labelled dependencies on X, T and T' labelled trace languages over D and D', resp.

(i) D and D' are said to be isomorphic, denoted by $D \cong D'$, if there exists a mapping φ from dom (D) onto dom (D') satisfying:

(i) φ preserves cases of actions, i.e. if x is a case of an action a, so is $\varphi(x)$,

(ii) φ preserves the dependence, i.e. $(x, y) \in D$ iff $(\varphi(x), \varphi(y)) \in D'$.

(ii) T and T' are said to be isomorphic, denoted by $T \cong T'$, iff $\forall t \in T$, $\exists t' \in T'$ such that $D(t) \cong D(t')$ and vice versa.

Definition 3. Let D be a labelled dependency on X.

(i) Cases (a, i) and (a, j) are said to be equivalent iff $\forall (b, k) \in \mathrm{dom}\,(D)$: $((b, k), (a, i)) \in D \Leftrightarrow ((b, k), (a, j)) \in D$.

(ii) A labelled dependency D' on X is said to be a reduced version of D iff D and D' are isomorphic and D' is reduced, i.e. for $(a, i), (a, j) \in \mathrm{dom}(D')$ with $i \neq j$ (a, i) and (a, j) are not equivalent.

Proposition 1. Every labelled dependency on X has its reduced version.

Proposition 2. Let D be a labelled dependency on X and T a labelled trace language over D. Assume that D is isomorphic to D' by an isomorphism φ. Then T and $\varphi(T)$ are isomorphic.

The above propositions follow immediately from the definitions 2 and 3.

3. Operations on labelled trace languages (on activities)

In the previous section we have defined some operations on labelled trace languages over a given labelled dependency. Those operations have restricted applications, as pointed out by Janicki [11], since concurrency relations are fixed. To improve upon this shortcomings we define our operations corresponding to ones on concurrent processes from Milner's and Hoare's works [7], [15].

In the sequel, let X be an alphabet, D_i a labelled dependency over X, and let t_i and U_i, respectively, be labelled traces and labelled trace languages over D_i, $i = 1, 2$.

a) *Sequential concatenation and concurrent composition.*

We intend to use the sequential concatenation to represent the fact that a process in U_2 starts only when a process in U_1 has terminated. By the concurrent composition, we shall represent a synchronization of processes corresponding to the synchronization mechanism introduced by Hoare [2], [7], Mazurkiewicz [14], and in our papers [8], [9], [10]. By this operation we want to construct a process t from t_1 and t_2, which behaves like t_1 and t_2, progressing in parallel and simultaneously participating in actions having cases in D_1 and D_2.

Having in mind our attention, we define some operations on labelled dependencies as follows.

Sequential composition of D_1 and D_2, denoted by $S(D_1, D_2)$, is the labelled dependency:

$$S(D_1, D_2) := D_1 \cup \{((a, i + \#(a, D_1)), (b, j + \#(b, D_1))) \mid ((a, i), (b, j)) \in D_2\} \cup$$

$$\cup \mathrm{syre}(\{((a, i), (b, j + \#(b, D_1))) \mid (a, i) \in \mathrm{dom}(D_1), (b, j) \in \mathrm{dom}(D_2)\}).$$

Together with $S(D_1, D_2)$ a mapping s from $T(D_2)$ to $T(S(D_1, D_2))$ is defined by

$$s((a, i)) = (a, i \# (a, D_1)).$$

The definition of s is reasonable by Lemma 1,

Concurrent composition of D_1 and D_2, denoted by $C(D_1, D_2)$ is a labelled dependency defined as follows. Let $Y = l(\mathrm{dom}(D_1)) \cap l(\mathrm{dom}(D_2))$ be the set of actions having cases in both D_1 and D_2. Then,

$$\mathrm{dom}(C(D_1, D_2)) := \{x \mid x \in \mathrm{dom}(D_1) \cup \mathrm{dom}(D_2) \,\&\, l(x) \notin Y\} \cup$$

$$\cup \{(a, i) \mid a \in Y \,\&\, i \leqq \#(a, D_1) \cdot \#(a, D_2)\}.$$

For two positive integers m, n let en (n, m) and rem (n, m) stand for the quotient and remainder of dividing n by m. For $i=1, 2$, define mappings

$$h_i\colon \operatorname{dom}(C(D_1, D_2)) \to \operatorname{dom}(D_i)\cup\{\varepsilon\} \quad \text{as follows.}$$

For $a\notin Y$, $i=1, 2$,

$$h_i((a, j)) = \begin{cases}(a, j) & \text{if } (a, j)\in\operatorname{dom}(D_i),\\ \varepsilon & \text{otherwise;}\end{cases}$$

for $a\in Y$

$$h_1((a, j)) = (a, \operatorname{rem}(j-1, \#(a, D_1))+1),$$

$$h_2((a, j)) = (a, \operatorname{en}(j-1, \#(a, D_1))+1).$$

Now, $C(D_1, D_2)$ is defined by

$C(D_1, D_2)=\{(x, y)|$ there exists an i in $\{1, 2\}$ such that $(h_i(x), h_i(y))\in D_i\}$.

Since h_1 and h_2 satisfy the condition of Lemma 1, h_1 and h_2 can be extended to homomorphisms from $T(C(D_1, D_2))$ to $T(D_1)$ and $T(D_2)$ respectively. h_1 and h_2 will be called the projections associated with $C(D_1, D_2)$.

Now, we are ready to define our operations on labelled trace languages.

Definition 4.

(i) The sequential concatenation $U_1\circ U_2$ of U_1 and U_2 is a labelled trace over $S(D_1, D_2)$ defined by $U_1\circ U_2=U_1 s(U_2)$, where U_1 is considered as a labelled trace language over $S(D_1, D_2)$ and the trace concatenation in the right side is for labelled trace languages over $S(D_1, D_2)$.

(ii) The concurrent composition $U_1\| U_2$ of U_1 and U_2 is a labelled trace language over $C(D_1, D_2)$ defined by:

$$U_1\| U_2 = \{t\in T(C(D_1, D_2))\,|\, h_1(t)\in U_1, h_2(t)\in U_2\}.$$

(iii) Sequential iteration (iteration for short) of U_1, denoted by $U_1^{\otimes}$, is defined by

$$U_1^{\otimes} = ((U_1-\{[\varepsilon]_{D_1}\})\circ(U_1-\{[\varepsilon]_{D_1}\}))^*(U_1\cup\{[\varepsilon]_{S(D_1, D_1)}\}),$$

where $U_1\cup\{[\varepsilon]_{S(D_1, D_2)}\}$ is considered as a labelled trace language over $S(D_1, D_1)$, and the trace iteration and trace concatenation are for labelled trace languages over $S(D_1, D_1)$.

When U_1 and U_2 contain a single element, say $U_1=\{t_1\}$, $U_2=\{t_2\}$, we write $t_1\circ t_2$, $t_1\| t_2$ instead of $\{t_1\}\circ\{t_2\}$, $\{t_1\}\|\{t_2\}$ resp.

Example 2.

(i) Let $D_1=\operatorname{syre}(\{((a, 1), (b, 1))\})$,

$$U_1 = [\operatorname{pref}((b, 1)(a, 1))^*]_{D_1},$$

$$D_2 = \operatorname{syre}(\{((b, 1), (c, 1)), ((b, 2), (c, 2)), ((b, 1), (b, 2)), ((c, 1), (c, 2))\})$$

$$U_2 = [\operatorname{pref}((c, 1)(b, 1)+(c, 2)(b, 2))^*]_{D_2}.$$

Then,

$$C(D_1, D_2) = \text{syre}(\{((a, 1), (b, 1)), ((a, 1), (b, 2))\}) \cup D_2,$$

$$h_1((a, 1)) = (a, 1), \quad h_1((b, 1)) = h_1((b, 2)) = (b, 1),$$

$$h_1((c, 1)) = h_1((c, 2)) = \varepsilon$$

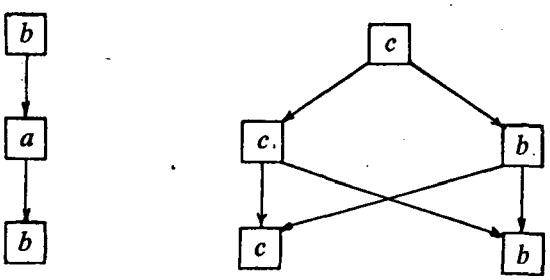
$$h_2(a, 1) = \varepsilon, \quad h_2((b, 1)) = (b, 1), h_2((b, 2)) = (b, 2), h_2((c, 1)) = (c, 1),$$

$$h_2((c, 2)) = (c, 2).$$

Let $t_1 = [(b, 1)(a, 1)(b, 1)]_{D_1} \in U_1$,

$$t_2 = [(c, 1)(b, 1)(c, 2)(b, 2)(c, 1)]_{D_2} \in U_2.$$

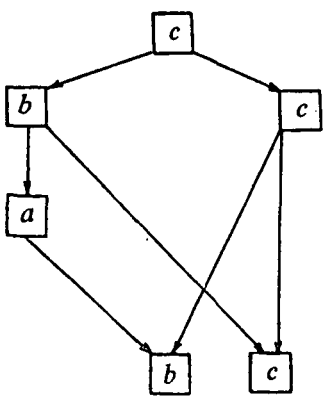
Reduced versions of dep-graphs of t_1 and t_2, resp., are of the form (i.e. transitive arcs are omitted):



By the definition of the concurrent composition, $t_1 \| t_2$ is a labelled trace over $C(D_1, D_2)$

$$t_1 \| t_2 = [(c, 1)(b, 1)(a, 1)(c, 2)(b, 2)(c, 1)]_{C(D_1, D_2)}$$

A reduced version of a dep-graph of $t_1 \| t_2$ is of the form:



It can be seen that

$$U_1 \| U_2 = [\text{pref}((c, 1)(b, 1)(a, 1) + (c, 2)(b, 2)(a, 1))^*]_{C(D_1, D_2)}.$$

We propose in the example that U_1 is the activity of the single portion-buffer, and U_2 is the activity of the two-portion buffer with a and b corresponding to "out" and "in" repectively, in the former, b and c corresponding to "out" and "in" in the latter. Then $U_1 \| U_2$ corresponds to a composition of the two buffers: the two-portion buffer inputs from its producer, then outputs to the single-portion buffer, and in turn, the single-portion buffer outputs to its consumer.

(ii) Let $D = \mathrm{syre}(\{((a, 1), (e, 1)), ((b, 1), (e, 1)), ((e, 1), (c, 1))\})$,

$$U = \{[(a, 1)(b, 1)(e, 1)(a, 1)(c, 1)]_D\}.$$

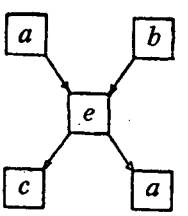
Then

$$S(D, D) = D \cup \{((x, 2), (y, 2)) \mid ((x,1), (y, 1)) \in D\} \cup$$
$$\cup \mathrm{syre}(\{((x,1), (y, 2)) \mid x, y \in l(\mathrm{dom}(D))\}).$$

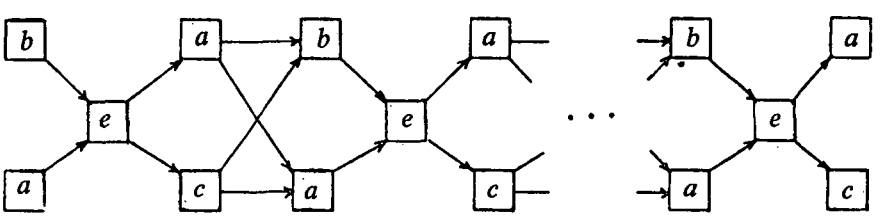
By the definition of the iteration

$$U^{\otimes} = [((a, 1)(b, 1)(e, 1)(a, 1)(c, 1)(a, 2)(b, 2)(e, 2)(a, 2)(c, 2))^*$$
$$(\varepsilon + (a, 1)(b, 1)(e, 1)(a, 1)(c, 1))]_{S(D, D)}.$$

A reduced version of a dep-graph of $t \in U$ has the form:



and reduced versions of dep-graphs of elements of $U^{\otimes}$ are of the form:



In the sequel, for $u \in X^*$, $Y \subseteq X$, by $u|_Y$ we denote the projection of u on Y, i.e. the image of u by the erasing homomorphism from X^* to Y^*. For $u, v \in X^*$ we also write $u|_v$ instead of $u|_{\mathrm{alph}(v)}$ without fear of confusion, where alph (v) denotes the set of symbols forming v.

Proposition 3. $t_1 \| t_2$ contains not more than one element.

Proof. By trivial induction on the length of elements of $T(C(D_1, D_2))$ we can show that if for $t, t' \in T(C(D_1, D_2))$ $h_1(t) = h_1(t')$ and $h_2(t) = h_2(t')$, then $t = t'$.

Proposition 4.

$$t_1 \| t_2 \neq \emptyset \quad \text{if and only if}$$

$$l(t_1)|_{l(\mathrm{dom}(D_2))} \cap l(t_2)|_{l(\mathrm{dom}(D_1))} \neq \emptyset.$$

Proof. The "only if" part is obvious and we prove the "if" part. Suppose that there exists $w \in l(t_1)|_{l(\mathrm{dom}(D_2))} \cap l(t_2)|_{l(\mathrm{dom}(D_1))} \subseteq X^*$. Then there exist $u_1 \in t_1$, $u_2 \in t_2 : l(u_1)|_{l(\mathrm{dom}(D_2))} = l(u_2)|_{l(\mathrm{dom}(D_1))} = u$. Let

$$u_1 = a_1 a_2 \ldots a_n \in (\mathrm{dom}(D_1))^*,$$

$$u_2 = b_1 b_2 \ldots b_m \in (\mathrm{dom}(D_2))^*,$$

$$w = c_1 c_2 \ldots c_k \in Y^* = (l(\mathrm{dom}(D_1)) \cap l(\mathrm{dom}(D_2)))^*.$$

Then, there exist monotonic functions $f_1 : \{1, 2, \ldots, k\} \to \{1, 2, \ldots, n\}$ and $f_2 : \{1, 2, \ldots, k\} \to \{1, 2, \ldots, m\}$ satisfying:

a_j is a case of an element in Y if and only if $j = f_1(i)$ for some $i \leqq k$, $j \leqq n$, and b_j is a case of an element in Y if and only if $j = f_2(i)$ for some $i \leqq k$, $j \leqq m$.

Let $g : \{c_1, c_2, \ldots, c_k\} \to \mathrm{dom}(C(D_1, D_2))$ be defined as follows.

Let $a_{f_1(i)} = (c_i, p)$, $b_{f_2(i)} = (c_i, q)$. Then $g(c_i) = (c_i, s)$, where s is determined from the equation system:

$$p = \mathrm{rem}(s-1, \#(c_i, D_1)) + 1.$$

$$q = \mathrm{en}(s-1, \#(c_i, D_1)) + 1.$$

Let $u_1' = a_1' a_2' \ldots a_n'$, $u_2' = b_1' b_2' \ldots b_m'$ be defined by

$$a_j' = \begin{cases} a_j & \text{if } j \notin f_1(\{1, 2, \ldots, k\}), \\ g(c_j) & \text{if } j = f_1(i), \text{ for } j \leqq n, \end{cases}$$

$$b_j = \begin{cases} b_j & \text{if } j \notin f_2(\{1, 2, \ldots, k\}), \\ g(c_i) & \text{if } j = f_2(i) \text{ for } j \leqq m. \end{cases}$$

Clearly, $u_1'|_{u_2'} = u_2'|_{u_1'} = g(c_1 c_2 \ldots c_k)$.

Hence, by Theorem 2 ([12], pp. 205) there exists $w' \in (\mathrm{dom}(C(D_1, D_2))^*$ such that $w'|_{u_1'} = u_1'$, $w'|_{u_2'} = u_2'$. It is obvious from the definition of g that

$$[w']_{C(D_1, D_2)} = t_1 \| t_2.$$

Proposition 5. Let D_3, D_1, D_2 be labelled dependencies on X, U, U_1, $U_2 \in P(D_1)$, V, V_1, $V_2 \in P(D_2)$, $Z \in P(D_3)$, $W \in P(C(D_1, D_2))$, $t \in T(C(D_1, D_2))$, and h_1, h_2 the homomorphisms associated with $C(D_1, D_2)$. Then

a) $C(D_1, D_2) \cong C(D_2, D_1)$ and $U \| V \cong V \| U$;
b) $C(D_1, C(D_2, D_3)) \cong C(C(D_1, D_2), D_3)$ and $(U \| V) \| Z \cong U \| (V \| Z)$;
c) $U \| \emptyset = \emptyset$;
d) $[\varepsilon]_{D_1} \| [\varepsilon]_{D_2} = [\varepsilon]_{C(D_1, D_2)}$;
e) $(U_1 \cup U_2) \| V = (U_1 \| V) \cup (U_2 \| V)$;
f) $U \| (V_1 \cup V_2) = (U \| V_1) \cup (U \| V_2)$;
g) $(h_1(t)U) \| (h_2(t)V) = t(U_1 \| U_2)$;
h) $W \subseteq h_1(W) \| h_2(W)$.

Proof.

a) For $a\in X$, $0<j\leqq \#(a, D_1)\cdot\#(a, D_2)$ the numbers $i=$ $=\mathrm{rem}\,(j-1,\ \#(a, D_1))+1$, $k=\mathrm{en}\,(j-1,\ \#(a, D_1))+1$ are defined uniquely, and for a pair (i, k) with $i\leqq \#(a, D_1)$ and $k\leqq \#(a, D_2)$ the integer $j'=\#(a, D_2)\times$ $\times(i-1)+k$ is determined uniquely. Thus the correspondence $f_a(j)=j'$ is an one-to-one mapping from $\{1, \ldots, \#(a, D_1)\cdot(a, D_2)\}$ to $\{1, \ldots, \#(a, D_1)\cdot\#(a, D_2)\}$.

By the definitions of $C(D_1, D_2)$, $C(D_2, D_1)$ and h_1, h_2, the mapping h: dom $(C(D_1, D_2))\to$dom $C(D_2, D_1))$ defined by

$$h((a, j)) = \begin{cases}(a, j) & \text{if} \quad \#(a, D_1)\cdot\#(a, D_2) = 0,\\ (a, f_a(j)) & \text{if} \quad \#(a, D_1)\cdot\#(a, D_2) > 0\end{cases}$$

is an one-to-one isomorphism. Furthermore, let h_1' and h_2' be the projections associated with $C(D_2, D_1)$, we have:

$$h_1((a, j)) = h_2'(h(a, j)),\ h_2((a, j)) = h_1'(h(a, j)).$$

Consequently, it follows a).

b) For $a\in X$ with $\#(a, C(D_1, C(D_2, D_3)))>0$ and for $i=1, 2, 3$ let the mappings g_{ia} and g'_{ia} from $\{1, 2, \ldots, \#(a, C(D_1, C(D_2, D_3)))\}$ to $\{1, 2, \ldots, \#(a,D_i)\}$ be defined as follows (g_{ia} and g'_{ia} are undefined if $\#(a, D_i)=0$).

If $\#(a, D)\cdot\#(a, D_2)\cdot\#(a, D_3)>0$, for $j\leqq \#(a, D_1)\cdot\#(a, D_2)\cdot\#(a, D_3)$, let

$$g_{1a}(j) = \mathrm{rem}\,(j-1,\ \#(a, D_1))+1,$$

$$g_{2a}(j) = \mathrm{rem}\,(\mathrm{en}\,(j-1,\ \#(a, D_1)),\ \#(a, D_2))+1,$$

$$g_{3a}(j) = \mathrm{en}(\mathrm{en}\,(j-1,\ \#(a, D_1)),\ \#(a, D_2))+1,$$

$$g'_{1a}(j) = \mathrm{rem}\,(\mathrm{rem}\,(j-1,\ \#(a, D_1)\cdot\#(a, D_2)),\ \#(a, D_1))+1,$$

$$g'_{2a}(j) = \mathrm{en}\,(\mathrm{rem}\,(j-1,\ \#(a, D_1)\cdot\#(a, D_2)),\ \#(a, D_1))+1$$

$$g'_{3a}(j) = \mathrm{en}\,(j'-1,\ \#(a, D_1)\cdot\#(a, D_2))+1.$$

If $\#(a, D_1)\cdot\#(a, D_2)\cdot\#(a, D_3)=0$, for $j\leqq \#(a, C(D_1, C(D_2, D_3)))$, then let

$$g_{1a}(j) = g'_{1a}(j) = \begin{cases}\text{undefined} & \text{if} \quad \#(a, D_1) = 0,\\ \mathrm{rem}\,(j-1,\ \#(a, D_1))+1 & \text{if} \quad \#(a, D_1) > 0,\end{cases}$$

$$g_{2a}(j) = g'_{2a}(j) = \begin{cases}\text{undefined} & \text{if} \quad \#(a, D_2) = 0,\\ \mathrm{en}\,(j-1,\ \#(a, D_1))+1 & \text{if} \quad \#(a, D_1)\cdot\#(a, D_2) > 0,\\ \mathrm{rem}\,(j-1,\ \#(a, D_2))+1 & \text{if} \quad \#(a, D_1) = 0\,\&\,\#(a, D_2) > 0,\end{cases}$$

$$g_{3a}(j) = g'_{3a}(j) = \begin{cases}\text{undefined} & \text{if} \quad \#(a, D_3) = 0,\\ \mathrm{en}\,(j-1,\ \#(a, D_1)+\#(a, D_2)) & \text{if} \quad \#(a, D_3) > 0.\end{cases}$$

Let $f_a(j)=j'$ iff for $i=1, 2, 3$ $g_{ia}(j)=g'_{ia}(j')$.

It can be seen easily that $((a, j), (b, j'))\in C(D_1, C(D_2, D_3))$ ($C(C(D_1, D_2), D_3)$, resp.) if and only if there exists i in $\{1, 2, 3\}$ such that $((a, g_{ia}(j)), (b, g_{ib}(j'))\in$

$\in D_i((a, g'_{ia}(j)), (b, g'_{ib}(j')))\in D_i$, resp.). Hence, the mapping f: dom $(C(D_1, C(D_2, D_3)))\to$ dom $(C(C(D_1, D_2), D_3))$ defined by

$$f((a, j)) = (a, f_a(j))$$

is an isomorphism between $C(D_1, C(D_2, D_3))$ and $C(C(D_1, D_2), D_3)$.

For $i=1, 2, 3$ mappings G_i, G'_i from $T(C(D_1, C(D_2, D_3)))$, $T(C(C(D_1, D_2), D_3))$ resp., to $T(D_i)$ defined by

$$G_i((a, j)) = (a, g_{ia}(j)),$$

$$G'_i((a, j)) = (a, g'_{ia}(j))$$

are homomorphisms by Lemma 1. Furthermore, for $t\in T(C(D_1, C(D_2, D_3)))$ $(T(C(C(D_1, D_2), D_3)),$ resp.) $t\in U\|(V\|Z)$ $((U\|V)\|Z,$ resp.) if and only if $G_1(t)\in U$, $G_2(t)\in V$, $G_3(t)\in Z$ $(G'_1(t)\in U$, $G'_2(t)\in V$, $G'_3(t)\in Z$, resp.). Hence, by Lemma 1 f can be extended to an isomorphism from $T(C(D_1, C(D_2, D_3)))$ to $T(C(C(D_1, D_2), D_3))$ and $f(U\|(V\|Z))=(U\|V)\|Z$. Thus, by Proposition 2, $U\|(V\|Z)\cong(U\|V)\|Z$.

The properties (c)—(h) are obvious.

The following theorem has been formulated by Mazurkiewicz for the case of trace languages. Fortunately, it is still true for labelled trace languages, although our operation of synchronization is more powerful and general than his one.

Theorem 1. The concurrent composition $\|$ is the least function from $P(D_1)\times$ $\times P(D_2)$ to $P(C(D_1, D_2))$ (w.r.t. the inclusion ordering of its values) meeting the following conditions:

(a) $$(h_1(x)U)\|(h_2(x))V) = x(U\|V),$$

(b) $$(U_1\cup U_2)\|V = (U_1\|V)\cup(U_2\|V),$$

(c) $$U\|(V_1\cup V_2) = (U\|V_1)\cup(U\|V_2),$$

(d) $$[\varepsilon]_{D_1}\|[\varepsilon]_{D_2} = [\varepsilon]_{C(D_1, D_2)},$$

for each actioncase x in $C(D_1, D_2)$, U, U_1, $U_2\in P(D_1)$, V, V_1, $V_2\in P(D_2)$.

The proof of the theorem is similar to the proof of Theorem 1, [14], pp. 352 and is omitted here.

b) *Union and intersection.*

We deal with the construction of activities from activities over different labelled dependencies. The union is intended for the nondeterministic choice and the intersection is intended to represent the tied synchronization.

Let $N(D_1, D_2)$ be the labelled dependency defined by

$$N(D_1, D_2) = D_1\cup\{((a, i+\#(a, D_1)), (b, j+\#(b, D_1)))\,|\,((a, i), (b, j))\in D_2\}\cup$$

$$\cup \text{syre}(\{(a, j), (a, i)+\#(a, D_1)\,|\,(a, j)\in\text{dom}(D_1), (a, i)\in\text{dom}(D_2)\}).$$

A mapping s: $T(D_2)\to T(N(D_1, D_2))$ associated to $N(D_1, D_2)$ is defined by

$$s((a, i)) = (a, i+\#(a, D_1)), \quad (a, i)\in\text{dom}(D_2).$$

Let $I(D_1, D_2)$ be the labelled dependency defined by $I(D_1, D_2) = C(D_1, D_2) \cap \cap (Y \times \{1, 2, \ldots\})^2$, where $Y = l(D_1) \cap l(D_2)$. Since $C(D_1, D_2) \cap (\mathrm{dom}\,(I(D_1, D_2)))^2 = I(D_1, D_2)$ each labelled trace over $I(D_1, D_2)$ is a labelled trace over $C(D_1, D_2)$. Let h_1, h_2 be homomorphisms associated with $C(D_1, D_2)$.

Definition 5. (i) A nondeterminic composition $U_1 \square U_2$ of U_1 and U_2 is a labelled trace language over $N(D_1, D_2)$ defined by

$U_1 \square U_2 = U_1 \cup s(U_2)$, where the operation $\cup$ on the right hand side is for labelled trace languages over $N(D_1, D_2)$ with considering U_1 as a labelled trace language over $N(D_1, D_2)$.

(ii) The intersection $U_1 \sqcap U_2$ of U_1 and U_2 is a labelled trace language over $I(D_1, D_2)$ defined by

$$U_1 \sqcap U_2 = \{t \in T(I(D_1, D_2)) \mid h_1(t) \in U_1, h_2(t) \in U_2\}.$$

Proposition 6. For $U', U'' \in P(D_1)$, $V' \in P(D_2)$,

a) $U' \square U'' \cong U' \cup U''$; $N(D_1, D_1) \cong D_1$,

b) $U' \| V' = U' \sqcap V'$ if $D_1 = D_2$.

This follows immediately from Definition 3.

Proposition 7. Let D_1, D_2, D_3, D be labelled dependenceis on X, $U \in P(D)$, $V \in P(D_1)$, $W \in P(D_2)$, $Z \in P(D_3)$, $t_1 \in T(D_1)$, $t_2 \in T(D_2)$. Then

a) $[\varepsilon]_{D_1} \circ U \cong U \circ [\varepsilon]_{D_1} \cong U$,

b) $(U \circ V) \circ W = U \circ (V \circ W)$,

c) $U \circ (V \square W) \cong (U \circ V) \square (U \circ W)$, $(V \square W) \circ U \cong (V \circ U) \square (W \circ U)$,

d) $t_1 \circ U \circ t_2 \cong t_1 \circ V \circ t_2 \Rightarrow U \cong V$.

Proof. a), b) and d) are obvious. To prove c) consider a mapping h from $\mathrm{dom}\,(N(S(D, D_1), S(D, D_2)))$ onto $\mathrm{dom}\,(S(D, N(D_1, D_2)))$ defined by:

$$h((a, j)) = \begin{cases} (a, j), & \text{if } j \leqq \#(a, D) + \#(a, D_1), \\ (a, j - \#(a, D)), & \text{otherwise.} \end{cases}$$

By the definition of the operations S, N on labelled dependencies, $((a, i), (b, j)) \in N(S(D, D_1), S(D, D_2))$ iff $(h((a, i)), h((b, j)) \in S(D, N(D_1, D_2))$. Hence, h can be extended to a homomorphism from $T(N(S(D, D_1), S(D, D_2)))$ to $T(S(D, N(D_1, D_2)))$ in the obvious way (by Lemma 1). Furthermore, it can be seen easily that $h((U \circ V) \square (U \circ W)) = U \circ (V \square W)$. By Proposition 2, $U \circ (V \square W) \cong (U \circ V) \square (U \circ W)$. The remaining case of c) is proved similarly.

4. Relations to other models

As mentioned in the section 2 each labelled trace induces a lebelled partial ordering over X, and each labelled partial ordering over X is a finite labelled event structure over X ([16]). Thus, a labelled trace language over a labelled dependency on X is a set of labelled event structures having a very simple representative: a (finite) labelled dependency and a word language (may be represented by a regular expression). In our paper [8] we have related labelled trace languages to Petri nets and some

interesting results have been obtained. In this section, we relate labelled trace languages to semilanguages introduced by Starke [18], [19].

Definition 6 ([19] pp. 337).

(i) A *labelled partial ordering* (lpo for short) over X is a triple (A, S, β), where (A, S) is a irreflexive partial ordering, $\beta: A \to X$ is a labelling mapping.

(ii) Two lpo's (A, S, β) and (A', S', β') are said to be isomorphic iff there exists a bijection b from A onto A' which preserves the labelling and the ordering:

$$aSc \Leftrightarrow b(a)S'b(c) \& \beta(a) = \beta'(b(a)).$$

The isomorphy class $[(A, S, \beta)]$ of a finite lpo (A, S, β), i.e. the class of all lpo's which are isomorphic with (A, S, β) is called a *partial word* over X. A partial word $[(A, S, \beta)]$ over X such that for all a, b from A

$$\beta(b) = \beta(a) \Rightarrow aSb \vee bSa \vee a = b, \tag{1}$$

i.e. where all the sets $\beta^{-1}(x)$ (for $x \in X$) are chains w.r.t. S is called a *semiword* over X.

Let $p(t)$ denote a partial word over X induced by a labelled trace t over a labelled dependency on X (see section 2) i.e. $p(t) = [(A, S, \beta)]$ where (A, S, β) is the labelled partial ordering induced by t.

Theorem 2. For each labelled trace t over a labelled dependency on X, $p(t)$ is a semiword over X.

Proof. Let $D(t)$ be a dep-graph of t, where t is a labelled trace over a labelled dependency D on X. By the definition of D, if x, y are cases of an action $a \in X$, $(x, y) \in D$. Thus, the labelled partial ordering over X induced by t satisfies (1). Consequently, $p(t)$ is a semiword over X.

It follows from Theorem 2 that every labelled trace language over a labelled dependency on X generates a semilanguage over X in the natural way.

For $U \in P(D)$, denote by

$$SL(U) = \{p(t) | t \in U\}.$$

$SL(U)$ is called *semilanguage generated* by U.

Theorem 3. Let $U \in P(D_1)$, $V \in P(D_2)$, where D_1, D_2 are labelled dependencies on X. Then

$$SL(U^{\otimes} \circ V) = SL(U \circ U^{\otimes} \circ Y \square Y).$$

Proof.

By (iii) of Definition 4

$$U^{\otimes} = ((U - \{[\varepsilon]_{D_1}\}) \circ (U - \{[\varepsilon]_{D_1}\}))^* (U \cup \{[\varepsilon]_{S(D_1, D_1)}\}) \in P(S(D_1, D_1)),$$

where $U \cup \{[\varepsilon]_{S(D_1, D_1)}\}$ is considered as a labelled trace language over $S(D_1, D_1)$. It follows from the definition of $S(D_1, D_1)$ and of the sequential concatenation that for $t \in T(S(D_1, D_1))$, $p(t) \in U^{\otimes}$ if and only if there exist $t_1, t_2, \ldots, t_n \in U - \{[\varepsilon]_{D_1}\}$ such that

(i) $D(t) = \emptyset$ iff $n = 0$, and

(ii) Let $D(t_1), \ldots, D(t_n)$ be dep-graphs of $t_1, \ldots, t_n$ over D_1, $D(t_i) = (V_i, E_i, X_i, \beta_i)$, $i = 1, 2, \ldots, n$, $V_i \cap V_j = \emptyset$ for $i \neq j$, $i, j \leqq n$.

Then $D(t) \cong (\bigcup_{i=1}^{n} V_i, E, X, \beta)$, where

$$E = (\bigcup_{i=1}^{n} E_i) \cup \{(a, b) \mid a \in V_i, b \in V_{i+1}, i \leqq n-1\}, \beta|_{V_i} = \beta_i.$$

Hence, since $V_i \neq \emptyset$ for $i \leqq n$, for $i < j \leqq n$, $a \in V_i$, $b \in V_j$, (a, b) is an arc of the transitive closure of $D(t)$. From the definition of the sequential concatenation of labelled trace languages it follows that

$$SL(U \circ U^{\circledast}) \cup \{p([\varepsilon]_{D_1})\} = SL(U^{\circledast}),$$

$$SL(U^{\circledast} \circ V) = SL(U \circ U^{\circledast} \circ V) \cup SL(V).$$

By the definition of the operation $\square$ our theorem is obtained from the last equality.

To relate labelled trace languages to interleavings of strings we recall the synchronization mechanism introduced by Hoare [7], E. Knuth [12].

Definition 7 ([7]). Let $L_1, L_2 \subseteq X^*$, $Y \subseteq X$. The synchronized parallel composition $L_1 \|_Y L_2$ is the set $\bigcup_{\substack{w_1 \in L_1 \\ w_2 \in L_2}} w_1 \|_Y w_2$, where $w_1 \|_Y w_2$ denotes the set of all successful interleavings of w_1 and w_2 with synchronising communications in Y and is defined inductively as follows:

(i) $\varepsilon \|_Y \varepsilon = \{\varepsilon\}$

(ii) $aw \|_Y \varepsilon = \varepsilon \|_Y aw = \begin{cases} \emptyset & \text{if } a \in Y \\ a(w \|_Y \varepsilon) & \text{if } a \notin Y, \end{cases}$

(iii) $aw \|_Y bw' = bw' \|_Y aw = \begin{cases} a(w \|_Y w') & \text{if } a = b \in Y \\ \emptyset & \text{if } a \neq b \wedge a, b \in Y \\ a(w \|_Y bw') & \text{if } a \notin Y \wedge b \in Y \\ a(w \|_Y bw') \cup b(aw \|_Y w') & \text{if } a \notin Y, b \notin Y. \end{cases}$

Theorem 4. For a labelled trace language $U \in P(D)$ let $\text{inter}(U) = \bigcup_{t \in U} l(t)$. Then, for $U \in P(D_1)$, $V \in P(D_2)$, (where D_1, D_2 are labelled dependencies) and $Y = l(\text{dom } D_1) \cap l(\text{dom } (D_2))$,

a) $\text{inter } (U \circ V) = \text{inter } (U) \text{ inter } (V)$,

b) $\text{inter } (U \| V) = \text{inter } (U) \|_Y \text{ inter } (V)$,

c) $\text{inter } (U^{\circledast}) = (\text{inter } (U))^*$,

d) $\text{inter } (U \square V) = \text{inter } (U) \cup \text{inter } (V)$,

e) $\text{inter } (U \sqcap V) = \text{inter } (U) \cap \text{inter } (V)$.

Proof. a) and c), d) are obvious. e) follows from b).

b) is proved as follows.

Let h_1 and h_2 be the projections associated with $C(D_1, D_2)$. Clearly, for $t \in U \| V$ $h_1(t) \in U$, $h_2(t) \in V$, and $l(t)|_{l(\text{dom } (D_2))} = l(h_2(t)) \subseteq \text{inter } (V)$, $l(t)|_{l(\text{dom } (D_1))} = l(h_1(t)) \subseteq \subseteq \text{inter } (U)$. Thus, $\text{inter } (U \| V) \subseteq \text{inter } (U) \|_Y \text{ inter } (V)$.

Let $y \in \text{inter}(U) \|_Y \text{inter}(V)$. By Definition 7, $y|_{l(\text{dom}(D_1))} \in \text{inter}(U)$, $y|_{l(\text{dom}(D_2))} \in \text{inter}(V)$. There exist $u \in U$, $v \in V$ such that $y|_{l(\text{dom}(D_1))} \in l(u)$, $y|_{l(\text{dom}(D_2))} \in l(v)$. By Proposition 4, $t = u \| v$ is defined.

It follows easily from the definition of h_1 and h_2 that $y \in l(t)$. This completes the proof of the theorem.

In the sequel, for simplicity of denotation, if L_1 and L_2 are considered over fixed alphabets, say Σ_1 and Σ_2, and $Y = \Sigma_1 \cap \Sigma_2$, we shall writte $L_1 \| L_2$ instead of $L_1 \|_Y L_2$.

Proposition 8. Let L_1, L_2, L_3 are languages over $\Sigma_1, \Sigma_2. \Sigma_3$ respectively. Then

$$L_1 \| L_2 = L_2 \| L_1, \quad \text{and} \quad (L_1 \| L_2) \| L_3 = L_1 \| (L_2 \| L_3).$$

Proof. Straightforward from the definition of the operation $\|$.

5. Labelled trace languages as a noninterleaving semantics for CSP

The notion of CSP presented in this paper is at an abstract level necessary for our purpose.

Let *Comm* be a finite set of actions. A process P over *Comm* is in one of the following forms:

$$P := P_1; P_2; \ldots; P_n,$$

$$P := [P_1 \| P_2 \| \ldots \| P_n],$$

$$P := \underline{\otimes} P_1,$$

$$P := [P_1 \square P_2 \square \ldots \square P_n],$$

$$P := a \to P_1, a \in Comm,$$

$$P := \textbf{skip}, (\textbf{skip} \notin Comm),$$

$$P := P_1 \setminus \{b_1, b_2, \ldots, b_n\},$$

where $P_1, P_2, \ldots, P_n$ are processes over *Comm*.

The meaning of the above constructions of processes is given informally as follows.

$P_1; P_2; \ldots; P_n$ specifies sequential excution of $P_1, P_2, \ldots, P_n$ in the order written (process by process, P_{i+1} starts only P_i has terminated, $1 \leqq i \leqq n-1$), and starts with the start of P_1, terminates with the termination of P_k.

$[P_1 \| P_2 \| \ldots \| P_n]$ specifies concurrent excution of its constituent processes. They all start simultaneously and the process $P = [P_1 \| \ldots \| P_n]$ terminates successfully only if and when they have all successfully terminated. The relative speed with which they are excuted is arbitrary. The set of actions excuted by each of them is required to be disjoint from those executed by the rest. $P_1, P_2, \ldots, P_n$ are synchronized by the actions intended. P_i excutes an action intended to synchronize with P_j (in the construction) if and only if P_j excutes a corresponding action (intended to synchronize P_j with P_i) simultaneously (see [6], [7], [11]).

$\underline{\otimes} P$ specifies as many iterations as necessary of P sequentially.

$[P_1 \square P_2 \square \ldots \square P_n]$ specifies excution of exactly one of its consituent processes and the choice between them is fully nondeterministic, cannot be influenced by the environment.

$a \rightarrow P_1$ specifies excution of the action a followed by excution of P_1.

Skip specifies the process having no effect and never fails.

Now, we identify the action intended to synchronize P_i with P_j with a corresponding action intended to synchronize P_j with P_i in a construct $[P_1 \| P_2 \| \ldots \| P_n]$. We can suppose that the set of actions excuted by P_i may not be disjoint from the one by P_j and the actions in their intersection require that P_i and P_j must excute each of them simultaneously. (This abstraction has been made by Hoare in [7], [2], Janicki in [11]).

The interleaving semantics for CSP given by Hoare [2], [7] is a follows.

Each process over *Comm* is identified with a subset of *Comm** called its interleaving semantics:

$$\mathbf{skip} := \{\varepsilon\},$$

$$a \rightarrow P := aP,$$

$$P_1;\ P_2;\ \ldots;\ P_n := P_1 P_2 \ldots P_n,$$

$$[P_1 \square P_2 \square \ldots \square P_n] := P_1 \cup P_2 \cup \ldots \cup P_n,$$

$$[P_1 \| P_2 \| \ldots \| P_n] := P_1 \| P_2 \| \ldots \| P_n, \quad \text{where}$$

the operation $\|$ on languages is defined in the previous section and $P_1, \ldots, P_n$ are considered as languages over alph $(P_1), \ldots,$ alph (P_n) respectively. (Here for a language L, alph (L) denotes the smallest alphabet, over which L is a language).

$$\underline{\otimes} P := P^*,$$

$$P \setminus \{b_1, \ldots, b_n\} := P|_{\text{alph}(P_1) \setminus \{b_1, \ldots, b_n\}},$$

where $P|_A$ denotes the projection of P on A^*.

Because of the presence of the hidding operation in CSP, to relate our model to CSP we have to extend the notion of labelled trace languages.

An ε-labelled dependency on X is a symmetric relation $D_\varepsilon \subseteq ((X \cup \{\varepsilon\}) \times \{1, 2, \ldots\})^2$ satisfying:

(i) $(a, i) \in \text{dom}(D_\varepsilon) \Rightarrow (a, j) \in \text{dom}(D_\varepsilon)$ for $j \leqq i$,

(ii) $((a, i), (a, j)) \in D_\varepsilon$ for $(a, i), (a, j) \in \text{dom}(D_\varepsilon)$ and $a \neq \varepsilon$.

An ε-labelled dependency on X may not be reflexive in its domain. However, this has no effect in the definition of trace languages and the notion of trace languages is extended to this case. Then, a trace language over D_ε is called an ε-labelled trace language over D_ε. All the notions and the results presented in the previous sections are valid for ε-labelled trace languages as well with the only exception that the set Y in the definition of the operation $\|$ of labelled trace languages is modified as

$$Y = (l(\text{dom}(D_1)) \cap l(\text{dom}(D_2))) \setminus \{\varepsilon\}.$$

ε-labelled trace semantics proposed for CSP is presented bellow. Each process over *Comm* is identified with an ε-labelled trace language over an ε-labelled dependency on *Comm* as follows:

$$a \to P := \{[(a, 1)]_{\{((a,1),(a,1))\}} \circ P,$$

$$P_1; P_2; \ldots; P_n := P_1 \circ P_2 \circ \ldots \circ P_n;$$

$$[P_1 \| P_2 \| \ldots \| P_n] := P_1 \| P_2 \| \ldots \| P_n;$$

$$[P_1 \square \ldots \square P_n] := P_1 \square P_2 \square \ldots \square P_n;$$

$$\underline{\otimes} P := P^{\otimes}$$

$$P \setminus \{b_1. \ldots, b_n\} := h_{\{b_1, \ldots, b_n\}}(P),$$

where $h_{\{b_1, \ldots, b_n\}}$ is defined as follows: For an ε-labelled dependency D_ε on X let

$$h_{\{b_1, \ldots, b_n\}}((a, i)) = \begin{cases} (a, i) & \text{if} \quad a \notin \{b_1, \ldots, b_n\} \wedge (a, i) \in \mathrm{dom}\,(D_\varepsilon) \\ (\varepsilon, i) & \text{if} \quad a \in \{b_1, \ldots, b_n\} \wedge (a, i)\, \mathrm{dom}\,(D_\varepsilon), \end{cases}$$

$$D_\varepsilon\{b_1, \ldots, b_n\} = \{(h_{\{b_1, \ldots, b_n\}}(x), h_{\{b_1, \ldots, b_n\}}(y)) | (x, y) \in D_\varepsilon\}.$$

By Lemma 1, $h_{\{b_1, \ldots, b_n\}}$ is considered as a homomorphism from $T(D_\varepsilon)$ to $T(D_\varepsilon | \{b_1, \ldots, b_n\})$.

The correspondence between ε-labelled trace semantics and interleaving semantics for CSP is stated by the following theorem, which follows immediately from Theorem 4.

Theorem 5. For a process P over *Comm*. Let $LT(P)$, $INTER\ (P)$ denote the ε-labelled trace semantics, interleaving semantics, respectively, for P. Then

$$\mathrm{inter}\,(LT(P)) = INTER\,(P).$$

Proposition 9. If a process P over *Comm* does not contain a construction $[P_1 \square \ldots \square P_n]$, $LT(P)$ contains, at most, one element.

The Proposition follows from Proposition 3.

6. Conclusion

We have presented an extention of the theory of traces as an attempt to provide a mathematical description for the behaviour of concurrent systems, more specifically, CSP. Labelled trace languages have been shown to be more powerful than trace languages and to have a simple representation.

However, the construction of the theory of CSP based upon labelled trace languages requires a deeper study on labelled trace languages concluding a construction of domains of the operations on processes so that the operations are continuous and the representation of the properties of processes in its semantics in the model. This will be presented in our future work.

COMPUTER AND AUTOMATION INSTITUTE
HUNGARIAN ACADEMY OF SCIENCES
VICTOR HUGO U. 18—22
1123 BUDAPEST
HUNGARY

References

[1] I. J. AALBERSBERG and G. ROZENBERG, Theory of traces, Institute of Applied Mathematics and Computer Science, Univ. of Leiden, The Netherlands, Rep. 85—16, August 1985.

[2] S. D. BROOKES, C. A. R. HOARE, A. W. ROSCOE, A theory of communicating sequential processes, J. of ACM, Vol. 31, N. 3, July 1984, pp. 560—599.

[3] P. DEGANO and U. MONTANARI, Distributed Systems, partial orderings of events, and event structures, Proc. of the International Summer School "Control Flow and Data Flow: Concepts of Distributed Programming", NATO ASI Series, M. Broy, Ed., Vol. 38F14, Springer-Verlag, 1985, pp. 7—106.

[4] U. GOLTZ and W. REISIG, CSP-programs as nets with individual tokens, LNCS, Springer-Verlag, Vol. 188, 1985, pp. 169—196.

[5] M. HENNESSY, W. LI, G. PLOTKIN, "A first attempt at translating CSP into CCS", Proc. of the 2nd International Conference on Distributed Computing, *IEEE*, N. 81, CH 1591—7, Paris 1981.

[6] C. A. R. HOARE, Communicationg sequential processes, *Comm. of ACM*, Vol. 21, N. 8, 1978, pp. 666—677.

[7] C. A. R. HOARE, Specification-oriented semantics for communicating processes, Acta Informatica, Vol. 23 Springer-Verlag, 1986, pp. 9—66.

[8] D. V. HUNG and E. KNUTH, Labelled trace languages and Petri nets, Working Paper, MTA-SZTAKI.

[9] D. V. HUNG, Notes on trace languages, Projections and synthesized computation systems, Közlemények, MTA-SZTAKI, Vol. 32, 1985, pp. 87—104.

[10] D. V. HUNG and M. SZIJJÁRTÓ, Synchronized parallel composition of languages, Proc. of Conference of Automata, Languages and Programming Systems, Salgótarján, Hung. May 1986.

[11] R. JANICKI, Trace Semantics for Communicating Sequential Processes, Institute for Elektroniske Systemer, Danmark, R—85—12, 1985.

[12] E. KNUTH, GY. GYŐRY and L. RÓNYAI, *A Study of the projection operation*, Application and Theory of Petri Nets, W. Reisig, Ed. Springer-Verlag, Vol. 52, 1982.

[13] A. MAZURKIEWICZ, Concurrent Program Schemes and Their Interpretations, DAIMI PB—78, Aarhus Univ., Press, 1977.

[14] A. MAZURKIEWICZ, *Semantics of concurrent systems: a modular fixed-point trace approach*, LNCS, Vol. 188, 1985, pp. 353—375.

[15] R. MILNER, Calculi for synchrony and asynchrony, *TCS*, Vol. 25, 1983, pp. 267—310.

[16] M. NIELSEN, G. PLOTKIN, and F. WINSKEL, Petri nets, event structures and domains, Part I, *TCS*, Vol. 13, 1981, pp. 85—108.

[17] C. A. PETRI, Non-sequential processes, GMD—ISF Report, ISF—77—05, 1977, pp. 1—24.

[18] P. H. STARKE, Processes in Petri nets, *Informationverarb u. Kybernet.* EIK, Vol. 17, 1981, pp. 389—416.

[19] P. H. STARKE, *Traces and semiwords*, LNCS, Computation Theory, Andrezej, Ed., Fifth. Symposium, Vol. 208, 1985, pp. 332—350.

(Received June 20, 1987)

TAPSOFT '87 Proceedings of the International Joint Conference on Theory and Practice of Software Development Pisa, Italy, March 1987.

Volume 1: Advanced Seminar on Foundations of Innovative Software Development I and Colloquium on Trees in Algebra and Programming (CAAP'87) (Lecture Notes in Computer Science Vol. 249) XIV+289 pages, Springer Verlag, Berlin—Heidelberg—New York—Tokio, 1987. Edited by Hartmut Ehrig, Robert Kowalski, Giorgio Levi and Ugo Montanari.

Volume 2: Advanced Seminar on Foundations of Innovative Software Development II and Colloquium on Functional and Logic Programming and Specifications (CFLP) (Lecture Notes in Computer Science Vol. 250) XIV+336 pages, Springer Verlag, Berlin—Heidelberg—New York—Tokio, 1987. Edited by Hartmut Ehrig, Robert Kowalski, Giorgio Levi and Ugo Montanari.

These two books contain a selected collection of papers presented at TAPSOFT '87 held in Pisa, Italy, March 1987.

TAPSOFT '87 consists of three parts:

i, Advanced Seminar on Foundations of Innovative Software Development concerns new directions in software development on the basis of recent technological and theoretical advances.

ii, Colloquium on Trees in Algebra and Programming covers the formal aspect and properties of trees, and more generally, combinatorial and algebraic structures in all fields of Computer Science. Besides the customary topics, CAAP includes contributions related to specifications, communicating systems and type theory.

iii, Colloquium on Functional and Logic Programming and Specifications focuses on those aspects of Functional and Logic Programming which are most important in innovative software development.

Contents of Volume 1

Contents of Volume 2

These well edited interesting volumes present the state of the art in theory and practice of software development. It is recommended for those people interested in the latest results of the field.

S. Vágvölgyi

A SZERKESZTŐ BIZOTTSÁG CÍME:

6720 SZEGED
SOMOGYI U. 7.

EDITORIAL OFFICE:

6720 SZEGED
SOMOGYI U. 7.
HUNGARY

Information for authors

Acta Cybernetica publishes only original papers in the field of computer sciences mainly in English, but also in French, German or Russian. Authors should submit two copies of manuscripts to the Editorial Board. The manuscript must be typed double-spaced on one side of the paper only. Footnotes should be avoided and the number of figures should be as small as possible. For the form of references, see one of the articles previously published in the journal. A list of special symbols used in the manuscript should be supplied by the authors.

A galley proof will be sent to the authors. The first-named author will receive 50 reprints free of charge.

INDEX — TARTALOM

ISSN 0324—721 X

Felelős szerkesztő és kiadó: Gécseg Ferenc
A kézirat a nyomdába érkezett: 1987 augusztus
Terjedelem: 7,7 (A/5) ív
Készült monószedéssel, íves magasnyomással
az MSZ 6601 és az MSZ 5602—55 szabvány szerint
87-4083 — Szegedi Nyomda — Felelős vezető: Surányi Tibor igazgató