58725

Tomus 5. Fasciculus 4.

ACTA CYBERNETICA

FORUM CENTRALE PUBLICATIONUM CYBERNETICARUM HUNGARICUM

Szeged, 1982
Curat: Universitas Szegediensis de Attila József nominata

5. kötet 4. füzet

ACTA CYBERNETICA

A HAZAI KIBERNETIKAI KUTATÁSOK
KÖZPONTI PUBLIKÁCIÓS FÓRUMA

Szeged, 1982. június

A Szegedi József Attila Tudományegyetem gondozásában

Tomus 5.

ACTA CYBERNETICA

FORUM CENTRALE PUBLICATIONUM
CYBERNETICARUM HUNGARICUM

Szeged, 1982

Curat: Universitas Szegediensis de Attila József nominata

INDEX

Tomus 5.

81- 5569— Szegedi Nyomda — Felelős vezető: Dobó József igazgató

On the role of blocking in rewriting systems

By H. C. M. KLEIJN*, G. ROZENBERG*, R. VERRAEDT**

Introduction

A rewriting system G generates a set of sentential froms $\underline{\text{sent}}\, G$ (see, e.g., [9]). If G is "pure" (see. e.g., [5]), i.e. it does not use nonterminals, then $\underline{\text{sent}}\, G$ forms also the language of G, denoted $L(G)$. In this sense every sentential form of G is successful. If G is not pure, i.e. it uses nonterminals, then the language of G consists of only those sentential forms that do not contain nonterminal symbols. In this case a sentential form is (potentially) successful if it can be rewritten (perhaps in a number of steps) into an element of $L(G)$.

Thus, naturally, $\underline{\text{sent}}\, G$ gets divided into "blocking" and "nonblocking" (hence successful) sentential forms.

The possibility of having blocking sentential forms in a grammar is often useful. In a particular derivation of a word w, G may "guess" a property of a sentential form currently rewritten and if the guess was incorrect G will take care of the fact that the derivation is dead-ended. This is a typical way of programming a language through a context-sensitive grammar (see, e.g., [9]). Also the synchronization mechanism in E(T)OL systems (see for example [7] and [8]) is a typical example of the use of a blocking mechanism.

In this paper we investigate the role that this blocking mechanism plays in rewriting systems. In particular, we do this for the grammars of the Chomsky hierarchy (Section II), EOL systems (Section III) and ETOL systems (Section IV).

I. Preliminaries and basic definitions

We assume the reader to be familiar with the rudiments of formal language theory as, e.g., in the scope of [7] and [9]. In order to fix our notation we recall some basic notions now.

For a word x, $|x|$ denotes its length and $\underline{\text{alph}}\, x$ denotes the set of letters occurring in x. For a language K, $\underline{\text{alph}}\, K = \bigcup_{x \in K} \underline{\text{alph}}\, x$. The empty word is denoted by Λ.

Let Σ_1 and Σ be alphabets, such that $\Sigma_1 \subseteq \Sigma$. Then the homomorphism $\underline{\text{Pres}}_{\Sigma,\Sigma_1}$ from Σ^* into Σ_1^* is defined as follows. If $a \in \Sigma_1$, then $\underline{\text{Pres}}_{\Sigma,\Sigma_1} a = a$ and if $a \in \Sigma \setminus \Sigma_1$, then $\underline{\text{Pres}}_{\Sigma,\Sigma_1} a = \Lambda$. To avoid cumbersome notation we often write $\underline{\text{Pres}}_{\Sigma_1}$ instead of $\underline{\text{Pres}}_{\Sigma,\Sigma_1}$, whenever Σ is understood from the context.

The mapping $\underline{\text{mir}}$ from Σ^* into Σ^* is defined by: if $w=xy$, with $x\in\Sigma^*$ and $y\in\Sigma$, then $\underline{\text{mir}}\, w=y\, \underline{\text{mir}}\, x$; $\underline{\text{mir}}\, \Lambda=\Lambda$.

Definition I.1. (i) A *grammar* is an ordered quadruple $G=(V, \Sigma, P, S)$, where V is a finite non-empty alphabet, the *total alphabet of* G, $\Sigma\subset V$ is the *terminal alphabet of* G, $V\setminus\Sigma$ is the *nonterminal alphabet of* G, $S\in V\setminus\Sigma$ is the *axiom of* G and P is a finite subset of $V^*(V\setminus\Sigma)V^*\times V^*$; the elements of P are called the *productions of* G and for $(\alpha, \beta)\in P$ we write $\alpha\to\beta$.

(ii) A word $v\in V^*$ *directly derives* a word $w\in V^*$ *according to* G, denoted $v\underset{G}{\Rightarrow}w$, if there are $x, y, \alpha, \beta\in V^*$ such that $v=x\alpha y$, $w=x\beta y$ and $\alpha\to\beta$ is a production of G. We write $x\overset{0}{\underset{G}{\Rightarrow}}x$ for every $x\in V^*$ and for $n\geqq 1$, $x\overset{n}{\underset{G}{\Rightarrow}}y$ if for some $z\in V^*$, $x\overset{n-1}{\underset{G}{\Rightarrow}}z\underset{G}{\Rightarrow}y$. We write $x\overset{+}{\underset{G}{\Rightarrow}}y$ $(x\overset{*}{\underset{G}{\Rightarrow}}y,\ x\overset{\leqq m}{\underset{G}{\Rightarrow}}y$, respectively) if $x\overset{t}{\underset{G}{\Rightarrow}}y$ for some integer $t>0$ $(t\geqq 0,\ t\leqq m$, respectively). If no confusion is possible we use, $\Rightarrow$, $\overset{+}{\Rightarrow}, \overset{*}{\Rightarrow}, \overset{n}{\Rightarrow}, \overset{\leqq n}{\Rightarrow}$ rather than $\underset{G}{\Rightarrow}, \overset{+}{\underset{G}{\Rightarrow}}, \overset{*}{\underset{G}{\Rightarrow}}, \overset{n}{\underset{G}{\Rightarrow}}, \overset{\leqq n}{\underset{G}{\Rightarrow}}$.

(iii) The set of *sentential froms* of G, denoted $\underline{\text{sent}}\, G$, is defined by $\underline{\text{sent}}\, G=\{w\in V^*: S\overset{*}{\underset{G}{\Rightarrow}}w\}$.

(iv) The *language of* G, denoted $L(G)$ is defined by $L(G)=\{w\in\Sigma^*: S\overset{*}{\underset{G}{\Rightarrow}}w\}=\underline{\text{sent}}\, G\cap\Sigma^*$.

Definition I.2. Let $G=(V, \Sigma, P, S)$ be a grammar.

(i) G is termed *regular*, if $\alpha\to\beta\in P$ implies $\alpha\in V\setminus\Sigma$ and $\beta\in\Sigma(V\setminus\Sigma)$ or $\beta\in\Sigma$.

(ii) G is termed *context-free,* if $\alpha\to\beta\in P$ implies $\alpha\in V\setminus\Sigma$ and $\beta\in V^+$.

(iii) G is termed *context-sensitive (monotonic)* if $\alpha\to\beta\in P$ implies $|\alpha|\leqq|\beta|$.

The families of languages generated by regular, context-free, context-sensitive and arbitrary grammars will be denoted by $\mathcal{L}(\text{Reg})$, $\mathcal{L}(\text{CF})$, $\mathcal{L}(\text{CS})$ and $\mathcal{L}(\text{RE})$ respectively.

Definition I.3. (i) An ETOL *system* is an ordered quadruple $H=(V, \Sigma, \mathcal{P}, \omega)$, where V, Σ and $V\setminus\Sigma$ are as in the definition of a grammar, $\omega\in V^+$ is the *axiom of* H and $\mathcal{P}$ is a finite non-empty set of *tables* $P_1, \ldots, P_n$, $n\geqq 1$. A table P_i, $1\leqq i\leqq n$, is a finite subset of $V\times V^*$, such that for each $\alpha\in V$ there exists a $\beta\in V^*$ with $(\alpha, \beta)\in P_i$. An element (α, β) of P_i, $1\leqq i\leqq n$, is called *a-production* and is usually written as $\alpha\to\beta\cdot\alpha\to\beta$ is called an *α-production* and the fact that $\alpha\to\beta$ belongs to P_i, $1\leqq i\leqq n$, respectively to $\mathcal{P}$, is often abbreviated as $\alpha\underset{P_i}{\to}\beta$, respectively $\alpha\underset{\mathcal{P}}{\to}\beta$.

(ii) A word $v\in V^*$ *directly derives* a word $u\in V^*$ *according to* H, denoted $v\underset{H}{\Rightarrow}u$, if $v=\alpha_1\ldots\alpha_k$, $\alpha_i\in V$ for $1\leqq i\leqq k$, $u=\beta_1\ldots\beta_k$, $\beta_i\in V^*$ for $1\leqq i\leqq k$, and there exists a $j\in\{1, \ldots, n\}$ such that $\alpha_i\underset{P_j}{\to}\beta_i$, for all $i\in\{1, \ldots, n\}$. We write $x\overset{0}{\underset{H}{\Rightarrow}}x$ for every $x\in V^*$ and for $n\geqq 1$, $x\overset{n}{\underset{H}{\Rightarrow}}y$ if for some $z\in V^*$, $x\overset{n-1}{\underset{H}{\Rightarrow}}z\underset{H}{\Rightarrow}y$. We write

$x \overset{+}{\underset{H}{\Rightarrow}} y \left(x \overset{*}{\underset{H}{\Rightarrow}} y,\ x \overset{\leq m}{\underset{H}{\Rightarrow}} y,\right.$ respectively) if $x \overset{t}{\underset{H}{\Rightarrow}} y$ for some integer $t>0\,(t\geq 0, t\leq m,$ respectively). If no confusion is possible we use $\Rightarrow, \overset{+}{\Rightarrow}, \overset{*}{\Rightarrow}, \overset{n}{\Rightarrow}, \overset{\leq n}{\Rightarrow}$ rather than $\underset{H}{\Rightarrow}, \overset{+}{\underset{H}{\Rightarrow}}, \overset{*}{\underset{H}{\Rightarrow}}, \overset{n}{\underset{H}{\Rightarrow}}, \overset{\leq n}{\underset{H}{\Rightarrow}}$.

(iii) The set of *sentential forms of H*, denoted $\underline{\text{sent}}\,H$, is defined by $\underline{\text{sent}}\,H = \{v \in V^*\colon \omega \overset{*}{\underset{H}{\Rightarrow}} v\}$.

(iv) The *language of H*, denoted $L(H)$ is defined by $L(H) = \{v \in \Sigma^*\colon \omega \overset{*}{\underset{H}{\Rightarrow}} v\} = \underline{\text{sent}}\,H \cap \Sigma^*$.

Definition I.4. Let $H=(V, \Sigma, \mathcal{P}, \omega)$ be an ETOL system, with $\mathcal{P} = \{P_1, \ldots, P_n\}$.

(i) If $\mathcal{P}$ consists of one table only, say $\mathcal{P}=\{P\}$, then H is termed an EOL *system* and denoted $H=(V, \Sigma, P, \omega)$.

(ii) If, for every $\alpha \underset{\mathcal{P}}{\rightarrow} \beta$, $\beta \in V^+$, then H is termed a *propagating* ETOL system, denoted EPTOL system.

(iii) If for all $i \in \{1, \ldots, n\}$, $\alpha \underset{P_i}{\rightarrow} \beta$ and $\alpha \underset{P_i}{\rightarrow} \gamma$ implies $\beta=\gamma$, then H is termed a *deterministic* ETOL system, denoted EDTOL system.

(iv) If $\Sigma=V$, then H is termed a TOL *system*.

From the above definition it follows that we consider OL, POL, DOL, PDOL, TOL, PTOL, DTOL, PDTOL, EOL, EPOL, EDOL, EPDOL, ETOL, EPTOL, EDTOL and EPDTOL systems. The family of languages generated by X systems, where X stands for one of the above mentioned abbreviations, will be denoted by $\mathcal{L}(X)$.

Let H be an ETOL system. If the sequence $D=(x_0, x_1, \ldots, x_n)$ is such that $x_i \underset{H}{\Rightarrow} x_{i+1}$, $0 \leq i < n$, then each occurrence of a letter in every word from $x_0, \ldots, x_{n-1}$ has a unique contribution to x_n. If A is an occurrence of a letter in x_i, $0 \leq i < n$, then we use $\underline{\text{ctr}}_{D, x_i} A$ to denote this contribution.

Two languages, L_1 and L_2, are considered to be equal if $L_1 \cup \{\Lambda\} = L_2 \cup \{\Lambda\}$. We consider two families of languages, $\mathcal{L}_1$ and $\mathcal{L}_2$, to be equal if they differ at most by $\{\Lambda\}$. Two language generating devices G and H are said to be *equivalent* if $L(G)=L(H)$.

Definition I.5. Let $H=(V, \Sigma, P, \omega)$ be an EOL system. If there exists a subset $\Phi \subseteq V \setminus \Sigma$ such that for all $\alpha \in \Sigma \cup \Phi$, $\alpha \underset{P}{\rightarrow} \beta$ implies $\beta \in \Phi^+$, then H is called a *synchronized* EOL system, abbreviated sEOL system. Φ is called the *set of synchronization symbols of H.*

The following result is well known, see, e.g., [3].

Lemma I.1. For every EOL system, there exists an equivalent sEOL system.

The following is the central notion of this paper.

Definition I.6. (i) A grammar $G=(V, \Sigma, P, S)$ is *nonblocking* if for every word $v \in \underline{\text{sent}}\,G$ there exists a word $u \in \Sigma^*$, such that $v \overset{*}{\underset{G}{\Rightarrow}} u$.

(ii) An ETOL system $H=(V, \Sigma, \mathscr{P}, \omega)$ is *nonblocking* if for every word $v \in \underline{\text{sent}}\, H$ there exists a word $u \in \Sigma^*$, such that $v \overset{*}{\underset{H}{\Rightarrow}} u$.

REMARK. Note that if G is a nonblocking grammar or a nonblocking ETOL system, then either $L(G) \setminus \{\Lambda\} \neq \emptyset$ or $S \overset{+}{\underset{G}{\Rightarrow}} \Lambda$ and $L(G)=\{\Lambda\}$.

The families of languages generated by nonblocking regular, nonblocking context-free, nonblocking context-sensitive, nonblocking arbitrary grammars or by nonblocking X systems (where X stands for ETOL or one of its subclasses) will be denoted by $\mathscr{L}$(nbReg), $\mathscr{L}$(nbCF), $\mathscr{L}$(nbCS), $\mathscr{L}$(nbRE) and $\mathscr{L}(\text{nb}X)$, respectively.

Lemma I.2. If $X \in \{\text{Reg, CF, CS, RE}\}$ or X stands for ETOL or one of its subclasses, then $\mathscr{L}(\text{nb}X) \subseteq \mathscr{L}(X)$.

II. The Chomsky hierarchy

In this section we impose the nonblocking condition on regular, context-free, context-sensitive and arbitrary grammars.

We start by recalling a well known fact concerning the first two types of grammars.

Lemma II.1. For every context-free (regular) grammar generating a non-empty language, there exists an equivalent nonblocking context-free (regular) grammar.

Proof. Since for every context-free (regular) grammar, there exists an equivalent context-free (regular) grammar in which every nonterminal is useful (see, e.g., [9], otherwise the generated language is empty) the lemma holds. □

Thus we get the following result.

Theorem II.1. (i) $\mathscr{L}(\text{nbReg})=\mathscr{L}(\text{Reg})$.
(ii) $\mathscr{L}(\text{nbCF})=\mathscr{L}(\text{CF})$.

For context-sensitive grammars generating non-empty languages we have a similar situation. However, the proof is much more involved. For this reason we give only an intuitive description of the proof. For a formal, detailed proof, we refer the interested reader to the Appendix.

Lemma II.2. For every context-sensitive grammar, generating a non-empty language there exists an equivalent nonblocking context-sensitive grammar.

Proof. Let $K \subseteq \Sigma^*$ be a non-empty language, generated by a context-sensitive grammar. We distinguish two cases.

(i) K is finite. Then, obviously, the context-sensitive grammar $(\Sigma \cup \{S\}, \Sigma, P, S)$ with $P=\{S \to x : x \in K\}$ is nonblocking and generates K.

(ii) K is infinite. Let $\Sigma'=\{[a, b, c, d] : a, b, c, d \in \Sigma\} \cup \{[a, b, c] : a, b, c \in \Sigma\} \cup \{[a, b] : a, b \in \Sigma\} \cup \{[a] : a \in \Sigma\}$; let h be the homomorphism from Σ'^* into Σ^* defined by $h([a, b, c, d])=abcd$, $h([a, b, c])=abc$, $h([a, b])=ab$ and $h([a])=a$. Let

$$K' = \{[a_1, a_2, a_3, a_4]\ldots[a_{4n-3}, a_{4n-2}, a_{4n-1}, a_{4n}]:\ n \geqq 2,\ a_1 \ldots a_{4n} \in K\} \cup$$
$$\{[a_1, a_2, a_3, a_4]\ldots[a_{4n-3}, a_{4n-2}, a_{4n-1}, a_{4n}][a_{4n+1}]:\ n \geqq 2,\ a_1 \ldots a_{4n+1} \in K\} \cup$$
$$\{[a_1, a_2, a_3, a_4]\ldots[a_{4n-3}, a_{4n-2}, a_{4n-1}, a_{4n}][a_{4n+1}, a_{4n+2}]:\ n \geqq 2,\ a_1 \ldots a_{4n+2} \in K\} \cup$$
$$\{[a_1, a_2, a_3, a_4]\ldots[a_{4n-3}, a_{4n-2}, a_{4n-1}, a_{4n}][a_{4n+1}, a_{4n+2}, a_{4n+3}]:\ n \geqq 2,\ a_1 \ldots a_{4n+3} \in K\}.$$

Clearly K' is context-sensitive, say it is generated by a context-sensitive grammar $G' = (V', \Sigma', P', S')$. Moreover $h(K') = K \setminus \{x \in K: |x| < 8\}$. Now we can construct a nonblocking context-sensitive grammar $G = (V, \Sigma, P, S)$ generating K. It works as follows.

(1) $S \to x$ is in P for $x \in K$ with $|x| < 8$.

(2) $P' \subseteq P$.

(3) S directly derives S' surrounded by markers. Hence K' can be derived, surrounded by these markers. A successful derivation in G terminates by rewriting elements of Σ' into elements of Σ (after it was checked by markers that a current sentential form consists of letters from Σ') and making the markers disappear. (The deletion of markers and rewriting symbols of Σ' into symbols of Σ is paired together so that the monotonicity of the productions is guaranteed).

(4) From the above it follows that $K \subseteq L(G)$.

(5) At any stage in the derivation process of a word from K' (modulo markers) a "dead" symbol N can be introduced. Then all symbols (except the leftmost and rightmost marker) in the current sentential form can (and will) eventually be replaced by N; to the right of the rightmost marker (which now also changes into N) the axiom S' of G', surrounded by markers, will be introduced again. This process may be repeated an arbitrary number of times.

(6) If from S' a word w of K' is derived, then termination can take place if w is long enough (K' is infinite!) to "absorb" all dead symbols and markers, when the symbols of Σ' are rewritten into symbols of Σ. Again, during this termination process, there still is a possibility to change all symbols of the current sentential forms into N's and to place S', surrounded by markers to the right of this string. In this case the derivation process "switches" again into state (5).

(7) Now (5) and (6) imply that $L(G) \subseteq K$, G is nonblocking and monotonic. This together with (4) implies the result. □

Corollary II.1. For every arbitrary grammar, generating a non-empty language, there exists an equivalent nonblocking grammar.

Thus we have the following result.

Theorem II.2. (i) $\mathscr{L}(\text{nbCS}) = \mathscr{L}(\text{CS})$.

(ii) $\mathscr{L}(\text{nbRE}) = \mathscr{L}(\text{RE})$.

Although it follows from Lemma II.2 that for any context-sensitive grammar, generating a non-empty language, there exists an equivalent nonblocking context-sensitive grammar, the proof of this fact was not effective; it is well known that it is not effectively decidable whether or not the language generated by a context-sensitive grammar is finite (see, e.g., [9]). Moreover, there is no algorithm which, given an arbitrary context-sensitive grammar G (generating a non-empty language) yields an equivalent nonblocking context-sensitive grammar. We also show that it is undecidable whether or not an arbitrary context-sensitive grammar G itself is nonblocking.

We prove the above two statements using Post's Correspondence Problem (see, e.g., [9]).

Definition II.1. An *instance of Post's Correspondence Problem* over an alphabet Σ is a pair (A, B), where $A=\{\alpha_1, \ldots, \alpha_n\}$, $B=\{\beta_1, \ldots, \beta_n\}$, $n \geqq 1$ with $\alpha_i \in \Sigma^+$ and $\beta_i \in \Sigma^+$, for $1 \leqq i \leqq n$. (A, B) is said to have a *solution* if there exists a non-empty finite sequence of indices $\{i_1, \ldots, i_k\}$, $i_j \in \{1, \ldots, n\}$ for $1 \leqq j \leqq k$, such that $\alpha_{i_1} \ldots \alpha_{i_k} = \beta_{i_1} \ldots \beta_{i_k}$.

Theorem II.3. There is no algorithm to decide whether or not an arbitrary instance of Post's Correspondence Problem over a two letter alphabet has a solution.

Theorem II.4. There is no algorithm that given an arbitrary context-sensitive grammar generating a non-empty language constructs an equivalent nonblocking context-sensitive grammar.

Proof. Let (A, B) be an arbitrary instance of Post's Correspondence Problem, $A=\{\alpha_1, \ldots, \alpha_n\}$ and $B=\{\beta_1, \ldots, \beta_n\}$, with $n \geqq 1$, $\alpha_i \in \{a, b\}^+$ and $\beta_i \in \{a, b\}^+$, for $1 \leqq i \leqq n$. The context-sensitive grammar G is defined as follows. $G=(V, \{c, d\}, P, S)$, where $V=\{S, Z, a, b, \overleftarrow{M}, \overrightarrow{M}, \overrightarrow{M}_a, \overrightarrow{M}_b, \overleftarrow{M}_a, \overleftarrow{M}_b, Q, N, c, d\}$ and P is given in (1) through (9).

(1) $S \rightarrow c$.
(2) $S \rightarrow c\alpha_i Z \,\underline{\text{mir}}\, \beta_i c$, for $1 \leqq i \leqq n$, and $Z \rightarrow \alpha_i Z \,\underline{\text{mir}}\, \beta_i$, for $1 \leqq i \leqq n$.
(3) $Z \rightarrow \overleftarrow{M} d$.
(4) $\alpha\overleftarrow{M} \rightarrow \overleftarrow{M}\alpha$, for $\alpha \in \{a, b, d\}$, and $c\overleftarrow{M} \rightarrow c\overrightarrow{M}$.
(5) $\overrightarrow{M}\alpha \rightarrow c\overrightarrow{M}_\alpha$, for $\alpha \in \{a, b\}$, and $\overrightarrow{M}d \rightarrow dQ$.
(6) $\overrightarrow{M}_\alpha\beta \rightarrow \beta\overrightarrow{M}_\alpha$, for $\beta \in \{a, b, d\}$, and $\overrightarrow{M}_\alpha c \rightarrow \overleftarrow{M}_\alpha c$, for $\alpha \in \{a, b\}$.
(7) $\alpha\overleftarrow{M}_\alpha \rightarrow \overleftarrow{M}c$, for $\alpha \in \{a, b\}$.
(8) $\beta\overleftarrow{M}_\alpha \rightarrow Nc$, for $\alpha, \beta \in \{a, b\}$ and $\alpha \neq \beta$.
(9) $Q\alpha \rightarrow Nc$, for $\alpha \in \{a, b\}$ and $Qc \rightarrow cc$.

It is rather easy to see that $L(G)=\{c\}$ if (A, B) has no solution and that $L(G)$ is infinite otherwise.

Assume that we could effectively construct an equivalent nonblocking grammar $G'=(V', \{c, d\}, P', S')$ for G. Let $n_0=\min\{|w| : S' \overset{*}{\underset{G'}{\Rightarrow}} w \text{ and } |w| \geqq 2\}$. Obviously we can effectively decide whether or not n_0 exists because G' is monotonic. Since G' is nonblocking, if n_0 exists then $L(G')=L(G)$ contains a word of length at least two and so (A, B) has a solution. If n_0 does not exist, then $L(G')=L(G)=\{c\}$ and hence (A, B) has no solution.

Hence if the algorithm in question exists then Post's Correspondence Problem is decidable; this contradicts Theorem II.3. □

Theorem II.5. It is undecidable whether or not an arbitrary context-sensitive grammar generating a non-empty language is nonblocking.

Proof. Let (A, B) be as in the proof of Theorem II.4. Let $H=(V, \{c, d\}, P'', S)$ be the context-sensitive grammar which is defined as follows. V and S are as in the grammar $G=(V, \{c, d\}, P, S)$ defined in the proof of Theorem II.4. P'' is defined by (1) through (8) as stated there and additionally by:

(9′) $Q\alpha \to \alpha Q$ and $\alpha Qc \to Ncc$ for $\alpha \in \{a, b\}$ and

(10) $\alpha N \to Nc$, for $\alpha \in \{a, b\}$, $dN \to Nd$ and $cN \to cc$.

Hence $L(H) \neq \emptyset$ $(c \in L(H))$ and H is nonblocking if and only if (A, B) has no solution.

Thus, if we would have an effective decision procedure for the nonblocking property of context-sensitive grammars, then Post's Correspondence Problem would be decidable. This contradicts Theorem II.3. □

We conclude this section with the following observations.

For an arbitrary grammar generating a non-empty language, there exists an effective procedure to construct an equivalent nonblocking grammar. This is a consequence of the possibility of using length-decreasing productions for the markers and the dead symbols (as used in the proof of Lemma II.2). Hence we do not need arbitrarily large words to "absorb" all those garbage symbols. Consequently, it is not needed anymore to distinguish between the case of a finite and the case of an infinite language (which made the proof of Lemma II.2 ineffective).

It is well known that it is not decidable whether an arbitrary context-sensitive grammar generates the empty language (see, e.g. [9]). Consequently it is not decidable whether or not an arbitrary context-sensitive grammar has an equivalent nonblocking context-sensitive grammar. Note that in the case of context-free grammars these questions are decidable: finiteness and emptiness are decidable for those grammars.

III. Systems without tables

We will now investigate the effect that the nonblocking condition has on the language generating power of E(P)(D)OL systems.

First we compare EOL and nbEOL systems.

It turns out that the nonblocking restriction is a real restriction. This result should be compared with the results of the previous section.

Lemma III.1. $\mathscr{L}(\text{EPOL}) \setminus \mathscr{L}(\text{nbEOL}) \neq \emptyset$.

Proof. We will prove that $K=\{a^3\} \cup \{a^{2^n}: n \geqq 0\} \in \mathscr{L}(\text{EPOL}) \setminus \mathscr{L}(\text{nbEOL})$.

(i) Let G be the EPOL system which is defined by

$$G = (\{S, A, N, a\}, \{a\}, \{S \to a^3, S \to A, A \to AA, A \to a, a \to N, N \to N\}, S).$$

Obviously $L(G)=K$. Thus $K \in \mathscr{L}(\text{EPOL})$.

(ii) The fact that $K \notin \mathscr{L}(\text{nbEOL})$ is proved by a contradiction. Assume that $K \in \mathscr{L}(\text{nbEOL})$. Then there exists a nbEOL system $H=(V, \Sigma, P, \omega)$ such that $L(H)=K$ or $L(H)=K \cup \{\Lambda\}$.

Since H is nonblocking for every $v \in K$, $v \overset{*}{\Rightarrow} v' \in a^*$ holds. Since H is an EOL system, it must be that $v \overset{+}{\Rightarrow} v' \in a^*$ holds for all $v \in K$.

In particular $a^3 \overset{+}{\Rightarrow} a^k$, for some $k \in \{0, 3\} \cup \{2^n: n \geqq 0\}$.

(1) Assume that $a^3 \overset{+}{\Rightarrow} \Lambda$. Hence $a \overset{+}{\Rightarrow} \Lambda$. Then for each $\alpha \in V$ such that $\alpha \overset{+}{\Rightarrow} x \in a^+$ it holds that $\alpha \overset{\leq t}{\Rightarrow} \Lambda$ where t equals the cardinality of V. Choose r such that $2^r > \max(\{j: \alpha \overset{\leq t}{\Rightarrow} a^j, \alpha \in V\} \cup \{0, 3\})$. Thus $a^{2^{r+1}} \in L(H)$ and by the choice of r we may write $\omega \overset{*}{\Rightarrow} x_1 \alpha x_2 \overset{t}{\Rightarrow} y_1 z y_2 = a^{2^{r+1}}$ such that $\alpha \in V, x_1 x_2 \in V^+, y_1 y_2 \in a^+$, $\alpha \overset{t}{\Rightarrow} z$ and $1 \leqq |z| < 2^n$. On the other hand we have $\omega \overset{*}{\Rightarrow} x_1 \alpha x_2 \overset{t}{\Rightarrow} y_1 y_2 \in a^+$ and $2^{r+1} - 2^r = 2^r < |y_1 y_2| < 2^{r+1}$; a contradiction.

(2) Assume that $a^3 \overset{+}{\Rightarrow} a^3$. Hence there exists a t such that $a \overset{t}{\Rightarrow} a$. Consider the t^{th} speed up $\bar{H}$ of H, $L(\bar{H}) = L(H)$. (See, e.g., [7]). Hence $\bar{H}$ must have a production $a \rightarrow a$. This implies $L(\bar{H}) \in \mathscr{L}(\text{CF})$ (see. e.g., [7]); a contradiction.

(3) Assume that $a_3 \overset{+}{\Rightarrow} a^{2^n}$. If $n \leqq 1$, then $a \overset{+}{\Rightarrow} \Lambda$ which yields a contradiction as in (1). Hence $n \geqq 2$. This implies that $a \overset{+}{\Rightarrow} a^i$ for some $i > 1$. Hence $a^3 \overset{+}{\Rightarrow} a^{3i} \notin$ $\notin K \cup \{\Lambda\}$; a contradiction. □

It follows from the above that there are EOL languages that are not nbEOL languages. However the following theorem demonstrates that there is only a "small difference" between nbEOL and EOL languages.

Theorem III.1. Let $K \in \mathscr{L}(\text{EOL})$ and let § be a symbol, $§ \notin \underline{\text{alph}}\, K$. Then $K \cup §^+ \in \mathscr{L}(\text{nbEPOL})$.

Proof. Let K and § be as in the statement of the theorem. Let $G = (V, \Sigma, P, S)$ be an sEPOL system such that $§ \notin V$, $S \in V \setminus \Sigma$ and $L(G) = K$. Moreover assume without loss of generality that N is the synchronization symbol of $G, \alpha \underset{P}{\rightarrow} N$ for each $\alpha \in V$, and $\alpha \rightarrow N$ is the only α-production for $\alpha \in \Sigma \cup \{N\}$. Then let $\bar{G} = (\bar{V}, \bar{\Sigma}, \bar{P}, S)$ be the EPOL system which is defined as follows.

(i) $W = \{[p]: p \in P\}$, $W \cap (V \cup \{§\}) = \emptyset$, and $\bar{V} = V \cup W \cup \{§\}$.

(ii) $\bar{\Sigma} = \Sigma \cup \{§\}$.

(iii) $\bar{P} = \{\alpha \rightarrow [p]: p = \alpha \underset{P}{\rightarrow} x\} \cup \{[p] \rightarrow x: p = \alpha \underset{P}{\rightarrow} x\} \cup \{\alpha \rightarrow §: \alpha \in V\} \cup \{§ \rightarrow N, § \rightarrow NN\}$.

(1) We first show that $L(\bar{G}) = K \cup §^+$. Let $x \in L(\bar{G})$ and let $D: S \underset{\bar{G}}{\Rightarrow} x_1 \underset{\bar{G}}{\Rightarrow} x_2 \underset{\bar{G}}{\Rightarrow} \dots \underset{\bar{G}}{\Rightarrow} x_n = x \in \bar{\Sigma}^+$ be a derivation in $\bar{G}$. If $x \in \Sigma^+$, then clearly n is even and all productions used in D belong to $\{\alpha \rightarrow [p]: p = \alpha \underset{P}{\rightarrow} x\} \cup \{[p] \rightarrow x: p = \alpha \underset{P}{\rightarrow} x\}$. Hence $D': S \underset{G}{\Rightarrow} x_2 \underset{G}{\Rightarrow} x_4 \underset{G}{\Rightarrow} \dots \underset{G}{\Rightarrow} x_n = x$ is a derivation in G and thus $x \in K$. If $§ \in \underline{\text{alph}}\, x$, n must be odd and consequently (the form of $\bar{P}$ implies that) $x \in §^+$. Thus $L(\bar{G}) \subseteq K \cup §^+$. Since each derivation step in G can be simulated in two steps in $\bar{G}$, $K \subseteq L(\bar{G})$. Moreover $S \underset{\bar{G}}{\Rightarrow} § \underset{\bar{G}}{\Rightarrow} N^2 \underset{\bar{G}}{\Rightarrow} §^2 \underset{\bar{G}}{\Rightarrow} N^3 \underset{\bar{G}}{\Rightarrow} \dots$, yields $§^+ \subseteq L(\bar{G})$. Thus $K \cup §^+ \subseteq L(\bar{G})$. Hence $L(\bar{G}) = K \cup §^+$.

(2) Next we show that $\bar{G}$ is nonblocking. Let $x \in \underline{\text{sent}}\, \bar{G}$. A close inspection of $\bar{P}$ yields that either $x \in V^+$ or $x \in (W \cup \{§\})^+$. If $x \in (W \cup \{§\})^+$ then $x \underset{\bar{G}}{\Rightarrow} y \in V^+$.

If $x \in V^+$ and $|x|=k$ then $x \underset{\bar{G}}{\Rightarrow} \S^k$. Thus $x \overset{\leqq 2}{\underset{\bar{G}}{\Rightarrow}} z \in \S^+$ for all $x \in \underline{\text{sent}}\, \bar{G}$. Hence $\bar{G}$ is nonblocking. □

We now turn to the comparison of the language families $\mathcal{L}(\text{E}X\text{OL})$, $\mathcal{L}(\text{nbE}X\text{OL})$, $\mathcal{L}(X\text{OL})$ where X denotes either P, D, PD or the empty word. We need the following lemmas.

Lemma III.2. (i) $\mathcal{L}(\text{EDOL}) \subseteq \mathcal{L}(\text{nbEOL})$, and
(ii) $\mathcal{L}(\text{EPDOL}) \subseteq \mathcal{L}(\text{nbEPOL})$.

Proof. (i) Our first observation is that every EDOL system generating an infinite language can be considered as an nbEOL system. Every finite non-empty language K with $\underline{\text{alph}}\, K = \Sigma$ can be generated by a nbEOL system, namely $G = (\{S\} \cup \Sigma, \Sigma, \{S \to x \colon x \in K\} \cup \{\alpha \to \alpha \colon \alpha \in \Sigma\}, S)$.
The two observations from the above conclude the proof of (i).
(ii) Analogous to (i). □

Lemma III.3. $\mathcal{L}(\text{DOL}) \setminus \mathcal{L}(\text{nbEPOL}) \neq \emptyset$.

Proof. We will prove that $K = \{ab\} \cup \{a^{2^n}bc \colon n \geqq 1\} \in \mathcal{L}(\text{DOL}) \setminus \mathcal{L}(\text{nbEPOL})$.
(i) Let G be the DOL system which is defined by $G = (\{a, b, c\}, \{a, b, c\}, \{a \to a^2, b \to bc, c \to \Lambda\}, ab)$. Obviously $L(G) = K$. Thus $K \in \mathcal{L}(\text{DOL})$.
(ii) The fact that $K \notin \mathcal{L}(\text{nbEPOL})$ is proved by a contradiction. Assume that $K \in \mathcal{L}(\text{nbEPOL})$. Then $K = L(H)$ for an nbEPOL system $H = (V, \Sigma, P, \omega)$. Since H is nonblocking, for each $v \in K$, $v \overset{+}{\underset{H}{\Rightarrow}} v' \in K$. Thus $a^2bc \overset{t}{\underset{H}{\Rightarrow}} x \in K$ for a positive integer t. Since H is propagating, $|x| \geqq 4$. Moreover x cannot equal a^2bc because this would imply that K is context-free. Thus $a^2bc \overset{t}{\underset{H}{\Rightarrow}} a^{2^n}bc$ for an $n \geqq 2$. Clearly $a \overset{t}{\underset{H}{\Rightarrow}} y$ implies $y \in a^+$, thus $a \overset{t}{\underset{H}{\Rightarrow}} a^i$ for an $i > 0$. $b \overset{t}{\underset{H}{\Rightarrow}} a^k b$ ($b \overset{t}{\underset{H}{\Rightarrow}} a^k$ respectively), $k > 0$ is impossible because then $ab \overset{t}{\underset{H}{\Rightarrow}} a^{i+k}b$ ($ab \overset{t}{\underset{H}{\Rightarrow}} a^{i+k}$ respectively) which contradicts the fact that $L(H) = K$. Hence we must have $a \overset{t}{\underset{H}{\Rightarrow}} a^i$, $i > 1$ and $b \overset{t}{\underset{H}{\Rightarrow}} b$. But then $ab \overset{t}{\underset{H}{\Rightarrow}} a^i b$ which again contradicts the fact that $L(H) = K$. Thus $K \notin \mathcal{L}(\text{nbEPOL})$.
Then (i) and (ii) yield the lemma. □

Lemma III.4. $\mathcal{L}(\text{POL}) \setminus \mathcal{L}(\text{EDOL}) \neq \emptyset$.

Proof. Let $K = \{a^n \colon n \geqq 1\}$. It is proved in [6] that $K \in \mathcal{L}(\text{POL}) \setminus \mathcal{L}(\text{EDOL})$. □

Lemma III.5. $\mathcal{L}(\text{EPDOL}) \setminus \mathcal{L}(\text{nbEDOL}) \neq \emptyset$.

Proof. We will prove that $K = \{a^2b^2, b^4(ac)^2\} \in \mathcal{L}(\text{EPDOL}) \setminus \mathcal{L}(\text{nbEDOL})$.
(i) Let G be the EPDOL system which is defined by $G = (\{A, a, b, c\}, \{a, b, c\}, \{A \to A, a \to b^2, b \to ac, c \to A\}, a^2b^2)$. Obviously $L(G) = K$. Thus $K \in \mathcal{L}(\text{EPDOL})$.

(ii) The fact that $K \notin \mathscr{L}$(nbEDOL) is proved by a contradiction. Assume that $K \in \mathscr{L}$(nbEDOL). Then $K = L(H)$ for an nbEDOL system $H = (V, \Sigma, P, \omega)$. Since H is deterministic there exists a positive integer t such that either $a^2b^2 \overset{t}{\underset{H}{\Rightarrow}} b^4 \cdot (ac)^2$ or $b^4(ac)^2 \overset{t}{\underset{H}{\Rightarrow}} a^2b^2$. The latter implies $b \overset{t}{\underset{H}{\Rightarrow}} \Lambda$ and $(ac)^2 \overset{t}{\underset{H}{\Rightarrow}} a^2b^2$ which is clearly impossible. Hence $a^2b^2 \overset{t}{\underset{H}{\Rightarrow}} b^4(ac)^2$. There are three cases to consider.

(a) $a \overset{t}{\underset{H}{\Rightarrow}} \Lambda$. Then however $b^2 \overset{t}{\underset{H}{\Rightarrow}} b^4(ac)^2$ which contradicts the fact that H is deterministic.

(b) $a \overset{t}{\underset{H}{\Rightarrow}} b$. Then however $b^2 \overset{t}{\underset{H}{\Rightarrow}} b^2(ac)^2$ which contradicts the fact that H is deterministic.

(c) $a \overset{t}{\underset{H}{\Rightarrow}} b^2$. Then $b^2 \overset{t}{\underset{H}{\Rightarrow}} (ac)^2$. The fact that H is deterministic yields $b \overset{t}{\underset{H}{\Rightarrow}} ac$. Observe that

(III.1)... $a \overset{*}{\underset{H}{\Rightarrow}} x$ implies $|x| \geqq 1$, and $b \overset{*}{\underset{H}{\Rightarrow}} x$ implies $|x| \geqq 1$. Clearly $a^2b^2 \overset{t}{\underset{H}{\Rightarrow}} b^4 \cdot (ac)^2 \overset{t}{\underset{H}{\Rightarrow}} (ac)^4(b^2x_1)^2 \overset{t}{\underset{H}{\Rightarrow}} (b^2x_1)^4((ac)^2x_2)^2 = z$ for some $x_1, x_2 \in V^*$.

Now the form of z and (III.1) yield that

(III.2)... for all words v such that $z \overset{*}{\underset{H}{\Rightarrow}} v$, $|v| \geqq 12$. Since the longest word of $L(H) = K$ has length 8, (III.2) contradicts the fact that H is nonblocking. Having established a contradiction for all possible cases, we get that $K \notin \mathscr{L}$(nbEDOL) which concludes the proof of (ii).

Hence the lemma holds. □

Lemma III.6. $\mathscr{L}$(nbEPDOL)$\setminus\mathscr{L}$(OL)$\neq\emptyset$.

Proof. We will prove that $K = \{a^{2^{5n}}b : n \geqq 0\} \cup \{a^{2^{5n+2}}c : n \geqq 0\} \in \mathscr{L}$(nbEPDOL)$\setminus$ $\setminus\mathscr{L}$(OL).

(i) Let G be the nbEPDOL system which is defined by $G = (\{A, B, C, a, b, c\}$, $\{a, b, c\}$, $\{A \to c, B \to C, C \to b, a \to aa, b \to A, c \to B\}$,ab). Obviously $L(G) = K$. Thus $K \in \mathscr{L}$(nbEPDOL).

(ii) The fact that $K \notin \mathscr{L}$(OL) is proved by a contradiction. Assume that $K \in$ $\in \mathscr{L}$(OL). Then $K = L(H)$ for a OL system $H = (V, V, P, \omega)$. Without loss of generality we can assume that $V = \{a, b, c\}$.

(ii.1) Clearly $a \underset{P}{\to} x$ implies $x \in a^*$, $b \underset{P}{\to} x$ implies $x \in a^*b \cup a^*c$; and $c \underset{P}{\to} x$ implies $x \in a^*b \cup a^*c$ (otherwise $L(H)$ would contain words not belonging to K).

(ii.2) The set P contains only one a-production. For assume to the contrary that there exist two different a-productions in P, say $a \to a^i$ and $a \to a^j$, $i > j$. Let $b \to x$ be an arbitrary b-production of P. Then for all $n \geqq 0$, $a^{2^{5n}}b \underset{H}{\Rightarrow} a^{2^{5n} \cdot i}x$ and

$a^{25^n}b \underset{H}{\Rightarrow} a^{25^n \cdot i - i + j}x$. Thus for all $n \geqq 0$, $a^{25^n \cdot i}x$ and $a^{25^n \cdot i - i + j}x$ belong to $L(H)$ which (for n large enough) contradicts the fact that $L(H) = K$.

(ii.3) The only a-production of P cannot be $a \to \Lambda$ otherwise $L(H)$ would be finite, a contradiction.

(ii.4) Analogously to (ii.2) we can prove that P contains only one b-production and one c-production.

Now (ii.1) through (ii.4) yield that H must be a PDOL system.

Hence $ab \underset{H}{\Rightarrow} a^4c \underset{H}{\Rightarrow} a^{32}b$. There are four cases to consider.

(a) $a \underset{H}{\Rightarrow} a$ and $b \underset{H}{\Rightarrow} a^3c$. Then however $a^{32}b \underset{H}{\Rightarrow} a^{35}c$; a contradiction.

(b) $a \underset{H}{\Rightarrow} a^2$ and $b \underset{H}{\Rightarrow} a^2c$. Then however $a^{32}b \underset{H}{\Rightarrow} a^{66}c$; a contradiction.

(c) $a \underset{H}{\Rightarrow} a^3$ and $b \underset{H}{\Rightarrow} ac$. Then however $a^{32}b \underset{H}{\Rightarrow} a^{97}c$; a contradiction.

(d) $a \underset{H}{\Rightarrow} a^4$ and $b \underset{H}{\Rightarrow} c$. Then $a^4c \underset{H}{\Rightarrow} a^{32}b$, $a \underset{H}{\Rightarrow} a^4$ and the fact that H is deterministic yield $c \underset{H}{\Rightarrow} a^{16}b$. Then however $a^{128}c \underset{H}{\Rightarrow} a^{528}b$; a contradiction.

Having established a contradiction for all possible cases, we get $K \notin \mathscr{L}(\text{OL})$. Then (i) and (ii) yield the lemma. □

We are now ready to state the main result of the section. As expected, if X denotes either P, D, PD or the empty word, we have that $\mathscr{L}(X\text{OL}) \subset \mathscr{L}(\text{nbE}X\text{OL}) \subset \subset \mathscr{L}(\text{E}X\text{OL})$.

Theorem III.2. The following diagram holds:

$$\begin{tikzcd}
 & & & \mathscr{L}(\text{EOL}) = \mathscr{L}(\text{EPOL}) & \\
 & & \mathscr{L}(\text{EDOL}) \arrow[r] & \mathscr{L}(\text{nbEOL}) \arrow[u] & \\
\mathscr{L}(\text{nbEDOL}) \arrow[urr] & & & & \mathscr{L}(\text{nbEPOL}) \arrow[ul] \\
 & \mathscr{L}(\text{nbEPDOL}) \arrow[r] \arrow[ul] & \mathscr{L}(\text{EPDOL}) \arrow[uu] \arrow[urr] & & \\
 & & & \mathscr{L}(\text{OL}) \arrow[uuu] & \\
\mathscr{L}(\text{DOL}) \arrow[uuu] \arrow[urrr] & & & & \mathscr{L}(\text{POL}) \arrow[uuu] \arrow[ul] \\
 & \mathscr{L}(\text{PDOL}) \arrow[uuu] \arrow[ul] \arrow[urrr] & & &
\end{tikzcd}$$

where, if there is a directed chain of edges in the diagram leading from a class X to a class Y then $X \subset Y$; otherwise X and Y are incomparable but not disjoint.

Proof. It is well known that $\mathscr{L}(\text{EOL}) = \mathscr{L}(\text{EPOL})$ (see, e.g., [7]). Inclusions follow from the definitions and Lemma III.2; strict inclusions and incomparabilities follow from Lemma III.1 and Lemmas III.3 through III.6. □

IV. Systems with tables

In the case of E(P)TOL systems the nonblocking restriction turns out to be no restriction with respect to the language generating power. This contrasts the results of the previous section.

Theorem IV.1. $\mathcal{L}(\text{nbEPTOL})=\mathcal{L}(\text{nbETOL})=\mathcal{L}(\text{EPTOL})=\mathcal{L}(\text{ETOL})$.

Proof. We shall show that $\mathcal{L}(\text{ETOL})\subseteq\mathcal{L}(\text{nbEPTOL})$. The theorem then follows from the definitions. Let $K\in\mathcal{L}(\text{ETOL})$. Then (see [6]) there exists a PTOL system $G=(V, V, \{P_1, P_2, \ldots, P_k\}, \omega)$, $k\geqq 1$ and a Λ-free homomorphism h: $V^*\to\Sigma^*$, such that $h(L(G))=K$. Without loss of generality assume that $V\cap\Sigma=\emptyset$. For $1\leqq i\leqq k$ let $Q_i=P_i\cup\{\alpha\to\alpha\colon \alpha\in\Sigma\}$. Let $Q=\{\alpha\to h(\alpha)\colon \alpha\in V\}\cup\{\alpha\to\alpha\colon \alpha\in\Sigma\}$. Finally define the EPTOL system $\bar{G}$ by $\bar{G}=(V\cup\Sigma, \Sigma, \{Q_1, Q_2, \ldots, Q_k, Q\}, \omega)$. Clearly $\bar{G}$ is nonblocking and $L(\bar{G})=K$. Thus $K\in\mathcal{L}(\text{nbEPTOL})$. Hence $\mathcal{L}(\text{ETOL})\subseteq\mathcal{L}(\text{nbEPTOL})$. □

Even in the case of E(P)DTOL systems the nonblocking condition has no consequences for the generating power of those systems. We first prove the following lemma.

Lemma IV.1. $\mathcal{L}(\text{EPDTOL})\subseteq\mathcal{L}(\text{nbEPDTOL})$.

Proof. Let $G=(V, \Sigma, \mathcal{P}, S)$ be an EPDTOL system where $\mathcal{P}=\{P_1, P_2, \ldots, \ldots, P_k\}$, $k\geqq 1$. Without loss of generality assume that $S\in V\setminus\Sigma$, $L(G)\neq\emptyset$ and $\underline{\text{alph}\, L(G)}=\Sigma$. Let $\bar{V}=\{\bar{\alpha}\colon \alpha\in V\}$, $\bar{V}\cap V=\emptyset$ and let $\bar{h}$ be the homomorphism on $\bar{V}^*$ defined by $\bar{h}(\alpha)=\bar{\alpha}$ for $\alpha\in V$. For each $\emptyset\neq X\subseteq V$ let w_X be a fixed word such that $\underline{\text{alph}}\, w_X=X$ and each letter occurs precisely once in w_X. Furthermore let $G_X=(V', \overline{\Sigma, \mathcal{P}'}, \bar{h}(w_X))$ be the ETOL system which is defined as follows. $V'=V\cup\bar{V}$, and $\mathcal{P}'=\{P'\colon P\in\mathcal{P}\}$ where for $P\in\mathcal{P}$, $P'=P\cup\{\bar{h}(\alpha)\to x\colon \alpha\underset{P}{\to}x\}$. Then $\text{SUC}(G)=\{\emptyset\neq X\subseteq V\colon L(G_X)\neq\emptyset\}$, in other words for a $w\in V^+$, $\underline{\text{alph}}\, w\in\text{SUC}(G)$ if and only if there exists a $w'\in\Sigma^+$ such that $w\overset{+}{\underset{G}{\Rightarrow}}w'$. For $X\in\text{SUC}(G)$ we define $\underline{\text{next}}\; X=\{i\colon P_i\in\mathcal{P}, w_X\underset{P_i}{\Rightarrow}y, \underline{\text{alph}}\; y\in\text{SUC}(G)$ or $\underline{\text{alph}}\; y\subseteq\Sigma\}$. Now we will construct an nbEPDTOL system H such that $L(G)=L(H)$. We proceed as follows. $\hat{V}=\{S\}\cup\Sigma\cup\bigcup_{X\in\text{SUC}(G)}\{[\alpha, X]_i\colon \alpha\in\underline{\text{alph}}\; X, i\in\underline{\text{next}}\; X\}$, $\hat{V}\cap(V\setminus(\{S\}\cup\Sigma))=\emptyset$. For $i\in\underline{\text{next}}\;\{S\}$, define $Q_{\text{in},i}=\{S\to[S, \{S\}]_i\}\cup\{\alpha\to\alpha\colon \alpha\in\hat{V}\setminus\{S\}\}$. For $X\in\text{SUC}(G)$, $w_X\underset{P_i}{\Rightarrow}y$, $\underline{\text{alph}}\; y=Y\in\text{SUC}(G)$ and $j\in\underline{\text{next}}\; Y$ define

$$Q_{X,i,j}=\{[\alpha, X]_i\to[\beta_1, Y]_j[\beta_2, Y]_j\ldots[\beta_m, Y]_j\colon \alpha\in X, \alpha\underset{P_i}{\to}\beta_1\beta_2\ldots\beta_m, m\geqq 1, \beta_l\in V$$
$$\text{for}\quad 1\leqq l\leqq m\}\cup\{\alpha\to\alpha\colon \alpha\in\hat{V}\setminus\{[\beta, X]_i\colon \beta\in X\}\}.$$

For $X\in\text{SUC}(G)$, $w_X\underset{P_i}{\Rightarrow}y$, $\underline{\text{alph}}\; y\subseteq\Sigma$ define

$$Q_{X,i,\text{fin}}=\{[\alpha, X]_i\to z\colon \alpha\in X, \alpha\underset{P_i}{\to}z\}\cup\{\alpha\to\alpha\colon \alpha\in\hat{V}\setminus\{[\beta, X]_i\colon \beta\in X\}\}.$$

Let $\hat{P}=\{Q_{\mathrm{in},i}: i\in \underline{\mathrm{next}}\ \{S\}\}\cup\{Q_{X,i,j}: X\in\mathrm{SUC}(G),\ w_X\underset{P_i}{\Rightarrow}y,\ \underline{\mathrm{alph}}\ y=Y\in\mathrm{SUC}(G)$ and $j\in\underline{\mathrm{next}}\ y\}\cup\{Q_{X,i,\mathrm{fin}}: X\in\mathrm{SUC}(G),\ w_X\underset{P_i}{\Rightarrow}y,\ \underline{\mathrm{alph}}\ Y\subseteq\Sigma\}$.

Finally let H be the EPDTOL system defined by $H=(\hat{V}, \Sigma, \hat{\mathscr{P}}, S)$. First we show that $L(H)=L(G)$. For $X\in\mathrm{SUC}(G)$ and $i\in\underline{\mathrm{next}}\ X$ the homomorphism $h_{X,i}$ on V^* is defined by $h_{X,i}(\alpha)=[\alpha, X]_i$ if $\alpha\in V$; furthermore the homomorphism g on $\hat{V}^*$ is defined by $g(\alpha)=\alpha$ if $\alpha\in\{S\}\cup\Sigma$ and $g([\alpha, X]_i)=\alpha$ if $X\in\mathrm{SUC}(G)$, $\alpha\in X$ and $i\in\underline{\mathrm{next}}\ X$. Let $x\in L(G)$, thus $S=x_0\underset{P_{i_1}}{\Rightarrow}x_1\underset{P_{i_2}}{\Rightarrow}x_2\Rightarrow\ldots\underset{P_{i_n}}{\Rightarrow}x_n=x$, $n\geqq 1$, $i_1, \ldots, \ldots, i_n\in\{1, \ldots, k\}$. Then obviously, if for $0\leqq l\leqq n$ we denote $\underline{\mathrm{alph}}\ x_l=X_l$,

$$S \underset{Q_{in,i_1}}{\Rightarrow} h_{X_0,i_1}(x_0) \underset{Q_{X_0,i_1,i_2}}{\Rightarrow} h_{X_1,i_2}(x_1)\Rightarrow\ldots \underset{Q_{X_{n-2},i_{n-1},i_n}}{\Rightarrow} h_{X_{n-1},i_n}(x_{n-1}) \underset{Q_{X_{n-1},i_n,\mathrm{fin}}}{\Rightarrow} x_n=x.$$

Consequently $x\in L(H)$. Hence $L(G)\subseteq L(H)$.

Conversely let $x\in L(H)$ and let D: $S=x_0\underset{H}{\Rightarrow}x_1\underset{H}{\Rightarrow}x_2\underset{H}{\Rightarrow}\ldots\underset{H}{\Rightarrow}x_n=x$ be a shortest derivation of x in H. Thus, if for $0\leqq l\leqq n$ we denote $\underline{\mathrm{alph}}\ g(x_l)=X_l$,

$$D\colon S=x_0 \underset{Q_{\mathrm{in},i_1}}{\Rightarrow} x_1 \underset{Q_{X_1,i_1,i_2}}{\Rightarrow} x_2\Rightarrow\ldots \underset{Q_{X_{n-2},i_{n-2},i_{n-1}}}{\Rightarrow} x_{n-1} \underset{Q_{X_{n-1},i_{n-1},\mathrm{fin}}}{\Rightarrow} x_n=x,$$

$n\geqq 2$ and $i_1, \ldots, i_{n-1}\in\{1, \ldots, k\}$. Consequently

$$S=x_0\underset{P_{i_1}}{\Rightarrow}g(x_2)\underset{P_{i_2}}{\Rightarrow}\ldots\underset{P_{i_{n-2}}}{\Rightarrow}g(x_{n-1})\underset{P_{i_{n-1}}}{\Rightarrow}g(x_n)=x$$

and thus $x\in L(G)$. Hence $L(H)\subseteq L(G)$.

We end the proof of the lemma by showing that H is nonblocking. Let $x\in\underline{\mathrm{sent}}\ H$. Then there are three possible cases: $x=S$ or $x\in\Sigma^+$ or $x=h_{X,i}(v)$, $v\in V^+$, $X\in\mathrm{SUC}(G)$ and $i\in\underline{\mathrm{next}}\ X$. Since $L(H)=L(G)\neq\emptyset$ it suffices to consider sentential forms of the third kind. Thus $x=h_{X,i}(v)$, $v\in V^+$, $X\in\mathrm{SUC}(G)$ and $i\in\underline{\mathrm{next}}\ X$. Hence there exist v' and v'' such that $v\underset{P_i}{\Rightarrow}v'\overset{*}{\underset{G}{\Rightarrow}}v''\in\Sigma^+$. Then inspecting the proof of $L(G)\subseteq L(H)$ one can easily see that $x=h_{X,i}(v)\overset{*}{\underset{H}{\Rightarrow}}v''$ which shows that H is nonblocking. □

As a corollary we obtain the answer to an open problem stated in [6].

Definition IV.1. A language L is contained in $\mathscr{L}$(NPDTOL) if and only if there exists a PDTOL system H and a non-erasing homomorphism h such that $L=h(L(H))$.

Corollary IV.1. $\mathscr{L}$(NPDTOL)$=\mathscr{L}$(EPDTOL).

Proof. We will use the notation from the proof of Lemma IV.1. Fix a $u_s\in\Sigma^+$ such that $S\overset{+}{\underset{G}{\Rightarrow}}u_s$ and for each $X\in\mathrm{SUC}(G)$ let D_X: $w_X\overset{+}{\underset{G}{\Rightarrow}}u_X\in\Sigma^+$ be a fixed derivation. Then define the Λ-free homomorphism h on $\hat{V}^*$ as follows: $h(S)=u_S$, $h([\alpha, X]_i)=\underline{\mathrm{ctr}}_{D_X, w_X}\alpha$ if $X\in\mathrm{SUC}(G)$, $\alpha\in\underline{\mathrm{alph}}\ X$ and $i\in\underline{\mathrm{next}}\ X$, and $h(\alpha)=\alpha$ if

$\alpha\in\Sigma$. Let $\hat{H}'$ be the PDTOL system defined by $\hat{H}'=(\hat{V}, \hat{V}, \hat{\mathscr{P}}, S)$. Clearly $L(G)=h(L(\hat{H}'))$. Hence $\mathscr{L}$(EPDTOL)$\subseteq\mathscr{L}$(NPDTOL). Since also $\mathscr{L}$(NPDTOL)$\subseteq\mathscr{L}$(EPDTOL) (see [6]), the corollary holds. □

For the deterministic case we obtain a result analogous to the statement of Theorem IV.1.

Theorem IV.2. $\mathscr{L}$(nbEPDTOL)$=\mathscr{L}$(nbEDTOL)$=\mathscr{L}$(EPDTOL)$=\mathscr{L}$(EDTOL).

Proof. From the definitions we get $\mathscr{L}$(nbEPDTOL)$\subseteq\mathscr{L}$(nbEDTOL)$\subseteq\mathscr{L}$(EDTOL). It is well known (see, e.g., [1]) that $\mathscr{L}$(EDTOL)$=\mathscr{L}$(EPDTOL). From Lemma IV.1 we get $\mathscr{L}$(EPDTOL)$\subseteq\mathscr{L}$(nbEPDTOL). Combining the above results, the theorem immediately follows. □

Let X and Y denote P, D, PD or the empty word. Then Theorem IV.1 and Theorem IV.2 show that $\mathscr{L}$(nbEXTOL)$=\mathscr{L}$(EXTOL). Thus comparing $\mathscr{L}$(nbEXTOL) and $\mathscr{L}$(YTOL) is the same as comparing $\mathscr{L}$(EXTOL) and $\mathscr{L}$(YTOL). For completeness only we present here the diagram in the case of tabled L systems. The proof is given using well known results from the literature.

Theorem IV.3. The following diagram holds:

$$\begin{tikzcd}
 & & \mathscr{L}(\text{ETOL}) = \mathscr{L}(\text{nbETOL}) = \mathscr{L}(\text{EPTOL}) = \mathscr{L}(\text{nbEPTOL}) & & \\
 & \mathscr{L}(\text{EDTOL}) = \mathscr{L}(\text{nbEDTOL}) = \mathscr{L}(\text{EPDTOL}) = \mathscr{L}(\text{nbEPDTOL}) \arrow[ur] & & \mathscr{L}(\text{TOL}) \arrow[ul] & \\
 & & \mathscr{L}(\text{DTOL}) \arrow[ul] \arrow[ur] & & \mathscr{L}(\text{PTOL}) \arrow[ul] \\
 & & & \mathscr{L}(\text{PDTOL}) \arrow[ul] \arrow[ur] &
\end{tikzcd}$$

where, if there is a directed chain of edges in the diagram leading from a class X to a class Y then $X\subset Y$; otherwise X and Y are incomparable but not disjoint.

Proof. Inclusions follow from the definitions, equalities follow from Theorem IV.1 and Theorem IV.2. Strict inclusions and incomparabilities follow from the following three observations.

(i) $\{ba^{2^n}: n\geqq 0\}\cup\{bc^{3^n}: n\geqq 0\}\in\mathscr{L}(\text{DTOL})\setminus\mathscr{L}(\text{PTOL})$ (see, e.g., [3]).

(ii) $\{w\in\{a, b\}^*: |w|=2^n \text{ for some } n\geqq 0\}\in\mathscr{L}(\text{PTOL})\setminus\mathscr{L}(\text{EDTOL})$ (see, e.g., [7]).

(iii) All finite languages are in $\mathscr{L}$(EDTOL) and there are finite languages which are not TOL languages (see, e.g., [3]). □

Since emptiness is a decidable property for ETOL systems (see, e.g., [7]) and since all constructions used in this section are effective, it follows that for every system, considered in this section, generating a non-empty language, there exists effectively an equivalent nonblocking system. This contrasts Theorem II.4. Moreover it turns out that nonblocking is a decidable property for ETOL systems. This result should be compared with Theorem II.5.

Theorem IV.4. Let G be an ETOL system. Then it is decidable whether or not G is nonblocking.

Proof. Let $G=(V, \Sigma, \mathscr{P}, \omega)$ be an ETOL system. Let $\bar{V}=\{\bar{\alpha}: \alpha\in V\}$, $\bar{V}\cap V=\emptyset$ and let $\bar{h}$ be the homomorphism on V^* defined by $\bar{h}(\alpha)=\bar{\alpha}$ for $\alpha\in V$. For each $\emptyset\neq X\subseteq V$ let w_X be a fixed word such that $\mathrm{alph}\, w_X=X$ and each letter occurs precisely once in w_X. Furthermore let $G_X=(V', \Sigma, \mathscr{P}', \bar{h}(w_X))$ be the ETOL system which is defined as follows. $V'=V\cup\bar{V}$, and $\mathscr{P}'=\{P': P\in\mathscr{P}\}$ where for $P\in\mathscr{P}$, $P'=P\cup\{\bar{h}(\alpha)\to x: \alpha \underset{P}{\to} x\}$. Let $\mathscr{A}=\{G_X: \mathrm{sent}\, G\cap\{x\in X^*: \mathrm{alph}\, x=X\}\neq\emptyset\}$. Obviously G is nonblocking if and only if $\mathscr{A}\neq\emptyset$ and for each $H\in\mathscr{A}$, $L(H)\neq\emptyset$. The decidability of the latter question follows from the closure properties of $\mathscr{L}$(ETOL), the effectiveness of the construction of $\mathscr{A}$ and the decidability of the emptiness problem for ETOL systems. Hence the theorem holds. □

Discussion

In this paper we have investigated the effect that the nonblocking restriction has on the language generating power of various classes of rewriting systems. Since the blocking facility forms a typical "programming tool" in generating a language, we believe that our results shed some light on the nature of the generation of languages by grammars.

The research started in this paper can be continued in several directions.

(1) The class of languages generated by the "nonblocking subclass" of a class X of rewriting systems should be often investigated on its own (whenever the nonblocking restriction influences the language generating power of the class X). Such a typical candidate to investigate is $\mathscr{L}$(nbEOL); for example the closure properties and the combinatorial properties of languages in this class. Also the decidability status of the question "Does an arbitrary EOL system generate a language in $\mathscr{L}$(nbEOL)?" forms an interesting open problem.

(2) The role of the nonblocking restriction in classes of rewriting systems different from those investigated in this paper should also be investigated.

(3) Clearly the way that we have formally defined the nonblocking of a rewriting system is only one of several possibilities. Other possibilities should also be investigated.

(4) A nonblocking condition can be also defined for various types of automata, for example one could require that for every state of an automaton there exists a computation that leads from this state to an accepting state. (Conditions of this type are often considered in the theory of Petri-nets (see, e.g., [2]), where they are referred to as "liveness conditions".) The effect of nonblocking on the generative power of various classes of automata should be investigated.

Appendix

Here we give the full proof of Lemma II.2.

For every context-sensitive grammar, generating a non-empty language there exists an equivalent nonblocking context-sensitive grammar.

Proof. Let $K \subseteq \Sigma^*$ be a non-empty language generated by a context-sensitive grammar.

1) If K is finite, then let $G=(\Sigma \cup \{S\}, \Sigma, P, S)$ be the context-sensitive grammar with $P=\{S \to x : x \in K\}$. Obviously; G is a nonblocking context-sensitive grammar and $L(G)=K$.

2) If K is infinite, we proceed as follows. Let $\Sigma' = \{[a, b, c, d] : a, b, c, d \in \Sigma\} \cup \cup \{[a, b, c] : a, b, c \in \Sigma\} \cup \{[a, b] : a, b \in \Sigma\} \cup \{[a] : a \in \Sigma\}$ with $\Sigma' \cap \Sigma = \emptyset$.

Let $K' = \{[a_1, a_2, a_3, a_4] \dots [a_{4n-3}, a_{4n-2}, a_{4n-1}, a_{4n}] : n \geqq 2, a_i \in \Sigma$, for $1 \leqq i \leqq 4n$, and $a_1 a_2 \dots a_{4n} \in K\} \cup \{[a_1, a_2, a_3, a_4] \dots [a_{4n-3}, a_{4n-2}, a_{4n-1}, a_{4n}][a_{4n+1}] : n \geqq 2, a_i \in \Sigma$, for $1 \leqq i \leqq \leqq 4n+1$, and $a_1 a_2 \dots a_{4n+1} \in K\} \cup \{[a_1, a_2, a_3, a_4] \dots [a_{4n-3}, a_{4n-2}, a_{4n-1}, a_{4n}][a_{4n+1}, a_{4n+2}] : : n \geqq 2, a_i \in \Sigma$, for $1 \leqq i \leqq 4n+2$, and $a_1 a_2 \dots a_{4n+2} \in K\} \cup \{[a_1, a_2, a_3, a_4] \dots \dots [a_{4n-3}, a_{4n-2}, a_{4n-1}, a_{4n}][a_{4n+1}, a_{4n+2}, a_{4n+3}] : n \geqq 2, a_i \in \Sigma$, for $1 \leqq i \leqq 4n+3$, and $a_1 a_2 \dots a_{4n+3} \in K\}$.

Let h be the homomorphism from Σ'^* into Σ^* defined by $h([a_1, a_2, a_3, a_4]) = = a_1 a_2 a_3 a_4$, $h([a_1, a_2, a_3]) = a_1 a_2 a_3$, $h([a_1, a_2]) = a_1 a_2$ and $h([a_1]) = a_1$, for $a_i \in \Sigma$, $1 \leqq i \leqq 4$. Clearly $h(K') = K \setminus \{x \in K : |x| < 8\}$ and hence $K' \in \mathscr{L}(\mathrm{CS})$. (See, e.g., [4].) Let $G' = (V', \Sigma', P', S')$ be a context-sensitive grammar, such that $(V' \setminus \Sigma') \cap \cap \Sigma = \emptyset$ and $L(G') = K'$. Without loss of generality we assume that no terminals occur in the left-hand side of any production of P'.

The context-sensitive grammar $G=(V, \Sigma, P, S)$ is defined as follows. $V= = \overline{V} \cup V' \cup \Sigma$, where $\overline{V} = \{S, L, R, L_1, R_1, N, N_L, \vec{N}, \vec{B}, \overleftarrow{B}, \vec{M}_0, \overleftarrow{M}_0, \vec{M}_1, \overleftarrow{M}_1, M_2, \vec{M}_2, \vec{M}_3, \overleftarrow{M}_3, X_1, X_2\}$ and $\overline{V} \cap (V' \cup \Sigma) = \emptyset$.

P consists of the following productions.

(1) $S \to x$, if $x \in K$ and $|x| < 8$.
(2) $S \to L\vec{M}_0 S'R$.
(3) All productions from P'.
(4) $\vec{M}_0 \alpha \to \alpha \vec{M}_0$, if $\alpha \in \Sigma'$.
(5) $\vec{M}_0 \to \overleftarrow{B}$.
(6) $[a_1, a_2, a_3, a_4]\vec{M}_0 R \to \overleftarrow{M}_0 a_1 a_2 a_3 a_4$,
$[a_1, a_2, a_3, a_4][a_5]\vec{M}_0 R \to \overleftarrow{M}_0 a_1 a_2 a_3 a_4 a_5$,
$[a_1, a_2, a_3, a_4][a_5, a_6]\vec{M}_0 R \to \overleftarrow{M}_0 a_1 a_2 a_3 a_4 a_5 a_6$ and
$[a_1, a_2, a_3, a_4][a_5, a_6, a_7]\vec{M}_0 R \to \overleftarrow{M}_0 a_1 a_2 a_3 a_4 a_5 a_6 a_7$, for $a_i \in \Sigma$, $1 \leqq i \leqq 7$.
(7) $\alpha \overleftarrow{B} \to \overleftarrow{B} N$, if $\alpha \in V' \cup \Sigma \cup \{N, L_1, \vec{N}, M_2, \vec{M}_2\}$.
(8) $L\overleftarrow{B} \to N_L \vec{B}$ and $N_L \overleftarrow{B} \to N_L \vec{B}$.
(9) $[a_1, a_2, a_3, a_4][a_5, a_6, a_7, a_8]\overleftarrow{M}_0 \to [a_1, a_2, a_3, a_4]\overleftarrow{M}_0 a_5 a_6 a_7 a_8$ for $a_i \in \Sigma$, $1 \leqq i \leqq 8$.
(10) $L[a_1, a_2, a_3, a_4]\overleftarrow{M}_0 \to a_1 a_2 a_3 a_4$, for $a_i \in \Sigma$, $1 \leqq i \leqq 4$.
(11) $\vec{B}\alpha \to N\vec{B}$, if $\alpha \in V' \cup \{N\}$,

(12) $\vec{B}R \to NL_1\vec{M}_1S'R_1$ and $\vec{B}R_1 \to NL_1\vec{M}_1S'R_1$.
(13) $\vec{M}_1\alpha \to \alpha\vec{M}_1$, if $\alpha \in \Sigma'$.
(14) $\vec{M}_1 \to \overleftarrow{B}$.
(15) $\vec{M}_1R_1 \to \overleftarrow{M}_1R_1$.
(16) $\alpha\overleftarrow{M}_1 \to \overleftarrow{M}_1\alpha$, if $\alpha \in \Sigma'$.
(17) $L_1\overleftarrow{M}_1 \to NM_2$.
(18) $NM_2 \to M_2\vec{N}$.
(19) $N_LM_2 \to N_L\vec{M}_2$.
(20) $\vec{N}\alpha \to \alpha\vec{N}$, if $\alpha \in \Sigma$.
(21) $\vec{N}[a_1, a_2, a_3, a_4] \to a_1a_2a_3a_4$, for $a_i \in \Sigma$, $1 \leqq i \leqq 4$.
(22) $\vec{N}[a_1, a_2, a_3] \to \overleftarrow{B}N$, $\vec{N}[a_1, a_2] \to \overleftarrow{B}N$, $\vec{N}[a_1] \to \overleftarrow{B}N$ and $\vec{N}R_1 \to \overleftarrow{B}NR_1$, for $a_i \in \Sigma$, $1 \leqq i \leqq 3$.
(23) $\vec{M}_2\alpha \to \alpha\vec{M}_2$, if $\alpha \in \Sigma$.
(24) $\vec{M}_2[a_1, a_2, a_3, a_4] \to a_1a_2a_3a_4\vec{M}_3$, for $a_i \in \Sigma$, $1 \leqq i \leqq 4$.
(25) $\vec{M}_2[a_1, a_2, a_3] \to \overleftarrow{B}N$, $\vec{M}_2[a_1, a_2] \to \overleftarrow{B}N$, $\vec{M}_2[a_1] \to \overleftarrow{B}N$ and $\vec{M}_2R_1 \to \overleftarrow{B}NR_1$, for $a_i \in \Sigma$, $1 \leqq i \leqq 3$.
(26) $\vec{M}_3[a_1, a_2, a_3, a_4] \to \overleftarrow{M}_3[a_1, a_2, a_3, a_4]$, for $a_i \in \Sigma$, $1 \leqq i \leqq 4$.
(27) $\vec{M}_3[a_1, a_2, a_3] \to \overleftarrow{B}N$, $\vec{M}_3[a_1, a_2] \to \overleftarrow{B}N$, $\vec{M}_3[a_1] \to \overleftarrow{B}N$ and $\vec{M}_3R \to \overleftarrow{B}NR_1$, for $a_i \in \Sigma$, $1 \leqq i \leqq 3$.
(28) $\alpha\overleftarrow{M}_3 \to \overleftarrow{M}_3\alpha$, if $\alpha \in \Sigma$.
(29) $N_L\overleftarrow{M}_3 \to X_1X_2$.
(30) $X_1X_2\alpha \to \alpha X_1X_2$, if $\alpha \in \Sigma$.
(31) $X_1X_2[a_1, a_2, a_3, a_4][a_5, a_6, a_7, a_8] \to a_1a_2a_3a_4X_1X_2[a_5, a_6, a_7, a_8]$, for $a_i \in \Sigma$, $1 \leqq i \leqq 8$.
(32) $X_1X_2[a_1, a_2, a_3, a_4][a_5, a_6, a_7]R_1 \to a_1a_2a_3a_4a_5a_6a_7$,
$X_1X_2[a_1, a_2, a_3, a_4][a_5, a_6]R_1 \to a_1a_2a_3a_4a_5a_6$,
$X_1X_2[a_1, a_2, a_3, a_4][a_5]R_1 \to a_1a_2a_3a_4a_5$ and
$X_1X_2[a_1, a_2, a_3, a_4]R_1 \to a_1a_2a_3a_4$, for $a_i \in \Sigma$, $1 \leqq i \leqq 7$.

First we show that $L(G) \subseteqq K$. Starting from the axiom S only productions from (1) and (2) can be applied, resulting either in a word $x \in K$, $|x| < 8$, or in a word of $\underline{\text{sent}}\ G$ of type A, i.e. of the form $Lx\vec{M}_0yR$, with $x \in \Sigma'^*$ and $xy \in \underline{\text{sent}}\ G'$.

The productions, applicable to words of $\underline{\text{sent}}\ G$ which are of type A belong to (3), (4), (5) and (6). If a production from (3) or (4) is applied to a word of type A, the resulting word again is of type A.

If a production from (5) is applied to a word of $\underline{\text{sent}}\ G$ of type A, we get a word of type B, i.e. of the form $Lx\overleftarrow{B}yR$, with $xy \in (V' \cup \{N\})^*$. If a production from (6) is applied to a word of type A, the resulting word is of type C, i.e. of the form $Lx\overleftarrow{M}_0y$, with $x \in \Sigma'^+$, $y \in \Sigma^+$, $h(x)y \in K$ and $|h(x)y| \geqq 8$.

The productions, applicable to words of type B come from (3), (7) or (8). Application of productions from (3) and (7) to a word of type B again yields a word of type B, whereas application of productions from (8) yields a word of type D, i.e. of the form $N_LN^*\vec{B}xR$ or $N_LN^*\vec{B}xR_1$, $x \in (V' \cup \{N\})^*$.

The productions, applicable to words of type C belong to (9) or (10). Application of a production from (9) to a word of type C yields a word of the same type, whereas application of a production from (10) yields a word of K.

The productions, applicable to words of type D belong to (3), (11) or (12). The application of a production from (3) or (11) to a word of type D results in a word of the same type; the application of a production from (12) yields a word of type E, i.e. of the form $N_L N^+ L_1 x \overrightarrow{M}_1 y R_1$, $x \in \Sigma'^*$, $xy \in \underline{\text{sent}}\, G'$.

The productions, applicable to a word of type E come from (3), (13), (14) or (15). Application of a production from (3) or (13) to a word of type E yields a word of the same type. Application of a production from (14) to a word of type E yields a word of type F, i.e. of the form $N_L x \overleftarrow{B} y R_1$ with $xy \in (V' \cup \{L_1, N\})^*$. Application of a production from (15) to a word of type E yields a word of type $\hat{G}$, i.e. of the form $N_L N^+ L_1 x \overleftarrow{M}_1 y R_1$ with $xy \in \Sigma'^+$, $h(xy) \in K$ and $|h(xy)| \geqq 8$.

The productions, applicable to a word of type F come from (3), (7) or (8), and if applied, yield words of type F, type F and type D respectively.

The productions, applicable to a word of type $\hat{G}$, belong to (16) or (17), and, if applied, yield respectively words of type $\hat{G}$ and type H, i.e. of the form $N_L N^* M_2 (\{\overrightarrow{N}\} \cup \Sigma)^* \Sigma'^* R_1$, and furthermore if a word has this form, then also $h(\underline{\text{Pres}}_{\Sigma \cup \Sigma'}\, w) = w' \in K$ with $|w'| \geqq 8$.

The productions, applicable to a word of type H belong to (18), (19), (20), (21) or (22) and then yield words of type H, type I, type H, type H or type J respectively, where type I and type J are defined as follows.

A word w is of type I if $w \in N_L \Sigma^* \overrightarrow{M}_2 (\{\overrightarrow{N}\} \cup \Sigma)^* \Sigma'^* R_1$ and $h(\underline{\text{Pres}}_{\Sigma \cap \Sigma'}\, w) = = w' \in K$, with $|w'| \geqq 8$.

A word is of type J if it is of the form $N_L N^* M_2 x \overleftarrow{B} N^+ R_1$, with $x \in (\Sigma \cup \{N, \overrightarrow{N}\})^*$, or $N_L \Sigma^* \overrightarrow{M}_2 y B \overleftarrow{N}^+ R_1$, with $y \in (\Sigma \cup \{N, \overrightarrow{N}\})^*$, or $N_L \Sigma^* \overleftarrow{B} N^+ R_1$.

The productions, applicable to words of type I belong to (20), (21), (22), (23), (24) or (25) and then yield words of type I, type I, type J, type I, type L or type J respectively, where type L is defined as follows.

A word is of type L if it is of the form $N_L x \overrightarrow{M}_3 y R_1$, with $x \in \Sigma^*$, $y \in \Sigma'^+$, $xh(y) \in K$ and $|xh(y)| \geqq 8$.

The productions, applicable to words of type J belong to (7), (8), (18), (19), (20) or (23) and then yield either a word of type J or type D.

The productions, applicable to words of type L come from (26) or (27) and then yield words of type M or J respectively, where type M is defined as follows. A word is of type M if it is of the form $N_L x \overleftarrow{M}_3 y R_1$ with $x \in \Sigma^*$, $y \in \Sigma'^+$, $xh(y) \in K$ and $|xh(y)| \geqq 8$.

The only productions, applicable to a word of type M come from (28) through (32) and they lead in a deterministic way to $xh(y)$ if the word, they were applied to, was $N_L x \overleftarrow{M}_3 y R_1$.

The above reasoning shows that $L(G) \subseteqq K$.

That $K \subseteqq L(G)$ can be seen as follows.

If $x \in K$ and $|x| < 8$, then $S \underset{G}{\Rightarrow} x$ and hence $x \in L(G)$.

If $x \in K$ and $|x| \geqq 8$, say $x = a_1 \dots a_k$, $a_i \in \Sigma$ for $1 \leqq i \leqq k$ and $k \geqq 8$, then $S \underset{G}{\Rightarrow} L\overrightarrow{M}_0 S' R \overset{*}{\underset{G}{\Rightarrow}} L\overrightarrow{M}_0 y R$, with $y \in K'$ and $h(y) = x$ and $L\overrightarrow{M}_0 y R \overset{*}{\underset{G}{\Rightarrow}} L y \overrightarrow{M}_0 R \overset{*}{\underset{G}{\Rightarrow}} L y \overleftarrow{M}_0 R \overset{*}{\underset{G}{\Rightarrow}} x$. Thus $x \in L(G)$. We conclude $K \subseteqq L(G)$.

We end the proof by showing that G is nonblocking. To this aim we have to show that for each $w \in \underline{\text{sent}}\, G$, there exists a $\bar{w} \in L(G)$ such that $w \overset{*}{\underset{G}{\Rightarrow}} \bar{w}$. From the proof that $L(G) \subseteq K$ it should be clear that it suffices to prove that each word of $\underline{\text{sent}}\,(G)$ which is of type A through M can lead to a terminal word. For words of types C and M this was already proved in the above. Inspecting the productions of G, we make the following observations. Let $w \in \underline{\text{sent}}\, G$.

If w is of type A, then $w \overset{*}{\underset{G}{\Rightarrow}} w'$ for a w' of type B.

If w is of type B, then $w \overset{*}{\underset{G}{\Rightarrow}} w'$ for a w' of type D.

If w is of type E, then $w \overset{*}{\underset{G}{\Rightarrow}} w'$ for a w' of type F.

If w is of type F, then $w \overset{*}{\underset{G}{\Rightarrow}} w'$ for a w' of type D.

If w is of type $\hat{G}$, then $w \overset{*}{\underset{G}{\Rightarrow}} w'$ for a w' of type H.

If w is of type H, then $w \overset{*}{\underset{G}{\Rightarrow}} w'$ for a w' of type I.

If w is of type I, then $w \overset{*}{\underset{G}{\Rightarrow}} w'$ for a w' of type J or L.

If w is of type J, then $w \overset{*}{\underset{G}{\Rightarrow}} w'$ for a w' of type D.

If w is of type L, then $w \overset{*}{\underset{G}{\Rightarrow}} w'$ for a w' of type J or M.

Hence for each $w \in \underline{\text{sent}}\, G$ of type A, B, D through M, there exists a $w' \in \underline{\text{sent}}\, G$ such that $w \overset{*}{\Rightarrow} w'$ and w' is either of type D or of type M.

Since each word of $\underline{\text{sent}}\, G$ of type M can derive a word of K, it remains to show that each word of $\underline{\text{sent}}\, G$ of type D can derive a terminal word.

This is seen as follows. Let $w \in \underline{\text{sent}}\, G$ and w is of type D. Then $w \overset{*}{\underset{G}{\Rightarrow}} N_L N^i L_1 \vec{M}_1 S' R_1$ for some $i > 0$. Since K is infinite, K' is also infinite. Hence there is a word $x = a_1 \dots a_k$, with $a_j \in \Sigma'$, $1 \leqq j \leqq k$, such that $x \in K'$ and $k \geqq i+4$. Then

$$N_L N^i L_1 \vec{M}_1 S' R_1 \overset{*}{\underset{G}{\Rightarrow}} N_L N^i L_1 \vec{M}_1 x R_1 \overset{*}{\underset{G}{\Rightarrow}} N_L N^i L_1 x \vec{M}_1 R_1 \overset{*}{\underset{G}{\Rightarrow}} N_L N^i L_1 x \overleftarrow{M}_1 R_1 \overset{*}{\underset{G}{\Rightarrow}}$$

$$\overset{*}{\underset{G}{\Rightarrow}} N_L N^i L_1 \overleftarrow{M}_1 x R_1 \overset{*}{\underset{G}{\Rightarrow}} N_L N^{i+1} M_2 x R_1 \overset{*}{\underset{G}{\Rightarrow}} N_L M_2 \bar{N}^{i+1} a_1 a_2 \dots a_{i+1} a_{i+2} a_{i+3} \dots a_k R_1$$

$$\overset{*}{\underset{G}{\Rightarrow}} N_L M_2 h(a_1 \dots a_{i+1}) a_{i+2} a_{i+3} \dots a_k R_1 \overset{*}{\underset{G}{\Rightarrow}} N_L \vec{M}_2 h(a_1 \dots a_{i+1}) a_{i+2} a_{i+3} \dots a_k R_1$$

$$\overset{*}{\underset{G}{\Rightarrow}} N_L h(a_1 \dots a_{i+1}) \vec{M}_2 a_{i+2} a_{i+3} \dots a_k R_1 \overset{*}{\underset{G}{\Rightarrow}} N_L h(a_1 \dots a_{i+2}) \vec{M}_3 a_{i+3} \dots a_k R_1$$

$$\overset{*}{\underset{G}{\Rightarrow}} N_L h(a_1 \dots a_{i+2}) \overleftarrow{M}_3 a_{i+3} \dots a_k R_1 \overset{*}{\underset{G}{\Rightarrow}} N_L \overleftarrow{M}_3 h(a_1 \dots a_{i+2}) a_{i+3} \dots a_k R_1$$
$$\overset{*}{\underset{G}{\Rightarrow}} X_1 X_2 h(a_1 \dots a_{i+2}) a_{i+3} \dots a_k R_1 \overset{*}{\underset{G}{\Rightarrow}} h(a_1 \dots a_{i+2}) X_1 X_2 a_{i+3} \dots a_k R_1 \overset{*}{\underset{G}{\Rightarrow}} h(a_1 \dots a_k).$$

Since $a_1 \dots a_k \in K'$, $h(a_1 \dots a_k) \in K$ and hence w derives a word of K.

Thus G is a nonblocking context-sensitive grammar such that $L(G)=K$. Hence the lemma holds. □

Acknowledgements. The authors are indebted to NFWO Belgium for supporting their research. The second author gratefully acknowledges the support of NSF grant MCS 79-03838.

Abstract

A rewriting system G is called *nonblocking* if every sentential form of it can be rewritten into a word of the language of G; otherwise G is called *blocking*. The blocking facility is often used in generating languages by rewriting systems (for example in context-sensitive grammars and EOL systems). This paper initiates the formal investigation of the role that the nonblocking restriction has on the language generating power of various classes of rewriting systems. We investigate grammars of the Chomsky hierarchy as well as context independent L systems with and without tables.

* INSTITUTE OF APPLIED MATHEMATICS AND COMPUTER SCIENCE
UNIVERSITY OF LEIDEN
WASSENAARSEWEG 80
LEIDEN, THE NETHERLANDS

** RESEARCH ASSISTANT OF THE BELGIAN NATIONAL FUND FOR SCIENTIFIC RESEARCH (NFWO)
DEPARTMENT OF MATHEMATICS
UNIVERSITY OF ANTWERP, U.I.A
B—2610 WILRIJK, BELGIUM,

References

[1] ASVELD, P. R. J. and J. ENGELFRIET, Iterated deterministic substitution, *Acta Inform.*, v. 8, 1977, pp. 285—302.
[2] BRAUER, W. (ed.), *Net theory and its applications*, Lecture Notes in Computer Science 84, Springer Verlag, Berlin/New York, 1980.
[3] HERMAN, G. T. and G. ROZENBERG, *Developmental systems and languages*, North-Holland, Amsterdam/Oxford, 1975.
[4] HOPCROFT, J. E. and J. D. ULLMAN, *Introduction to automata, languages and computation*, Addison-Wesley, Reading, Mass., 1979.
[5] MAURER, H. A., A. SALOMAA and D. WOOD, Pure grammars, *Inform. and Control*, v. 44, 1980, pp. 47—72.
[6] NIELSSEN, M., G. ROZENBERG, A. SALOMAA and S. SKYUM, Nonterminals, homomorphisms and codings in different variations of OL systems, Part I: Deterministic systems, *Acta Inform.*, v. 4, 1974, pp. 87—106.
[7] ROZENBERG, G. and A. SALOMAA, *The mathematical theory of L systems*, Academic Press, New York, 1980.
[8] ROZENBERG, G. and R. VERRAEDT, Synchronized, desynchronized and coordinated EOL systems, *Inform. and Control*, v. 46, 1980, pp. 156—185.
[9] SALOMAA, A., *Formal languages*, Academic Press, New York, 1973.

(Received Sept. 14, 1981)

An algebraic definition of attributed transformations

By M. BARTHA

1. Magmoids and rational theories

The concept of magmoid was introduced in [1]. A magmoid $M=(\{M_s|s\in S\}, \cdot, \otimes, e, e_0)$, is a many sorted algebra with sorting set S, the set of all pairs of non-negative integers. Further on we shall write M_q^p instead of $M_{\langle p,q\rangle}$. Binary operations $\cdot$ and $\otimes$ are called composition and tensor product, respectively. The following axioms must be valid in M:

(i) $\cdot: M_q^p\times M_r^q\to M_r^p$ is associative.

(ii) $\otimes: M_{q_1}^{p_1}\times M_{q_2}^{p_2}\to M_{q_1+q_2}^{p_1+p_2}$ is associative.

(iii) $(a_1\cdot b_1)\otimes(a_2\cdot b_2)=(a_1\otimes a_2)\cdot(b_1\otimes b_2)$ for all composable pairs $\langle a_1, b_1\rangle$, $\langle a_2, b_2$.

(iv) $e\in M_1^1$, $e_0\in M_0^0$, and if e_n denotes $\underbrace{e\otimes\ldots\otimes e}_{n \text{ times}}$ $(n\geqq 1)$, then for each $p\geqq 0$, $q\geqq 0$, $a\in M_q^p$: $e_p\cdot a=a\cdot e_q=a\otimes e_0=e_0\otimes a=a$.

An element $a\in M_q^p$ will often be denoted by $a: p\to q$ if M is understood.

Let $\Sigma=\bigcup\limits_{n\geqq 0}\Sigma_n$ be a finite ranked alphabet, and define the structure $T(\Sigma)=$ $=(\{T(\Sigma)_q^p\ p, q\geqq 0\}, \cdot, \otimes, e, e_0)$ as follows:

For arbitrary $p\geqq 0$ and $q\geqq 0$, $T(\Sigma)_q^p=\{\langle q; t_1, \ldots, t_p\rangle|$ for each $1\leqq i\leqq p$, t_i is a finite Σ-tree over the variables $x_1, \ldots, x_q\}$. $\langle q;\rangle\in T(\Sigma)_q^0$ will be denoted by 0_q.

$$\langle q; t_1, \ldots, t_p\rangle\cdot\langle r; u_1, \ldots, u_q\rangle = \langle r; t_1[u_1, \ldots, u_q], \ldots, t_p[u_1, \ldots, u_q]\rangle,$$

where $[\ldots]$ denotes the composition of trees;

$$\langle q_1; t_1, \ldots, t_{p_1}\rangle\otimes\langle q_2; u_1, \ldots, u_{p_2}\rangle = \langle q_1+q_2; t_1, \ldots, t_{p_1}, u_1', \ldots, u_{p_2}'\rangle,$$

where $u_i'=u_i[x_{q_1+1}, \ldots, x_{q_1+q_2}]$; $e=\langle 1; x_1\rangle$, $e_0=0_0$.

We shall omit the component q of $\langle q; t_1, \ldots, t_p\rangle$ if it is understood. Moreover, we leave $\langle\ldots\rangle$ if $p=1$. It is known that $T(\Sigma)$ is a magmoid. $\tilde{T}(\Sigma)$ is a submagmoid of $T(\Sigma)$ such that $t=\langle q; t_1, \ldots, t_p\rangle\in\tilde{T}(\Sigma)_q^p$ if and only if the sequence of variables labeling the leaves of $t_1, \ldots, t_p$, read from left to the right, is exactly $x_1, \ldots, x_q$. $\tilde{T}(\Sigma)$ is the free magmoid generated by Σ, that is, every ranked alphabet map $h: \Sigma\to M^1$ into a magmoid M has a unique homomorphic extension $\bar{h}: \tilde{T}(\Sigma)\to M$. (Viewing $\sigma\in\Sigma_n$ as $\langle n; \sigma(x_1, \ldots, x_n)\rangle\in\tilde{T}(\Sigma)_n^1$).

Another important magmoid is θ, in which θ_q^p is the set of all mappings of $[p]=\{1, ..., p\}$ into $[q]$. Composition is that of mappings, and for $\vartheta_i \in \theta_{q_i}^{p_i}$, $i=1, 2$

$$\vartheta_1 \otimes \vartheta_2(j) = \begin{cases} \vartheta_1(j) & \text{if } j \in [p_1], \\ \vartheta_2(j-p_1)+q_1 & \text{if } p_1 < j \leqq p_2. \end{cases}$$

e and e_0 are the unique elements of θ_1^1 and θ_0^0, respectively. e_n will be denoted by id_n if $n \geqq 1$. The elements of θ are usually called torsions or base moprhisms.

A magmoid is called projective if it contains a submagmoid isomorphic to θ and every $a: p \to q$ is uniquely determined by its "projections", i.e. by the sequence $\langle \pi_p^i \cdot a | 1 \leqq i \leqq p \rangle$. π_p^i denotes the isomorphic image of the map $\pi_p^i: [1] \to [p]$ that picks out the integer i of $[p]$. $T(\Sigma)$ is projective, and it is the free projective magmoid generated by Σ. $P_F T(\Sigma)$ will denote the magmoid in which $(P_F T(\Sigma))_q^p = \{q; A_1, ..., ..., A_p\rangle |$ for each $i \in [p]$, A_i is a finite set of Σ-trees over the variables $x_1, ..., x_q\}$. (For the interpritation of the operations see [2].) $P_F T(\Sigma)$ is also projective. Let M be a projective magmoid, $a_1, ..., a_p \in M_q^1$. $\prec a_1, ..., a_p \succ$ will denote the unique element of M_q^p whose sequence of projections is $\langle a_1, ..., a_p \rangle$. This source-tupling can be viewed as a derived operation in M, and it can be extended as follows. Let $a_1: p_1 \to q$, $a_2: p_2 \to q$. Then $\prec a_1, a_2 \succ = \prec \pi_{p_1}^1 \cdot a_1, ..., \pi_{p_1}^{p_1} \cdot a_1, \pi_{p_2}^1 \cdot a_2, ..., \pi_{p_2}^{p_2} \cdot a_2 \succ$.

Rational theories were introduced in [3], based on the concept of algebraic theory. However, the only difference between nondegenerate algebraic theories and projective magmoids is that in algebraic theories source-tupling is a basic operation (and tensor product is a derived one). So, if we introduce rational theories by means of projective magmoids, we get a definition equivalent to the original one excluding the trivial degenerate rational theory.

A rational theory is also a many sorted algebra $R = (\{R_q^p | p, q \geqq 0\}, \cdot, \otimes, e, e_0, {}^+)$, where, apart from ${}^+$, R is a projective magmoid, the sets R_q^p are partially ordered, and ${}^+: R_{p+q}^p \to R_q^p$ is a new operation. For $f: p \to p+q$, f^+ is the least fixpoint of f, and some further conditions must hold concerning the ordering and the operations, that we do not list here.

Add a new symbol $\perp$ with rank 0 to Σ, to get the ranked alphabet $\Sigma_\perp$. There exists a rational theory $T_\infty(\Sigma)$ for which $T_\infty(\Sigma)_q^p = \{\langle q; t_1, ..., t_p \rangle |$ for each $i \in [p]$, t is a possibly infinite $\Sigma_\perp$-tree over the variables $x_1, ..., x_q\}$. For the interpretation of the operations, see [3]. It is known that $R(\Sigma)$, the free rational theory generated by Σ, is the smallest subtheory of $T_\infty(\Sigma)$ that contains $T(\Sigma)$ as a submagmoid.

Let $q \geqq 0$, $X_q = \{x_1, ..., x_q\}$, $\chi_q: \Sigma \to (\Sigma \cup X_q)^*$ such that for each $\sigma \in \Sigma_n$, length $(\chi_q(\sigma)) = n$. An infinite tree $t \in R(\Sigma)_q^p$ is called local of type χ_q if the following holds. If an interior node of t is labeled by $\sigma \in \Sigma_n$, then its direct descendants are labeled by $\chi_q(\sigma)$. If so, we will denote t by (ω, χ_q), where $\omega = \mathrm{root}(t) \in (\Sigma \cup X_q)^p$. $\mathrm{Rec}(\Sigma)$ will denote the smallest rational theory in $PT(\Sigma)$ that contains $P_F T(\Sigma)$ as a submagmoid.

2. The magmoid $R(k, l)$

Definition 2.1. Let R be a rational theory, $k \geqq 1, l \geqq 0$ integers. Define $R(k, l) = (\{R(k, l)_q^p \mid p, q \geqq 0\}, \cdot, \otimes, e, e_0)$ to be the following structure:

(i) $R(k, l)_q^p = R_{k\cdot q + l\cdot p}^{k\cdot p + l\cdot q}$;

(ii) if $a \in R(k, l)_q^p$, $b \in R(k, l)_r^q$, then

$$a \cdot b = \langle \mu^{k\cdot p}, \nu_{l\cdot r} \rangle \cdot \langle a \cdot \vartheta_{p,q,r}, b \cdot \psi_{p,q,r} \rangle^+,$$

where

$$\mu_m^n (= \mu^n \text{ if } m \text{ is understood}) = \mathrm{id}_n \otimes 0_m \in \theta_{n+m}^n,$$

$$\nu_m^n (= \nu_m \text{ if } n \text{ is understood}) = 0_n \otimes \mathrm{id}_m \in \theta_{n+m}^m,$$

$$\vartheta_{p,q,r} = \nu_{k\cdot q}^{k\cdot p + l\cdot q} \otimes \nu_{l\cdot p}^{(k+l)\cdot r},$$

$$\psi_{p,q,r} = 0_{k\cdot p} \otimes \langle \nu_{k\cdot r}^{(k+l)\cdot q + l\cdot r}, \mu_{k\cdot q + (k+l)\cdot r}^{l\cdot q} \rangle \otimes 0_{l\cdot p}.$$

See also Fig. 1.

(iii) if $a \in R(k, l)_{q_1}^{p_1}$, $b \in R(k, l)_{q_2}^{p_2}$, then

$$a \otimes b = \langle \mu_{l\cdot q_1}^{k\cdot p_1} \otimes \mu_{l\cdot q_2}^{k\cdot p_2}, \nu_{l\cdot q_1}^{k\cdot p_1} \otimes \nu_{l\cdot q_2}^{k\cdot p_2} \rangle \cdot (a \otimes b) \cdot \langle \mu_{l\cdot p_1}^{k\cdot q_1} \otimes \mu_{l\cdot p_2}^{k\cdot q_2}, \nu_{l\cdot p_1}^{k\cdot q_1} \otimes \nu_{l\cdot p_2}^{k\cdot q_2} \rangle^{-1}.$$

(iv) $e = \mathrm{id}_{k+l}$, $e_0 = 0_0$.

(We shall never add any distinctive mark to the sign of the operations when working in different magmoids in the same time, because only one interpretation is reasonable anywhere in the context.)

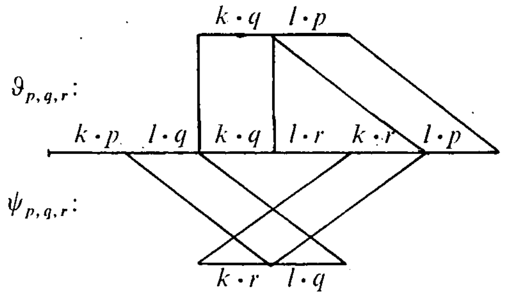


Fig. 1

Theorem 2.2. $R(k, l)$ is a magmoid.

Proof. All the requirements can be proved by the same method, so we only show the associativity of composition. Let

$$\begin{aligned} a &= \langle \underline{a}_1, \ldots, \underline{a}_{k\cdot p}, \bar{a}_1, \ldots, \bar{a}_{l\cdot q} \rangle \in R(k, l)_q^p, \\ b &= \langle \underline{b}_1, \ldots, \underline{b}_{k\cdot q}, \bar{b}_1, \ldots, \bar{b}_{l\cdot r} \rangle \in R(k, l)_r^q, \\ c &= \langle \underline{c}_1, \ldots, \underline{c}_{k\cdot r}, \bar{c}_1, \ldots, \bar{c}_{l\cdot s} \rangle \in R(k, l)_s^r. \end{aligned} \tag{1}$$

We must prove that $(a\cdot b)\cdot c=a\cdot(b\cdot c)$. Both sides of this equation can be considered as a polynomial in R over the variables $\underline{a}_i, \bar{a}_j, \ldots, \underline{c}_i, \bar{c}_j$. Since R is arbitrary, we have to show that these polynomials are identical. Let Σ be the smallest finite ranked alphabet satisfying the following conditions:

(i) for arbitrary $i\in[k\cdot p]$ and $j\in[l\cdot q]$, $\underline{A}_i, \bar{A}_j\in\Sigma_{k\cdot q+l\cdot p}$,
(ii) for arbitrary $i\in[k\cdot q]$ and $j\in[l\cdot r]$, $\underline{B}_i, \bar{B}_j\in\Sigma_{k\cdot r+l\cdot q}$,
(iii) for arbitrary $i\in[k\cdot r]$ and $j\in[l\cdot s]$, $\underline{C}_i, \bar{C}_j\in\Sigma_{k\cdot s+l\cdot r}$.

Change the small letters to capital ones in (1), to obtain the elements A, B, C of $R(\Sigma)$. Clearly, it is enough to show that $(A\cdot B)\cdot C=A\cdot(B\cdot C)$ holds in $R(\Sigma)(k, l)$. However, it is easy to check that $(A\cdot B)\cdot C=A\cdot(B\cdot C)=(\omega, \chi_n)$, where $n=k\cdot s+l\cdot p$ and

$$\omega = \langle\underline{A}_1, \ldots, \underline{A}_{k\cdot p}, \bar{C}_1, \ldots, \bar{C}_{l\cdot s}\rangle,$$

$$\chi_n(\underline{A}_i) = \chi_n(\bar{A}_j) = \langle\underline{B}_1, \ldots, \underline{B}_{k\cdot q}, x_{k\cdot s+1}, \ldots, x_{k\cdot s+l\cdot p}\rangle,$$

$$\chi_n(\underline{B}_i) = \chi_n(\bar{B}_j) = \langle\underline{C}_1, \ldots, \underline{C}_{k\cdot r}, \bar{A}_1, \ldots, \bar{A}_{l\cdot q}\rangle,$$

$$\chi_n(\underline{C}_i) = \chi_n(\bar{C}_j) = \langle x_1, \ldots, x_{k\cdot s}, \bar{B}_1, \ldots, \bar{B}_{l\cdot r}\rangle$$

for any appropriate choise of the integers i and j.

Let $\xi\colon R\to R'$ be a homomorphism between rational theories. Clearly, ξ defines a homomorphism $\xi(k, l)\colon R(k, l)\to R'(k, l)$, and so the operator (k, l) becomes a functor.

3. Attributed transformations

Definition 3.1. An attributed transducer is a 6-tuple $\mathfrak{A}=(\Sigma, R, k, l, h, S)$, where

(i) Σ is a finite ranked alphabet, $S\notin\Sigma$;
(ii) R is a rational theory, $k\geqq 1, l\geqq 0$ are integers;
(iii) $h\colon \Sigma_S\to R(k, l)$ is a ranked alphabet map, where $\Sigma_S=\Sigma\cup\{S\}$ with S having rank 1, and $h(S)=a\otimes 0_l$ for some $a\in R_k^{k+l}$. We say that $h(S)$ is a synthesizer.

$\tau_{\mathfrak{A}}\colon \tilde{T}(\Sigma)_0^1\to R_0^1$, the transformation induced by $\mathfrak{A}$, is the following function: $\tau_{\mathfrak{A}}(t)=a$, where $\pi_k^1\cdot h(S(t))=a\otimes 0_l$. It is clear that $\tau_{\mathfrak{A}}(t)$ is uniquely determined by this imlicit form. (As it is usual, we denoted the unique homomorphic extension of h also by h.)

Definition 3.2. An attributed tree transducer is a 6-tuple $\mathfrak{A}=(\Sigma, \Delta, k, l, h, S)$, where Σ, k, l and S are as in the previous definition, Δ is a finite ranked alphabet, $h\colon \Sigma_S\to P_F T(\Delta)$ is such that $h((\Sigma_S)_n)\subseteqq P_F T(\Delta)_{k\cdot n+l}^{k+l\cdot n}$ and $h(S)\in P_F T(\Delta)_k^{k+l}$. To define the transformation $\tau_{\mathfrak{A}}$, consider the attributed transducer $\mathfrak{B}=(\Sigma, \mathrm{Rec}(\Delta), k, l, h, S)$. $\mathfrak{B}$ is correct, since $P_F T(\Delta)\subseteqq \mathrm{Rec}(\Delta)$ and $h(S)$ is a synthesizer. Now

$$\tau_{\mathfrak{A}} = \{\langle t, u\rangle | t\in\tilde{T}(\Sigma)_0^1,\quad u\in\tau_{\mathfrak{B}}(t)\}.$$

$\mathfrak{A}$ is called deterministic if for arbitrary $n\geqq 0$ and $\sigma\in(\Sigma_S)_n$ all the components of $h(\sigma)$ contain at most one element.

Example 3.3. Let $k=l=2$,

$$\Sigma=\Sigma_0\cup\Sigma_1,\ \Sigma_0=\{\bar{a}\},\ \Sigma_1=\{f\},\ \Delta=\Delta_0\cup\Delta_1,\ \Delta_0=\{\bar{a}\},\ \Delta_1=\{f,g\},$$

$$h(f)=\langle 4; f(x_1), f(x_2), g(x_3), g(x_4)\rangle,\ h(\bar{a})=\langle 2; x_1, x_2\rangle,\ h(S)=\langle 4; x_1, \bar{a}, x_2, \bar{a}\rangle.$$

(Braces enclosing singletons are omitted.) Then $\mathfrak{A}=(\Sigma, \Delta, k, l, h, S)$ is a deterministic attributed tree transducer, and it is easy to see that for all $n\geqq 0$

$$h(f^n(x_1)) = \langle 4; f^n(x_1), f^n(x_2), g^n(x_3), g^n(x_4)\rangle.$$

Hence, $h(f^n(\bar{a}))=\langle 2; f^n g^n(x_1), f^n g^n(x_2)\rangle$, and

$$\tau_{\mathfrak{A}} = \{\langle f^n(\bar{a}), f^n g^n f^n g^n(\bar{a})\rangle | n\geqq 0\}.$$

Definition 3.2 might be interpreted as follows. Let $t\in\tilde{T}(\Sigma)^1$, α a node in t having some label $\sigma\in\Sigma_n$. A component of $h(\sigma)$ describes how to compute the value of a synthesized attribute of α (the first k components), or an inherited attribute of an immediate descendant of α (the last $l\cdot n$ components) as a function (polynomial) of the synthesized attributes of the immediate descendants (the variables $x_1, \ldots, x_{k\cdot q}$) and the inherited attributes of α itself (the variables $x_{k\cdot q+1}, \ldots, x_{k\cdot q+l}$). The role of the synthesizer $h(S)$ is to produce the final result of the computation.

It will be convenient to identify the nodes of a tree $t\in\tilde{T}(\Sigma)^1_q$ with the set $\operatorname{nds}(t)\subseteq \mathbf{N}^*\times(\Sigma\cup X_q)$, and the leaves of t with $\operatorname{lvs}(t)\subseteq\mathbf{N}^*\times X_q$ as follows:

(i) if $t=x_1$, then $\operatorname{nds}(t)=\operatorname{lvs}(t)=\{\langle\lambda, x_1\rangle\}$;

(ii) if $t=t_0\cdot(\mathrm{id}_{p-1}\otimes\sigma(x_1, \ldots, x_n)\otimes\mathrm{id}_{q-p})$ with $t_0\in\tilde{T}(\Sigma)^1_q$, $q\geqq 1$, $p\in[q]$, $n\geqq 0$, $\sigma\in\Sigma_n$, then $\operatorname{nds}(t)=\bigcup_{i=1}^{5} V_i$, where

$V_1=\{\langle w, x_j\rangle | j\in[p-1]$ and $\langle w, x_j\rangle\in\operatorname{lvs}(t_0)\}$,
$V_2=\{\langle w, x_j\rangle | j\geqq p+n$ and $\langle w, x_{j-n+1}\rangle\in\operatorname{lvs}(t_0)\}$,
$V_3=\{\langle wj, x_{p+j-1}\rangle | j\in[n]$ and $\langle w, x_p\rangle\in\operatorname{lvs}(t_0)\}$,
$V_4=\operatorname{nds}(t_0)\setminus\operatorname{lvs}(t_0)$,
$V_5=\{\langle w, \sigma\rangle\}$, where $\langle w, x_p\rangle\in\operatorname{lvs}(t_0)$.
$\operatorname{lvs}(t)=V_1\cup V_2\cup V_3$.

It is easy to verify that $\operatorname{nds}(t)$ and $\operatorname{lvs}(t)$ are uniquely defined by the above construction, and for each $w\in\mathbf{N}^*$ there exists at most one $\alpha\in\operatorname{nds}(t)$ having w as its first component. Clearly, $\|\operatorname{nds}(t)\|=r(t)$, the number of nodes in t.

Let $\mathfrak{A}=(\Sigma, \Delta, k, l, h, S)$ be an attributed tree transducer, fixed in the rest of the paper, $t\in\tilde{T}(\Sigma)^1_q$,

$$Z_t = \{x(\alpha, i), y(\alpha, m) | \alpha\in\operatorname{nds}(t), i\in[k], m\in[l]\}$$

a set of variable symbols. Construct a system $E_{t,h}$ of nondeterministic Δ-equations over the variables Z_t as follows

$$E_{t,h}=\{E_{x,h}(\alpha, i) | \alpha\in\operatorname{nds}(t)\setminus\operatorname{lvs}(t), i\in[k]\}\cup$$
$$\cup\{E_{y,h}(\alpha, m) | \alpha\in\operatorname{nds}(t)\setminus\{\langle\lambda, \operatorname{root}(t)\rangle\}, m\in[l]\},$$

where

(i) if $\alpha=\langle w, \sigma\rangle$ with $\sigma\in\Sigma_n$ and

$$h(\sigma) = \langle T_1, \ldots, T_k, Q_1, \ldots, Q_{l\cdot n}\rangle, \tag{2}$$

then the equation $E_x(\alpha, i)$ is of the form

$$x(\alpha, i) = T_i[x_{k\cdot(r-1)+p} \leftarrow x(\alpha_r, p), x_{k\cdot n+s} \leftarrow y(\alpha, s) | p\in[k], r\in[n], s\in[l]],$$

where $\leftarrow$ denotes variable substitution, $\alpha_r \in \text{nds}(t)$ is the unique node having wr as first component. (We omitted the index h, which is fixed.)

(ii) If $\alpha=\langle wj, a\rangle$ with $a\in\Sigma\cup X_q$, then consider the unique node $\bar\alpha=\langle w, \sigma\rangle$, where $\sigma\in\Sigma_n$, $n\geqq j$, and the nodes $\bar\alpha_r$, $r\in[n]$. (Naturally $\bar\alpha_j=\alpha$.) Let $h(\sigma)$ be as (2) above. Then the equation $E_y(\alpha, m)$ looks as

$$y(\alpha, m) = Q_{l\cdot(j-1)+m}[x_{k\cdot(r-1)+p} \leftarrow x(\bar\alpha_r, p), x_{k\cdot n+s} \leftarrow y(\bar\alpha, s) | p\in[k], r\in[n], s\in[l]].$$

The variables

$$Z_t^1 = \{x(\alpha, i) | \alpha\in\text{lvs}(t), i\in[k]\} \cup \{y(\langle\lambda, \text{root}(t)\rangle, m) | m\in[l]\}$$

do not occur on the left-hand side of these equations, so they are considered as parameters. On the other hand, the variables

$$Z_t^2 = \{x(\langle\lambda, \text{root}(t)\rangle, i) | i\in[k]\} \cup \{y(\alpha, m) | \alpha\in\text{lvs}(t), m\in[l]\}$$

do not occur on the right-hand side of the equations. If we identify the elements of Z_t with the variables $x_1, \dots, x_{(k+l)\cdot r(t)}$ by a bijection $\varepsilon_t\colon Z_t\to[(k+l)\cdot r(t)]$ so that the variables Z_t^1 get the highest and Z_t^2 the lowest indices, we get an $\omega'(t, \varepsilon_t)$: $(k+l)\cdot r(t)-(k\cdot q+l)\to(k+l)\cdot r(t)\in\text{Rec}(\Delta)$ for which $\omega'(t, \varepsilon_t)=0_{k+l\cdot q}\otimes\omega(t, \varepsilon_t)$ and $(\omega'(t, \varepsilon_t))^+=E_t^+$ (with respect to ε_t). E_t^+ denotes the solution of E_t.

Lemma 3.4. Let R be a rational theory, $k\geqq1, l\geqq0, q\geqq1, n\geqq0, p\in[q]$ integers, $a\in R(k, l)_q^1, b\in R(k, l)_n^1$. Then

$$a\cdot(e_{p-1}\otimes b\otimes e_{q-p}) = \mu^{k+l\cdot(q-1+n)}\cdot\big(0_{k+l\cdot(q-1+n)}\otimes \otimes(\varrho_{q,p,n}\cdot \lessdot a\cdot\eta_{q,p,n}, b\cdot\zeta_{q,p,n} \gtrdot)\big)^+, \tag{3}$$

where

$$\varrho_{q,p,n} = \lessdot \mu^{k+l\cdot(p-1)}, \nu_{l\cdot n}, 0_{k+l\cdot p}\otimes\mu^{l\cdot(q-p)+k}, 0_{k+l\cdot(p-1)}\otimes\mu^l \gtrdot :$$

$$k+l\cdot(p-1)+l\cdot n+l\cdot(q-p)+k+l \to k+l\cdot(p-1)+l+l\cdot(q-p)+k+l\cdot n,$$

$$\eta_{q,p,n} = \lessdot \nu^{k+l}_{k\cdot(p-1)}, \mu^k_{l+k\cdot(p+1)} \gtrdot \otimes \nu^{k\cdot n}_{k\cdot(q-p)+l}:$$

$$k\cdot(p-1)+k+k\cdot(q-p)+l \to k+l+k\cdot(p-1)+k\cdot n+k\cdot(q-p)+l,$$

$$\zeta_{q,p,n} = 0_k\otimes \lessdot \nu^{l+k\cdot(p-1)}_{k\cdot n}, \mu^l_{k\cdot(p-1+n)} \gtrdot \otimes 0_{k\cdot(q-p)+l}:$$

$$k\cdot n+l \to k+l+k\cdot(p-1)+k\cdot n+k\cdot(q-p)+l.$$

(The left-hand side of (3) is a polynomial in $R(k, l)$, while the right-hand side is a polynomial in R.)

Instead of presenting a complete proof we only remark that it would be enough to prove the lemma for one special free rational theory, analogously to the proof of Theorem 2.2. Then the proof reduces to an easy computation that we do not preform here. The following lemma can be proved in the same way

Lemma 3.5. Let R be a rational theory; $n_1, n_2, n_3, p_1, p_2, p_3, m, r, s$ nonnegative integers,

$$f\colon n_1+m+n_3+s \to s+p_1+r+p_3 \in R,$$
$$g\colon r+n_2 \to p_2+m \in R.$$

Then

$$\mu^{n_1+n_2+n_3} \cdot \big(0_{n_1+n_2+n_3} \otimes (\varrho \cdot \nless \mu^{n_1+m+n_3} \cdot (0_{n_1+m+n_3} \otimes f)^+ \cdot \eta,\ g \cdot \zeta)\big)^+ =$$
$$= \mu^{n_1+n_2+n_3} \cdot \big(0_{n_1+n_2+n_3} \otimes (\varrho_s \cdot \nless f \cdot \eta_s,\ g \cdot \zeta_s \ngtr)\big)^+, \tag{4}$$

where

$$\varrho = \nless \mu^{n_1}, \nu_{n_2}, 0_{n_1+m} \otimes \mu^{n_3+r}, 0_{n_1} \otimes \mu^m \ngtr : n_1+n_2+n_3+r+m \to n_1+m+n_3+r+n_2,$$

$$\varrho_s = \nless \mu^{n_1}, \nu_{n_2}, 0_{n_1+m} \otimes \mu^{n_3+s+r}, 0_{n_1} \otimes \mu^m \ngtr :$$
$$n_1+n_2+n_3+s+r+m \to n_1+m+n_3+s+r+n_2,$$

$$\eta = \nless \nu_{p_1}^{r+m}, \mu^r_{m+p_1} \ngtr \otimes \nu^{p_2}_{p_3}\colon p_1+r+p_3 \to r+m+p_1+p_2+p_3, \qquad \eta_s = \mathrm{id}_s \otimes \eta,$$

$$\zeta = 0_r \otimes \nless \nu^{m+p_1}_{p_2}, \mu^m_{p_1+p_2} \ngtr \otimes 0_{p_3}\colon\ p_2+m \to r+m+p_1+p_2+p_3, \zeta_s = 0_s \otimes \zeta.$$

Lemma 3.6. Let $q \geqq 0, t \in \tilde{T}(\Sigma)^1_q, t \neq x_1$. There exists a bijection $\varepsilon_t\colon Z_t \to$ $\to [(k+l) \cdot r(t)]$ such that

(A) for arbitrary $i \in [k], j \in [q], m \in [l]$ and appropriate $w \in \mathbf{N}^*$

$\varepsilon_t(x(\langle \lambda, \mathrm{root}\,(t)\rangle, i)) = i,$
$\varepsilon_t(y(\langle w, x_j\rangle, m)) = k + l \cdot (j-1) + m,$
$\varepsilon_t(x(\langle w, x_j\rangle, i)) = r(t) - (k \cdot q + l) + k \cdot (j-1) + i,$
$\varepsilon_t(y(\langle \lambda, \mathrm{root}\,(t)\rangle, m)) = r(t) - l + m;$

(B) $\mu^{k+l \cdot q} \cdot (0_{k+l \cdot q} \otimes \omega(t, \varepsilon_t))^+ = h(t).$

Proof. If $t = \sigma(x_1, \ldots, x_q)$ for some $\sigma \in \Sigma_q$, then ε_t is completely determined by (A). Obviusly, $\omega(t, \varepsilon_t) = h(t)$, so (B) is trivially satisfied. Now let $t = t_0 \cdot (\mathrm{id}_{p-1} \otimes$ $\otimes \sigma(x_1, \ldots, x_n) \otimes \mathrm{id}_{q-p})$, where $q \geqq 1, p \in [q], t_0 \in \tilde{T}(\Sigma)^1_q, t_0 \neq x_1, n \geqq 0, \sigma \in \Sigma_n$, and suppose the lemma is true for t_0. Let $s = (k+l) \cdot \|\mathrm{nds}\,(t_0) \setminus \mathrm{lvs}\,(t_0)\| - (k+l)$. Using the sets $V_1, \ldots, V_5$ introduced in the construction of nds (t), we define ε_t as follows. If $\alpha \in V_1, \alpha = \langle w, x_j\rangle$, then for arbitrary $i \in [k]$ and $m \in [l]$

$\varepsilon_t(x(\alpha, i)) = k + l \cdot (p-1) + l \cdot n + l \cdot (q-p) + s + k + l + k \cdot (j-1) + i,$
$\varepsilon_t(y(\alpha, m)) = k + l \cdot (j-1) + m.$

If $\alpha \in V_2, \alpha = \langle w, x_j\rangle$, then

$\varepsilon_t(x(\alpha, i)) =$
$= k + l \cdot (p-1) + l \cdot n + l \cdot (q-p) + s + k + l + k \cdot (p-1) + k \cdot n + k \cdot (j-1) + i,$
$\varepsilon_t(y(\alpha, m)) = k + l \cdot (p-1) + l \cdot n + l \cdot (j-1) + m.$

If $\alpha \in V_3, \alpha = \langle wj, x_{p+j-1}\rangle$, then

$\varepsilon_t(x(\alpha, i)) = k + l \cdot (p-1) + l \cdot n + l \cdot (q-p) + s + k + l + k \cdot (p-1) + k \cdot (j-1) + i,$
$\varepsilon_t(y(\alpha, m)) = k + l \cdot (p-1) + l \cdot (j-1) + m.$

If $\alpha \in V_4$ and $\alpha = \langle \lambda, \mathrm{root}(t) \rangle$, then

$\varepsilon_t(x(\alpha, i)) = i$,

$\varepsilon_t(y(\alpha, m)) =$

$= k + l\cdot(p-1) + l\cdot n + l\cdot(q-p) + s + k + l + k\cdot(p-1) + k\cdot n + k\cdot(q-p) + m$,

else

$\varepsilon_t(x(\alpha, i)) = \varepsilon_{t_0}(x(\alpha, i)) + l\cdot n - l$,

$\varepsilon_t(y(\alpha, m)) = \varepsilon_{t_0} y(\alpha, m)) + l\cdot n - l$.

If $\alpha \in V_5$, then $\alpha = \langle w, \sigma \rangle$ and

$\varepsilon_t(x(\alpha, i)) = k + l\cdot(p-1) + l\cdot n + l\cdot(q-p) + s + i$,

$\varepsilon_t(y(\alpha, m)) = k + l\cdot(p-1) + l\cdot n + l\cdot(q-p) + s + k + m$,

It is easy to see that ε_t is a bijection and satisfies (A). To prove (B), apply Lemma 3.6 for $R = \mathrm{Rec}(\Delta), f = \omega(t_0, \varepsilon_{t_0}), g = h(\sigma), n_1 = k + l\cdot(p-1), n_2 = l\cdot n, n_3 = l\cdot(q-p), m = l, r = k, p_1 = k\cdot(p-1), p_2 = k\cdot n, p_3 = k\cdot(q-p)$, (and $s = s$). Observe that $\varrho_s \cdot \langle\!\langle f\cdot\eta_s, g\cdot\zeta_s \rangle\!\rangle = \omega(t, \varepsilon_t)$, and the right-hand side of (4) equals to $\mu^{k+l\cdot(q-1+n)}\cdot (0_{k+l\cdot(q-1+n)} \otimes \omega(t, \varepsilon_t))^+$. So we must prove that the left-hand side of (4) equals to $h(t)$. By the inductive hypothesis $\mu^{n_1+m+n_3}\cdot(0_{n_1+m+n_3} \otimes f)^+ = h(t_0)$, so we have to see that

$$h(t) = h(t_0)\cdot(e_{p-1} \otimes h(\sigma) \otimes e_{q-p}) =$$
$$= \mu^{k+l\cdot(q-1+n)}\cdot(0_{k+l\cdot(q-1+n)} \otimes (\varrho \cdot \langle\!\langle h(t_0)\cdot\eta, h(\sigma)\cdot\zeta \rangle\!\rangle))^+.$$

This is exactly the statement of Lemma 3.5, so we are through.

Replacing Σ by Σ_s we get

Corollary 3.7. For each $t \in \tilde{T}(\Sigma)_0^1$, $\tau_{\mathfrak{A}}(t)$ equals to the $x(\langle \lambda, S \rangle, 1)$ component of $E^+_{S(t)}$.

This result links our work to [4], where the same technic was used to define the semantics of attribute grammars.

Now we turn our attention to the domain of $\tau_{\mathfrak{A}}$, that is the set $D\tau_{\mathfrak{A}} = \{t \in \tilde{T}(\Sigma)_0^1 |$ for some $u \in T(\Delta)_0^1 \langle t, u \rangle \in \tau_{\mathfrak{A}}\}$. Let $G(k, l)$ be the following finite set

$G(k, l) = \{(G; V_{1,1}, V_{1,2}, V_{2,1}, V_{2,2}) | G = (V, E)$ is a directed acyclic bipartite graph, and

(i) $V = V_1 \cup V_2$, $V = [k+l]$, $V_1 = [k]$, $V_2 = V \setminus V_1$, $E = E_1 \cup E_2$, $\mathrm{dom}(E_1) \subseteqq V_1$, $\mathrm{dom}(E_2) \subseteqq V_2$;

(ii) $V_1 = V_{1,1} \cup V_{1,2}$, $V_{1,1} \cap V_{1,2} = \emptyset$; $V_2 = V_{2,1} \cup V_{2,2}$, $V_{2,1} \cap V_{2,2} = \emptyset$;

(iii) for each $j \in V_{2,1}$ there exists an $i \in V_{1,1}$ such that $\langle i, j \rangle \in E_1$ and the vertices $V_{1,2} \cup V_{2,2}$ are isolated.$\}$

(A vertex is called isolated if there are no edges entering or leaving it.)

We construct a finite state top-down tree automaton $\mathfrak{B}$ that operates nondeterministically on $\tilde{T}(\Sigma)^1$ with states $A = G(k, l)$. Let $t \in D\tau_{\mathfrak{A}}$, α a node in t and suppose that $\mathfrak{B}$ passes through α in state $(G; V_{1,1}, \ldots, V_{2,2})$. The synthesized (inherited) attributes of α are represented as the nodes in V_1 (V_2, respectively). $V_{1,1} \cup V_{2,1}$ will contain the indices of those attributes that take part in the computation of $\tau_{\mathfrak{A}}(t)$. The edges of G will show how these "useful" attributes depend on each other. A similar construction was used in [5] for testing circularity of attribute grammars.

The fact that, starting from state a_0, $\mathfrak{B}$ is able to reach the vector of states $\langle a_1, \dots, a_q\rangle$ on input $t \in \tilde{T}(\Sigma)^1_q$ will be denoted by $a_0 t \overset{*}{\underset{\mathfrak{B}}{\vdash}} t(a_1, \dots, a_q)$. If for some $\sigma \in \Sigma_q$, $t = \sigma(x_1, \dots, x_q)$, we simply write $a_0 \sigma \overset{*}{\underset{\mathfrak{B}}{\vdash}} \sigma(a_1, \dots, a_q)$.

Let $\sigma \in (\Sigma_S)_n$, $h(\sigma) = \langle T_1, \dots, T_{k+l\cdot n}\rangle$, $I_\sigma = \{i \in [k+l\cdot n] \mid T_i = \emptyset\}$. The set of alternatives of σ is

$$A[\sigma] = \{\langle t_1, \dots, t_{k+l\cdot n}\rangle \mid \text{ if } i \in I_\sigma, \text{ then } t_i = \bot, \text{ else } t_i \in T_i\}.$$

We say that $c \in A[S]$ realizes the initial state $a = (G_c;\ V^c_{1,1}, \dots, V^c_{2,2})$ if the following conditions are satisfied:

(a) If $j \in V^c_{2,1}$, then $\langle j, i\rangle \in E^c_2$ if and only if x_i occurs in t_j.

(b) $V^c_{1,1} \supseteq Q = \{i \in [k] \mid x_i$ occurs in $t_1\}$, and for each $i \in V^c_{1,1} \setminus Q$ there exists an $i' \in Q$ such that $i' \overset{+}{\underset{G_c}{\vdash}} i$. $\overset{+}{\underset{G_c}{\vdash}}$ denotes the transitive closure of $\underset{G_c}{\vdash} = E^c$.

(c) $V^c_{2,2} \supseteq \{j > k \mid j \in I_S\}$.

Define the set of initial states of $\mathfrak{B}$ as $A_0 = \{a \in A \mid a$ is realized by some $c \in A[S]\}$.

Let $n \geqq 0$, $\sigma \in \Sigma_n$, $a_0, \dots, a_n \in A$, $a_m = (G_m;\ V^m_{1,1}, \dots, V^m_{2,2})$ for each $0 \leqq m \leqq n$, and $c = \langle t_1, \dots, t_{k+l\cdot n}\rangle \in A[\sigma]$. Construct the graph $G[c, a_0, \dots, a_n]$ by adding the edges $E[c, a_0, \dots, a_n]$ to the disjoint union of graphs G_m, $0 \leqq m \leqq n$. An edge $\langle\langle i, m_1\rangle, \langle j, m_2\rangle\rangle \in E[c, a_0, \dots, a_n]$ if and only if one of the following conditions is satisfied:

(i) $m_1 = m_2 = 0$, $i \in V^0_{1,1}$, $j > k$ and $x_{k\cdot n+(j-k)}$ occurs in t_i;

(ii) $m_1 = 0$, $m_2 \geqq 1$, $i \in V^0_{1,1}$, $j \leqq k$ and $x_{k\cdot(m_2-1)+j}$ occurs in t_i;

(iii) $m_1 \geqq 1$, $m_2 = 0$, $i \in V^{m_1}_{2,1}$, $j > k$ and $x_{k\cdot n+(j-k)}$ occurs in $t_{k+l\cdot(m_1-1)+(i-k)}$;

(iv) $m_1 \geqq 1$, $m_2 \geqq 1$, $i \in V^{m_1}_{2,1}$, $j \leqq k$ and $x_{k\cdot(m_2-1)+j}$ occurs in $t_{k+l\cdot(m_1-1)+(i-k)}$.

$G'[c, a_0, \dots, a_n]$ can be obtained from $G[c, a_0, \dots, a_n]$ by leaving the edges $E^0_1 \cup \left(\bigcup_{m=1}^{n} E^m_2\right)$. We say that c realizes the transition $a_0 \sigma \underset{\mathfrak{B}}{\vdash} \sigma(a_1, \dots, a_n)$ if the following conditions are satisfied. (The mark $[c, a_0, \dots, a_n]$ will be omitted from the right of G, G' and E.)

(A) Let $i \in I_\sigma$. If $i \leqq k$, then $i \in V^0_{1,2}$, else if for some $m \in [n]$ and $k < j \leqq k+l$, $i = l\cdot(m-1)+j$, then $j \in V^m_{2,2}$.

(B) For each $m \in [n]$, $i \in V^m_{1,1}$ if and only if there exists an $i' \in V^0_{1,1}$ such that $\langle i', 0\rangle \overset{+}{\underset{G}{\vdash}} \langle i, m\rangle$.

(C) For each $0 \leqq m \leqq n$, $\overset{+}{\underset{G'}{\vdash}} | G_m = \overset{+}{\underset{G_m}{\vdash}}$.

Now for each $\sigma \in \Sigma_n$, $a_0 \sigma \underset{\mathfrak{B}}{\vdash} \sigma(a_1, \dots, a_n)$ if and only if this transition is realized by some $c \in A[\sigma]$.

Let $q \geqq 0$, $t \in \tilde{T}(\Sigma)^1_q$. A deterministic part of $E_{S(t)}$ can be chosen as follows. Replace the equations of the form $z = \emptyset$ by $z = z$, then for each $z \in Z_{S(t)} \setminus Z^1_{S(t)}$ replace the right-hand side of the equation $z = T_z$ by an arbitrary $t_z \in T_z$. Further on $DE_{S(t)}$ will always denote a deterministic part of $E_{S(t)}$. For each $z \in Z_{S(t)} \setminus Z^1_{S(t)}$, $\pi(z) \cdot E^+_{S(t)} \neq \emptyset$ if and only if there exists a $DE_{S(t)}$ such that $\pi(z) \cdot DE^+_{S(t)} \neq \emptyset$. ($\pi(z)$. means the selection of the component z.) Let $\vdash DE_{S(t)}$ denote the dependence relation among the variables $Z_{S(t)}$ in a deterministic part of $E_{S(t)}$, that is, $z_1 \vdash DE_{S(t)} z_2$ if and only if z_2 occurs in t_{z_1}. It is clear that $\pi(z) \cdot DE_{S(t)} \neq \emptyset$ if and only if $z \overset{*}{\vdash} DE_{S(t)} z'$ implies $z' \overset{+}{\nvdash} DE_{S(t)} z'$.

For each $n \in [l]$ take a new symbol γ_n, and construct the ranked alphabet $\Gamma = \bigcup_{n=1}^{l} \Gamma_n$ with $\Gamma_n = \{\gamma_n\}$. Let $q \geqq 0$, $t \in \tilde{T}(\Sigma)_q^1$, $a_1, \ldots, a_q \in A$, $a_j = (G_j; V_{1,1}^j, \ldots, V_{2,2}^j)$ for each $j \in [q]$. By $E_t[a_1, \ldots, a_q]$ we mean the following system of equations

$$E_t[a_1, \ldots, a_q] = \{x(\langle w, x_j \rangle, i) = \gamma_n(y(\langle w, x_j \rangle, m_1), \ldots, y(\langle w, x_j \rangle, m_n)) | \\ j \in [q], \langle w, x_j \rangle \in \text{lvs}(t), i \in [k] \text{ and } m_1, \ldots, m_n \text{ are all the possible values of such an } m \text{ for which } \langle i, k+m \rangle \in E_1^j\}.$$

Lemma 3.8. Let $q \geqq 0$, $t \in \tilde{T}(\Sigma)_q^1$, $a_1, \ldots, a_q \in A$ and for each $j \in [q]$, $a_j = (G_j; V_{1,1}^j, \ldots, V_{2,2}^j)$. There exists an $a \in A_0$ for which $at \underset{\mathfrak{B}}{\overset{*}{\vdash}} t(a_1, \ldots, a_q)$ if and only if a $DE_{S(t)}$ can be chosen such that

(i) $\pi(x(\langle \lambda, S \rangle, 1)) \cdot (DE_{S(t)} \cup E_t[a_1, \ldots, a_q])^+ \neq \emptyset$;

(ii) for each $j \in [q]$, $\langle w, x_j \rangle \in \text{lvs}(S(t))$, $i \in [k]$, $x(\langle \lambda, S \rangle, 1) \overset{+}{\vdash} x(\langle w, x_j \rangle, i)$ holds in $DE_{S(t)} \cup E_t[a_1, \ldots, a_q]$ if and only if $i \in V_{1,1}^j$;

(iii) for each $m+k \in V_{2,1}^j$, $y(\langle w, x_j \rangle, m) \overset{+}{\vdash} x(\langle w, x_j \rangle, i)$ if and only if $m+k \underset{G_j}{\overset{+}{\vdash}} i$.

Proof. Only if: If $t = x_1$, then $a = a_1 \in A_0$. In this case $E_{S(t)}$ is the same as $h(S)$, written in the form of equations, so (i), (ii) and (iii) follow from the conditions (a), (b) and (c) that must hold for $a \in A_0$. Let $q \geqq 1$, $p \in [q]$, $n \geqq 0$, $\sigma \in \Sigma_n$, $t_0 \in \tilde{T}(\Sigma)_q^1$ and $t = t_0 \cdot (\text{id}_{p-1} \otimes \sigma(x_1, \ldots, x_n) \otimes \text{id}_{q-p})$. If $at \underset{\mathfrak{B}}{\overset{+}{\vdash}} t(a^1, \ldots, a^{p-1}, a_1, \ldots, a_n, a^{p+1}, \ldots, a^q)$, then there exists an $a_0 \in A$ such that $at_0 \underset{\mathfrak{B}}{\overset{*}{\vdash}} t_0(a^1, \ldots, a^{p-1}, a_0, a^{p+1}, \ldots, a^q)$ and $a_0 \sigma \underset{\mathfrak{B}}{\vdash} \sigma(a_1, \ldots, a_n)$. Suppose the *Only if* part is true for t_0 and states $a^1, \ldots, a^{p-1}$, $a_0, a^{p+1}, \ldots, a^q$, and the transition $a_0 \sigma \underset{\mathfrak{B}}{\vdash} \sigma(a_1, \ldots, a_n)$ is realized by $c = \langle t_1, \ldots, t_{k+l \cdot n} \rangle \in A[\sigma]$. Then there exists an appropriate $DE_{S(t_0)}$ satisfying the three conditions. For all $i \in [k]$ and $m \in [l]$, replace the variables $x(\langle w, x_p \rangle, i)$ and $y(\langle w, x_p \rangle, m)$ in $DE_{S(t_0)}$ by $x(\langle w, \sigma \rangle, i)$ and $y(\langle w, \sigma \rangle, m)$, respectively, and add the set of equations

$$\{x(\langle w, \sigma \rangle, i) = t_i[x_{k \cdot (j-1)+r} \leftarrow x(\langle wj, x_{j+p-1} \rangle, r), \\ x_{k \cdot n+s} \leftarrow y(\langle w, \sigma \rangle, s), \perp \leftarrow x(\langle w, \sigma \rangle, i) | j \in [n], r \in [k], s \in [l]] | i \in [k]\} \cup \\ \cup \{y(\langle wj, x_{j+p-1} \rangle, m) = t_{k+l \cdot (j-1)+m}[x_{k \cdot (u-1)+r} \leftarrow x(\langle wu, x_{u+p-1} \rangle, r), \\ x_{k \cdot n+s} \leftarrow y(\langle w, \sigma \rangle, s), \perp \leftarrow y(\langle wj, x_{j+p-1} \rangle, m) | u \in [n], r \in [k], s \in [l]] | j \in [n], m \in [l]\}$$

to obtain $DE_{S(t)}$. For $0 \leqq m \leqq n$ let $\alpha_m \in \text{nds}(S(t))$ such that $\alpha_m = \langle w, \sigma \rangle$ if $m = 0$, else $\alpha_m = \langle wm, x_{m+p-1} \rangle$. If $i \in [k+l]$, then $z(\langle i, m \rangle)$ will denote the following variable of $Z_{S(t)}$

$$z(\langle i, m \rangle) = \begin{cases} x(\alpha_m, i) & \text{if } i \in [k], \\ y(\alpha_m, i-k) & \text{if } i > k. \end{cases}$$

By the inductive hypothesis and conditions (A), (B), (C) imposed on the transitions of $\mathfrak{B}$ we have

(*) $\langle i_1, m_1\rangle \overset{+}{\underset{G}{\vdash}} \langle i_2, m_2\rangle$ if and only if for $j=1,2, i_j \in (V_{1,1}^{m_j} \cup V_{2,1}^{m_j})$ and $z(\langle i_1, m_1\rangle) \overset{+}{\vdash} DE_{S(t)} \cup E_t[a^1, \ldots, a^{p-1}, a_1, \ldots, a_n, a^{p+1}, \ldots, a^q]\, z(\langle i_2, m_2\rangle)$ are both satisfied ($G=G[c, a_0, \ldots, a_n]$).

To prove (i) suppose that $x(\langle\lambda, S\rangle, 1) \overset{*}{\vdash} z$ and $z \overset{+}{\vdash} z$ hold in $DE_{S(t)} \cup \cup E_t[a^1, \ldots, a^{p-1}, a_1, \ldots, a_n, a^{p+1}, \ldots, a^q]$ for some $z \in Z_{S(t)}$. By the inductive hypothesis we can assume that $z=z(\langle i, m\rangle)$ for some $i \in [k+l]$, $0 \leqq m \leqq n$. Using (*) and (C) we conclude that G_m contains a cycle, which is a contradiction.

Let $\alpha = \langle u, x_j\rangle \in \mathrm{lvs}\,(S(t))$. By (B) and (*), $i \in V_{1,1}^j$ if and only if there exists a $j' \in [q]$ and an $i' \in V_{1,1}^{j'}$ such that

$$x(\alpha_{j'}, i') \overset{*}{\vdash} DE_{S(t)} \cup E_t[a^1, \ldots, a^{p-1}, a_1, \ldots, a_n, a^{p+1}, \ldots, a^q]x(\alpha, i),$$

where $\alpha_{j'} = \langle w, \sigma\rangle$ if $j'=p$, else $\alpha_{j'}=\alpha$. Let $\bar{\alpha}_{j'} = \langle w, x_p\rangle$ if $j'=p$, else $\bar{\alpha}_{j'}=\alpha$. By the inductive hypothesis $i' \in V_{1,1}^{j'}$ if and only if $x(\langle\lambda, S\rangle, 1) \overset{+}{\vdash} x(\bar{\alpha}_{j'}, i')$ holds in $DE_{S(t_0)} \cup E_{t_0}[a^1, \ldots, a^{p-1}, a_0, a^{p+1}, \ldots, a^q]$, which is equivalent to

$$x(\langle\lambda, S\rangle) \overset{+}{\vdash} DE_{S(t)} \cup E_t[a^1, \ldots, a^{p-1}, a_1, \ldots, a_n, a^{p+1}, \ldots, a^q]\, x(\alpha_{j'}, i').$$

Thus, $i \in V_{1,1}^j$ if and only if $x(\langle\lambda, S\rangle, 1) \overset{+}{\vdash} x(\alpha, i)$, which proves (ii).

Let us remark that (iii) is already proved for $p \leqq j < p+n$ as a special case of (*). It is easy to prove it for other values of j, too.

If: The case $t=x_1$ is again trivial. Let $t=t_0 \cdot (\mathrm{id}_{p-1} \otimes \sigma(x_1, \ldots, x_n) \otimes \mathrm{id}_{q-p})$ as above, and suppose the *If* part is true for t_0 and any appropriate states $b_1, \ldots, b_q$. Let $DE_{S(t)}$ and the states $a^1, \ldots, a^{p-1}, a_1, \ldots, a_n, a^{p+1}, \ldots, a^q$ satisfy (i), (ii) and (iii). Split $DE_{S(t)}$ into $DE_{S(t_0)}$ and a part that can be derived from $c=\langle t_1, \ldots, t_{k+l\cdot n}\rangle \in \in A[\sigma]$. Let $a_0=(G_0; V_{1,1}^0, \ldots, V_{2,2}^0)$ be the following state

$i \in V_{1,1}^0$ if and only if $x(\langle\lambda, S\rangle, 1) \overset{+}{\vdash} x(\langle w, \sigma\rangle, i)$ holds in $DE_{S(t)} \cup \cup E_t[a^1, \ldots, a^{p-1}, a_1, \ldots, a_n, a^{p+1}, \ldots, a^q]$, where w is the first component of the node $\langle w, x_p\rangle$ in t_0;

$\langle i, j\rangle \in E_1^0$ if and only if $i \in V_{1,1}^0$ and $x(\langle w, \sigma\rangle, i) \overset{+}{\vdash} y(\langle w, \sigma\rangle, j-k)$, $V_{2,1}^0 = \{j \mid$ for some $i \in V_{1,1}^0 \langle i, j\rangle \in E_1^0\}$;

$\langle j, i\rangle \in E_2^0$ if and only if $j \in V_{2,1}^0$ and $y(\langle w, \sigma\rangle, j-k) \overset{+}{\vdash} x(\langle w, \sigma\rangle, i)$.

It is clear that $DE_{S(t_0)}$ and states $a^1, \ldots, a^{p-1}, a_0, a^{p+1}, \ldots, a^q$ satisfy (i), (ii) and (iii), hence, by the inductive hypothesis $at_0 \overset{*}{\underset{\mathfrak{B}}{\vdash}} t_0(a^1, \ldots, a^{p-1}, a_0, a^{p+1}, \ldots, a^q)$ for some $a \in A_0$. On the other hand it can easily be checked that $a_0\sigma \underset{\mathfrak{B}}{\vdash} \sigma(a_1, \ldots, a_n)$ is realized by c, so we are through.

Taking $q=0$ in the lemma we get

Theorem 3.9. The domain of attributed tree transformations is a regular tree language.

However, Lemma 3.8 is worth some further considerations. It can be seen that Lemma 3.8 remains valid if we require the states of $\mathfrak{B}$ not contain any redundant edges. (An adge $\langle i, j\rangle$ is redundant if there is another path from i to j containing more than one edge.) Let $\mathfrak{A}$ be deterministic, and suppose the states of $\mathfrak{B}$ satisfy the above additional requirement. The following statement can be proved by a bottom-up type induction combined with Lemma 3.8.

Proposition 3.10. Let $t\in D\tau_{\mathfrak{A}}$, $t=t_0\cdot u$ with $t_0\in\tilde{T}(\Sigma)_1^1$. There exists a unique $a\in A$ such that for some $a_0\in A_0$ we have $a_0t_0\overset{*}{\underset{\mathfrak{B}}{\vdash}} t_0(a)$ and $au\overset{+}{\underset{\mathfrak{B}}{\vdash}} u$. This unique $a=(G; V_{1,1}, \ldots, V_{2,2})$ is the following: $V_{1,1}\cup V_{2,1}=Z_\alpha=\{z\in Z_{S(t)}|$ the "node" index of z is $\alpha=\text{root}(u)$ and $x(\langle\lambda, S\rangle, 1)\overset{+}{\vdash} DE_{S(t)}z\}$, and $\overset{+}{\underset{G}{\vdash}}=\overset{+}{\vdash} DE_{S(t)}|Z_\alpha$. (Obviously, $DE_{S(t)}$ is unique in this case.)

As an application of Proposition 3.10 we finally show how to decide the K-visit property for deterministic attributed tree transducers. (Alternative proofs can be derived from [6] and [7].) Let $t\in D\tau_{\mathfrak{A}}$, $\alpha\in\text{nds}(t)$. Proposition 3.10 shows that the state $a=(G_\alpha; V_{1,1}^\alpha, \ldots, V_{2,2}^\alpha)$ in which $\mathfrak{B}$ passes through α during the recognition of t is uniquely determined, and it describes the dependence relation among the useful attributes of α. If p is a path in G_α ($p\in\text{path}(G_\alpha)$), then let $v_p=\|\{i\in V_{1,1}^\alpha | p$ passes through $i\}\|$, $v_\alpha=\max\{v_p | p\in\text{path}(G_\alpha)\}$. v_α shows how many times we must "enter" the subtree having root α to ask for the value of certain attributes. (Supposing an optimal, maximally paralleled evaluation of the useful attributes.) Define

$$v_{\mathfrak{A}} = \max\{v_\alpha | \alpha\in\text{nds}(t) \quad \text{for some} \quad t\in D\tau_{\mathfrak{A}}\}.$$

Since this set is finite, it is easy to give an algorithm that computes $v_{\mathfrak{A}}$, and obviously, $\mathfrak{A}$ is K-visit if and only if $v_{\mathfrak{A}}\leqq K$. Moreover, it follows from the construction that

$$\text{if } l < k, \quad \text{then} \quad v_{\mathfrak{A}} \leqq l+1, \quad \text{else} \quad v_{\mathfrak{A}} \leqq k.$$

A trivial consequence of this statement is the known fact that every deterministic attributed tree transducer is K-visit for some K.

Abstract

A general concept of attributed transformation is introduced by means of magmoids and rational theories. It is shown that the domain of attributed tree transformations is a regular tree language, and an alternative proof is given for the decidability of the K-visit property of deterministic attributed tree-transducers.

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

References

[1] ARNOLD, A. & M. DAUCHET, Theorie des magmoides, Preliminary work to the authors' theses, Univ. de Lille, France, 1977.

[2] DAUCHET, M., Transductions de forets, bimorphismes de magmoides, These, Univ. de Lille, France, 1977.

[3] WRIGHT, J. B., J. W. THATCHER, E. G. WAGNER & J. A. GOGUEN, Rational algebraic theories and fixed-point solutions, 17-th IEEE Symposium on Foundations of Computing, Houston, 1976, pp. 147—158.

[4] CHIRICA, L. M. & D. F. MARTIN, An order-algebraic definition of Knuthian semantics, *Math. Systems Theory*, v. 13, 1979, pp. 1—27.

[5] KNUTH, D. E., Semantics of context-free languages, *Math. Systems Theory*, v. 2, 1968, pp. 127—145; Correction, v. 5, 1971, pp. 95—96.

[6] RIIS, H. & S. SKYUM, *k*-visit attribute grammars, DAIMI PB-121, Aarhus University, Denmark, 1980.

[7] FÜLÖP, Z., Attribute grammars and attributed tree transducers, 15-th National Scientific Conference for Students, 1981, Budapest, Hungary.

(Received April 29, 1981)

Simple deterministic machines

By N. T. KHANH

§ 1. Introduction

Classes of formal languages are frequently characterized by different types of accepting automata. It is interesting to note that deterministic languages, i.e., languages accepted by deterministic automata are very difficult to characterize by other properties. However, as it will be shown in this paper, if we restrict ourselves to the so called simple deterministic machines then the corresponding language classes can be characterized by the prefix-free property. A language L is called prefix-free if and only if for every pair of words (x, y): $x \in L$ and $xy \in L$ jointly imply $y = \lambda$, where λ is the empty word. The hierarchy of simple deterministic machines will thus correspond to the intersections of the classes of deterministic languages in the Chomsky hierarchy with the family of all prefix-free languages. Simple machines introduced by E. P. FRIEDMAN in [3] will be compared with our simple deterministic pushdown machines and we show that the class of languages accepted by the former ones constitute a proper subset of those accepted by the latter machines. This means that although the languages accepted by Friedman's simple machines are prefix-free, they do not include every prefix-free deterministic language. The classes of languages characterized by our simple deterministic machines have different closure properties under the usual operations. We also define some specific operations with respect to the prefix-free property. The usefulness of our simple deterministic machines can be seen also from the properties of the corresponding language classes.

§ 2. Prefix-free languages

Definition 2.1. A language L is said to be prefix-free if for every pair of words (x, y): $x \in L$ and $xy \in L$ imply $y = \lambda$. The family of all prefix-free languages is denoted by $\mathcal{L}_p$. We can prove that $\mathcal{L}_p$ is closed under intersection and concatenation, but it is not closed under complementation and union.

Definition 2.2. Let L_1, L_2 be two languages over the alphabet Σ.
a) For x, y in Σ^*, write $x < y$ if $y = xz$ for some z in $\Sigma^* - \{\lambda\}$.
b) The p-quotient of L_1 by L_2 is defined by

$$L_1 p L_2 = \{y \in L_1 / \text{ if } x < y \text{ then } x \notin L_2\}.$$

c) The p-union of two languages L_1 and L_2 is defined by

$$L_1 \cup_p L_2 = (L_1 p L_2) \cup (L_2 p L_1).$$

It is easy to see that

$$L_1 \cup_p L_2 = L_1 \cup L_2 - \{y_1 \in L_1 / \text{ there is an } x < y_1 \text{ with } x \in L_2\}$$
$$- \{y_2 \in L_2 / \text{ there is an } x < y_2 \text{ with } x \in L_1\}.$$

Theorem 2.1. The family $\mathscr{L}_p$ is closed under p-quotient and p-union.

Proof. Let L_1, L_2 be prefix-free languages. It should be clear that $L_1 p L_2$ is prefix-free. We now prove that $L_1 \cup_p L_2$ is prefix-free. Assume on the contrary that there is an $x \in L_1 \cup_p L_2$ with $xy \in L_1 \cup_p L_2$ for some $y \neq \lambda$. Let $x \in L_1 p L_2$. (The case where $x \in L_2 p L_1$ is similar.) Since L_1 is prefix-free and $y \neq \lambda$, it suffices to consider the case where $xy \in L_2 p L_1$. But, by Definition 2.2, we can easily see that if $xy \in L_2 p L_1$ for $y \neq \lambda$ then $x \notin L_1$, i.e., $x \notin L_1 p L_2$ and the contradiction arises. □

Definition 2.3. Let Σ and Δ be two disjoint alphabets, and w any fixed string in Δ^*. We define the homomorphism $h_w \colon \Sigma^* \to (\Sigma \cup \Delta)^*$, such that

a) $h_w(\lambda) = \lambda$,

b) $h_w(a) = aw$ for all $a \in \Sigma$,

c) $h_w(PQ) = h_w(P)\, h_w(Q)$ for all $P, Q \in \Sigma^*$.

For a language L over Σ, we define

$$h_w(L) = \{h_w(P) / P \in L\}.$$

Theorem 2.2. A language $L \subseteq \Sigma^*$ is prefix-free if and only if $h_w(L) \subseteq (\Sigma \cup \Delta)^*$ is prefix-free.

Proof. The case where $w = \lambda$ is trivial, so we assume that $w = a_1 \ldots a_n$ for some $a_1, \ldots, a_n \in \Sigma$ with $n \geqq 1$. Similarly, we can assume that $L \neq \{\lambda\}$.

Part 1. $L \in \mathscr{L}_p \to h_w(L) \in \mathscr{L}_p$.

Let $x \in h_w(L)$ and $xy \in h_w(L)$, then there are $P \in L$ and $PQ \in L$ such that $x = h_w(P)$, $xy = h_w(PQ) = h_w(P) h_w(Q)$. Since L is prefix-free, $Q = \lambda$. Consequently, $y = h_w(Q) = h_w(\lambda) = \lambda$, Thus $h_w(L) \in \mathscr{L}_p$.

Part 2. $h_w(L) \in \mathscr{L}_p \to L \in \mathscr{L}_p$.

Assume on the contrary that there are $x \in L$ and $xy \in L$ for some $y \neq \lambda$. It is clear that $P = h_w(x) \in h_w(L)$, $PQ = h_w(xy) \in h_w(L)$ and $Q = h_w(y) \neq \lambda$. Thus, $h_w(L)$ is not prefix-free and the contradiction arises. □

§ 3. Simple finite deterministic machines

In this section first we investigate a special kind of finite deterministic automata called simple finite deterministic machines (abbreviated SFD-machines), and prove that the family of all languages accepted by SFD-machines is the intersection of the two families $\mathscr{L}_3$ and $\mathscr{L}_p$. Further, we note that this family is not closed under complementation and union, but it is closed under concatenation, intersection, p-quotient,

p-union and homomorphism h_w. The proofs of these facts will not be presented in this paper.

Let us consider the standard definition of a finite deterministic automaton (abbreviated FD-automaton, see [1]). That is: Let $M=(K, \Sigma, \delta, q_0, H)$ be an FD-automaton, where K is the set of states, Σ is the set of inputs, q_0 is an element of K (the initial state), H is a subset of K (the set of final states), and δ is a mapping from $K\times\Sigma$ to K.

Notation. Given an FD-automaton M let $\underset{M}{\vdash}$ be the relation on $K\times\Sigma^*$ defined as follows. For $a\in\Sigma$, $w\in\Sigma^*$, $q, p\in K$

$$qaw \vdash pw \text{ iff } \delta(q, a)=p.$$

We let $\overset{*}{\underset{M}{\vdash}}$ denote the transitive closure of $\underset{M}{\vdash}$. Finally, we define the language accepted by M to be

$$L(M) = \{w\in\Sigma^*/q_0 w \overset{*}{\underset{M}{\vdash}} p \quad \text{for some} \quad p\in H\}.$$

Definition 3.1. a) A simple finite deterministic machine (abbreviated SFD-machine is a 5-tuple $M=(K, \Sigma, \delta, q_0, H)$, where K, Σ, q_0, H are the same as in the definition of an FD-automaton and δ is a mapping from $(K-H)\times\Sigma$ to K. Similarly, we define the language accepted by an SFD-machine M to be

$$L(M) = \{w\in\Sigma^*/q_0 w \overset{*}{\underset{M}{\vdash}} p \quad \text{for some} \quad p\in H\}.$$

b) A language L is said to be a simple finite deterministic language (abbreviated sfd-language), if $L=L(M)$ for some SFD-machine. The family of all sfd-languages is denoted by $\mathscr{L}_{sd3}$.

Remark. For simplicity of Definition 3.1 we have restricted the definition of the mapping δ to $K-H$, so δ is not complete. By the following theorem we shall see that the SFD-machine is a special kind of the FD-automaton.

Theorem 3.1. Let L be any language over the alphabet Σ. L is an sfd-language if and only if L is prefix-free and $L\in\mathscr{L}_3$.

Proof. Part 1. $L\in\mathscr{L}_{sd3}\rightarrow L\in\mathscr{L}_p\cap\mathscr{L}_3$.

Let $L=L(M)$ for an SFD-machine $M=(K, \Sigma, \delta, q_0, H)$. Since the domain of δ is $K-H$, we can easily see that $L\in\mathscr{L}_p$. We now prove that $L\in\mathscr{L}_3$.

Construct an FD-automaton M' from M as follows: Let $M'=(K\cup\{\bar{q}\}, \Sigma, \delta', q_0, H)$, where $\bar{q}\notin K$ and δ' is defined so that

1) for every $q\in K-H, a\in\Sigma$: $\delta'(q; a)=\delta(q, a)$,

2) for all $p\in H\cup\{\bar{q}\}, a\in\Sigma$: $\delta'(p, a)=\bar{q}$.

It is clear that $L(M')=L(M)$. Thus, $L\in\mathscr{L}_p\cap\mathscr{L}_3$.

Part 2. $L\in\mathscr{L}_3\cap\mathscr{L}_p\rightarrow L\in\mathscr{L}_{sd3}$.

By Theorem 3.3 in [1], we may assume that $L=L(M)$, where $M=(K, \Sigma, \delta, q_0, H)$ is an FD-automaton. We construct an SFD-machine M' from M as follows, Let $M'=(K, \Sigma, \delta', q_0, H)$, where δ' is defined so that

1) for every $q \in K-H$, $a \in \Sigma$: $\delta'(q, a)=\delta(q, a)$,

2) for all $p \in H$, $a \in \Sigma$: $\delta'(p, a)$ is undefined.

We now prove that $L(M')=L(M)$.

$\langle \subseteqq \rangle$. By the definition of δ', we can easily see that if $w \in L(M')$ then $w \in L(M)$.

$\langle \supseteqq \rangle$. We shall prove that if $w \notin L(M')$ then $w \notin L(M)$. We now have two cases to consider:

Case 1. Let $w=w_1 a w_2$ for $a \in \Sigma$, $w_1, w_2 \in \Sigma^*$, and $q_0 w_1 a w_2 \overset{*}{\underset{M'}{\vdash}} p a w_2$ for some $p \in H$.

It is easy to see that $q_0 w_1 \overset{*}{\underset{M}{\vdash}} p$ for $p \in H$. Thus, $w_1 \in L(M)$. Since $L(M)$ is prefix-free and $y=aw_2 \neq \lambda$, $w=w_1 y \notin L(M)$.

Case 2. Let $q_0 w \overset{*}{\underset{M'}{\vdash}} q$ for some $q \notin H$.

It is clear that $q_0 w \overset{*}{\underset{M}{\vdash}} q$ for $q \notin H$. Thus, $w \notin L(M)$. □

§ 4. Simple deterministic pushdown machines

In this section we investigate a special kind of deterministic pushdown automata known as simple deterministic pushdown machines (abbreviated SDP-machines) and prove that the family of all languages accepted by SDP-machines is the intersection of $\mathscr{L}_p$ and the family of all deterministic context-free languages. Furthermore, we can prove that this family is not closed under intersection, complementation and union, but it is closed under concatenation, homomorphism h_w, and $L_1 p L_2$, $L_1 \cup_p L_2$ are accepted by SDP-machines if L_1 is accepted by an SDP-machine and L_2 is an sfd-language. In this paper, however, we do not present all these proofs.

Let us consider the standard definition of a deterministic pushdown automaton (abbreviated DP-automaton, see [2]). That is: Let $M=(K, \Sigma, \Gamma, \delta, q_0, z_0, H)$ be a DP-automaton, where K is the set of states, Σ is the input alphabet, Γ is the pushdown alphabet, $q_0 \in K$ is the initial state, $z_0 \in \Gamma$ is the initial pushdown symbol, $H \subseteqq K$ is the set of final states, and δ is a mapping from $K \times (\Sigma \cup \{\lambda\}) \times \Gamma$ to $K \times \Gamma^*$ satisfying the following conditions: for each $q \in K, z \in \Gamma$ either (i) $\delta(q, \lambda, z)$ is undefined and $\delta(q, a, z)$ contains exactly one element for all $a \in \Sigma$, or (ii) $\delta(q, \lambda, z)$ contains exactly one element and $\delta(q, a, z)$ is undefined for all $a \in \Sigma$.

Notation. Given a DP-automaton M let $\underset{M}{\vdash}$ be the relation on $K \times \Sigma^* \times \Gamma^*$ defined as follows

$$\text{for} \quad q, p \in K, a \in \Sigma \cup \{\lambda\}, w \in \Sigma^*, z \in \Gamma, \alpha, \beta \in \Gamma^*,$$
$$(q, aw, \alpha z) \underset{M}{\vdash} (p, w, \alpha\beta) \quad \text{iff} \quad \delta(q, a, z)=(p, \beta).$$

Let $\overset{*}{\underset{M}{\vdash}}$ denote the transitive closure of $\underset{M}{\vdash}$. Finally, we define the language accepted by M to be

$$L(M)=\{w\in\Sigma^*/(q_0, w, z_0)\overset{*}{\underset{M}{\vdash}}(p, \lambda, \alpha) \text{ for some } p\in H, \alpha\in\Gamma^*\}.$$

A language L is said to be deterministic context-free if $L=L(M)$ for some DP-automaton M. The family of all deterministic context-free languages is denoted by $\mathcal{L}_{d2}$.

Definition 4.1. a) A simple deterministic pushdown machine (abbreviated SDP-machine) is a 7-tuple $M=(K, \Sigma, \Gamma, \delta, q_0, z_0, H)$, where $K, \Sigma, \Gamma, q_0, z_0$ and H are the same as in the definition of a DP-automaton, and δ is a mapping from $(K-H)\times(\Sigma\cup\{\lambda\})\times\Gamma$ to $K\times\Gamma^*$ satisfying the following conditions: for each $q\in K-H, z\in\Gamma$ either (i) $\delta(q, \lambda, z)$ is undefined and $\delta(q, a, z)$ contains exactly one element for all $a\in\Sigma$ or (ii) $\delta(q, \lambda, z)$ contains exactly one element and $\delta(q, a, z)$ is undefined for all $a\in\Sigma$.

b) An input string is accepted by the SDP-machine M when the entire tape has been processed and the actual state is a final state. That is

$$L(M)=\{w\in\Sigma^*/(q_0, w, z_0)\overset{*}{\underset{M}{\vdash}}(p, \lambda, \alpha) \text{ for some } p\in H\}.$$

A language L is said to be simple deterministic context-free (abbreviated sdc-language) if $L=L(M)$ for some SDP-machine M. Finally, the family of all sdc-languages is defined by $\mathcal{L}_{sd2}$.

Theorem 4.1. Let L be any language over the alphabet Σ. L is an sdc-language if and only if L is deterministic context-free and prefix-free.

Proof. Part 1. $L\in\mathcal{L}_{sd2}\to L\in\mathcal{L}_{d2}\cap\mathcal{L}_p$.

Let $L=L(M)$ for an SDP-machine $M=(K, \Sigma, \Gamma, \delta, q_0, z_0, H)$. By the definition of δ, we can easily see that L is prefix-free. We now prove that $L\in\mathcal{L}_{d2}$. Construct a deterministic pushdown automaton M' from M as follows.

Let $M=(K\cup\{\bar{q}\}, \Sigma, \Gamma, \delta', q_0, z_0, H)$, where $\bar{q}\notin K$ and δ' is defined as

1) for every $q\in K-H$, $a\in\Sigma\cup\{\lambda\}$, $z\in\Gamma$, $\delta'(q, a, z)=\delta(q, a, z)$,

2) for all $p\in H\cup\{\bar{q}\}$, $z\in\Gamma$, $\delta'(p, \lambda, z)=(\bar{q}, \lambda)$.

It is clear that $L(M')=L(M)$. Thus, $L\in\mathcal{L}_{d2}\cap\mathcal{L}_p$.

Part 2. $L\in\mathcal{L}_{d2}\cap\mathcal{L}_p\to L\in\mathcal{L}_{sd2}$.

Let $L=L(M)$ for a DP-automaton $M=(K, \Sigma, \Gamma, \delta, q_0, z_0, H)$. Construct an SDP-machine M' from M as follows.

Let $M'=(K, \Sigma, \Gamma, \delta', q_0, z_0, H)$, where δ' is defined so that

1) for every $q\in K-H$, $z\in\Gamma$, $a\in\Sigma\cup\{\lambda\}$, $\delta'(q, a, z)=\delta(q, a, z)$,

2) for all $p\in H, z\in\Gamma, a\in\Sigma\cup\{\lambda\}$: $\delta'(p, a, z)$ is undefined.

We now prove that $L(M')=L(M)$.

($\subseteq$). By the construction of δ', we can easily see that if $w\in L(M')$ then $w\in L(M)$.

($\supseteq$) We shall prove that if $w\notin L(M')$ then $w\notin L(M)$. We now have three cases to consider:

Case 1. Let $w=w_1aw_2$ for $a\in\Sigma$, $w_1, w_2\in\Sigma^*$, and $(q_0, w_1aw_2, z_0)\overset{*}{\underset{M'}{\vdash}}(p, aw_2, \alpha)$ for some $p\in H$.

It is clear that $(q_0, w_1, z_0)\overset{*}{\underset{M}{\vdash}}(p, \lambda, \alpha)$ for $p\in H$. Consequently, $w_1\in L(M)$. Since $L(M)$ is prefix-free and $y=aw_2\neq\lambda$, $w\notin L(M)$.

Case 2. Let $w=w_1w_2$, where $w_1, w_2\in\Sigma^*$, and $(q_0, w_1w_2, z_0)\overset{*}{\underset{M'}{\vdash}}(q, w_2, \lambda)$ for some $q\in K-H$.

It is clear that $(q_0, w_1w_2, z_0)\underset{M}{\vdash}(q, w_2, \lambda)$ for $q\in K-H$. Thus, $w\notin L(M)$.

Case 3. Let $(q_0, w, z_0)\overset{*}{\underset{M'}{\vdash}}(q, \lambda, \alpha z)$ for some $q\in K-H$, $z\in\Gamma$ such that $\delta'(q, \lambda, z)$ is undefined.

It is easy to see that $w\notin L(M)$. □

In the following part we want to deal with a subfamily of simple deterministic context-free languages known as simple context-free languages (E. P. Friedman 1977).

Definition 4.2. (Definition 2.1 in [3]). a) A simple machine is a 4-tuple $M=(\Sigma, \Gamma, \delta, z_0)$, where Σ is a finite input alphabet, Γ is a finite pushdown alphabet, $z_0\in\Gamma$ is the initial pushdown symbol, δ is the partial transition function from $(\Sigma\cup\{\lambda\})\times\Gamma$ to Γ^* satisfying the following conditions: for each $z\in\Gamma$ either (i) $\delta(\lambda, z)$ is undefined and $\delta(a, z)$ contains exactly one element for all $a\in\Sigma$; or (ii) $\delta(\lambda, z)$ contains exactly one element and $\delta(a, z)$ is undefined for all $a\in\Sigma$.

Let $\underset{M}{\vdash}$ be the relation on $\Sigma^*\times\Gamma^*$ defined as follows: for each $a\in\Sigma\cup\{\lambda\}$, $w\in\Sigma^*$, $z\in\Gamma$, $\alpha, \beta\in\Gamma^*$, $(aw, \alpha z)\underset{M}{\vdash}(w, \alpha\beta)$ if $\delta(a, z)=\beta$.

Let $\overset{*}{\underset{M}{\vdash}}$ denote the transitive closure of $\underset{M}{\vdash}$. Finally, we define the language accepted by the simple machine M to be $L(M)=\{w\in\Sigma^*/(w, z_0)\overset{*}{\underset{M}{\vdash}}(\lambda, \lambda)\}$.

b) A language L is said to be simple context-free (abbreviated sc-language) if $L=L(M)$ for some simple machine M. It is easy to see that if L is an sc-language then L must be prefix-free. The family of all sc-languages is denoted by $\mathscr{L}_{sc}$.

Theorem 4.2. a) For every SFD-machine M, there is a simple machine M' such that $L(M')=L(M)$.

b) There is a simple machine M_1 such that $L(M_1)\notin\mathscr{L}_{sd3}$.

Proof. a) Let $M=(K, \Sigma, \delta, q_0, H)$ be an SFD-machine. We now construct the simple machine M' from M as follows.

Let $M'=(\Sigma, \Gamma', \delta', z_{q_0})$, where $\Gamma'=\{z_q/q\in K\}$ and δ' is defined so that

1) for each $q\in K-H$, $a\in\Sigma$, if $\delta(q, a)=p$ then $\delta'(a, z_q)=z_p$,

2) for all $p\in H$: $\delta'(\lambda, z_p)=\lambda$.

It is clear that $L(M')=L(M)$.

b) To prove the second statement, we reconsider the non-regular language (which is not an sfd-language), first seen in [2] $L=\{a^nb^n/n\geqq 1\}$.

We now provide a simple machine M_1 which accepts this language $M_1=(\{a, b\}, \{Z_0, A, E\}, \delta_1, Z_0)$, where δ_1 is defined so that

1) $\delta_1(a, Z_0)=A$,
2) $\delta_1(a, A)=AA$,
3) $\delta_1(b, A)=\lambda$,
4) $\delta_1(b, Z_0)=E$,
5) $\delta_1(\lambda, E)=E$. □

Theorem 4.3. a) For every simple machine M, there is an SDP-machine M' such that $L(M')=L(M)$.

b) There is an SDP-machine M_1 such that $L(M_1)$ is not sc-language.

Proof. a) Let $M=(\Sigma, \Gamma, \delta, Z_0)$ be a simple machine. Without loss of generality, we may assume again that for arbitrary $a\in \Sigma\cup\{\lambda\}$ and $Z\in\Gamma$ if $\delta(a, Z)=\alpha$ then $\alpha\in(\Gamma-\{Z_0\})^*$. In the opposite case we can introduce a new initial pushdown symbol $\bar{Z}_0$ and take the new machine $\bar{M}=(\Sigma, \Gamma\cup\{\bar{Z}_0\}, \bar{\delta}, \bar{Z}_0)$, where $\bar{\delta}(a, \bar{Z}_0)=\delta(a, Z_0)$ and $\bar{\delta}(a, Z)=\delta(a, Z)$ for each $Z\in\Gamma$, $a\in\Sigma\cup\{\lambda\}$.

Construct an SDP-machine M' from M as follows. Let $M'=(K, \Sigma, \Gamma', \delta', q_0, Z_0, \{q_h\})$, where $K=\{q_0, q_h\}$, $\bar{\Gamma}=\Gamma\cup\{\bar{Z}_0\}$ for $\bar{Z}_0\notin\Gamma$, and δ' is defined so that for each $a\in\Sigma\cup\{\lambda\}$, $Z\in\Gamma-\{Z_0\}$

1) if $\delta(a, Z_0)=\alpha$ then $\delta'(q_0, a, Z_0)=(q_0, \bar{Z}_0\alpha)$,
2) if $\delta(a, Z)=\alpha$ then $\delta'(q_0, a, Z)=(q_0, \alpha)$,
3) $\delta'(q_0, \lambda, \bar{Z}_0)=(q_h, \lambda)$.

First by induction on the length of $w\in\Sigma^*$ we can easily prove that $(w, Z_0)\overset{*}{\underset{M}{\vdash}}(\lambda, \alpha)$ iff $(q_0, w, Z_0)\overset{*}{\underset{M'}{\vdash}}(q_0, \lambda, \bar{Z}_0\alpha)$. Now, let $w\in\Sigma^*$, then

$$w\in L(M) \leftrightarrow (w, Z_0) \overset{*}{\underset{M'}{\vdash}} (\lambda, \lambda)$$

$$\leftrightarrow (q_0, w, Z_0) \overset{*}{\underset{M'}{\vdash}} (q_0, \lambda, \bar{Z}_0) \overset{*}{\underset{M'}{\vdash}} (q_h, \lambda, \lambda)$$

$$\leftrightarrow w\in L(M').$$

b) To prove the second statement, we reconsider the non sc-language first seen in [3] $L=\{a^iba^ib/i\geqq 1\}\cup\{a^ica^ic/i\geqq 1\}$. We can easily check that the following SDP-machine M_1 accepts this language.

Let $M_1=(K, \{a, b, c\}, \Gamma, \delta_1, q_0, Z_0, \{q_h\})$, where $K=\{q_0, q_1, q_2, \bar{q}, q_h\}$, $\Gamma=\{Z_0, A\}$ and δ_1 is defined so that

1) 1.a) $\delta_1(q_0, a, Z_0) = (q_0, Z_0AA)$,
1.b) $\delta_1(q_0, a, A) = (q_0, AA)$,
1.c) $\delta_1(q_0, b, A) = (q_1, \lambda)$,
1.d) $\delta_1(q_0, c, A) = (q_2, \lambda)$,
1.e) $\delta_1(q_0, b, Z_0) = \delta_1(q_0, c, Z_0) = (\bar{q}, \lambda)$;

2) 2.a) $\delta_1(q_1, a, A) = (q_1, \lambda)$,
2.b) $\delta_1(q_1, b, A) = (\bar{q}, \lambda)$,
2.c) $\delta_1(q_1, c, A) = (\bar{q}, \lambda,)$,
2.d) $\delta_1(q_1, b, Z_0) = (q_h, \lambda)$;

3) 3.a) $\delta_1(q_2, a, A) = (q_2, \lambda)$,
3.b) $\delta_1(q_2, c, A) = (\bar{q}, \lambda)$,
3.c) $\delta_1(q_2, b, A) = (\bar{q}, \lambda)$,
3.d) $\delta_1(q_2, c, Z_0) = (q_h, \lambda)$;

4) $\delta_1(\bar{q}, \lambda, Z_0) = \delta_1(\bar{q}, \lambda, A) = (\bar{q}, \lambda)$. □

Theorem 4.4. There exists a prefix-free context-free language which is not an sdc-language.

Proof. Let Σ be a finite nonempty alphabet. By Corollary 1 to Theorem 3.5 in [2], we can easily see that $L=\{ww^R/w\in\Sigma^*\}$ is a context-free language which is not deterministic context-free, where w^R is the mirror image of w. Let c be a symbol not is Σ, and set $L_1=L\cdot\{c\}=\{ww^Rc/w\in\Sigma^*\}$. It is easy to see that $L_1\in\mathcal{L}_2\cap\mathcal{L}_p$. We now prove that $L_1\notin\mathcal{L}_{sd2}$. Assume on the contrary that $L_1\in\mathcal{L}_{sd2}$. By Corollary to Theorem 3.4 in [2], if $L_1=L\cdot\{c\}$ is deterministic context-free then L is deterministic context-free, and the contradiction arises. Consequently, $L_1=\{ww^Rc/x\in\Sigma^*\}$ is a prefix-free context-free language which is not an sdc-language. □

§ 5. Simple deterministic linear bounded machines

In this section we investigate a special kind of deterministic linear bounded automata called simple deterministic linear bounded machines (abbreviated SDLB-machines), and prove that the family of all languages accepted by SDLB-machines is the intersection of the family $\mathcal{L}_p$ and the family of all deterministic context-sensitive languages. Furthermore, we mention without proof that this family is closed under concatenation, intersection, p-quotient, p-union and homomorphism h_w; but it is not closed under complementation and union.

Let us consider the standard definition of a deterministic linear bounded automaton (abbreviated DLB-automaton, see [7]). That is: Let $M=(\Gamma, K, \delta, q_0, H)$ be a DLB-automaton, where Γ is the tape alphabet, K is the set of states, $q_0\in K$ is the initial state, $H\subseteq K$ is a set of final states and $\delta: K\times\Gamma\to K\times(\Gamma\cup\{R, L\})$ is the mapping satisfying the condition: for arbitrary $q\in K$ and $x\in\Gamma$, $\delta(q, x)$ contains exactly one element.

Notation. An instantaneous configuration is a word of the form w_1qw_2, where $q\in K$, $w_1, w_2\in\Gamma^*$ and $w_1w_2\neq\lambda$. Given a DLB-automaton M let $\underset{M}{\vdash}$ be the relation on configurations of M defined as follows. For $q, p\in K, x, y\in\Gamma$, $w_1, w_2\in\Gamma^*$

$$w_1qxw_2 \underset{M}{\vdash} w_1pyw_2 \quad \text{iff} \quad \delta(q, x) = (p, y),$$

$$w_1qxw_2 \underset{M}{\vdash} w_1xpw_2 \quad \text{iff} \quad \delta(q, x) = (p, R),$$

$$w_1yqxw_2 \underset{M}{\vdash} w_1pyxw_2 \quad \text{iff} \quad \delta(q, x) = (p, L).$$

Let $\overset{*}{\underset{M}{\vdash}}$ denote the transitive closure of $\underset{M}{\vdash}$. Finally, we define the language accepted by a DLB-automaton M to be $L(M)=\{w\in\Sigma^*/q_0w\overset{*}{\underset{M}{\vdash}}\alpha p$ for some $p\in H\}$, where $\Sigma\subseteq\Gamma$. A language L is said to be deterministic context-sensitive (abbreviated dcs-language) if $L=L(M)$ for some DLB-automaton M. The family of all dcs-languages is denoted by $\mathscr{L}_{d1}$.

Lemma 5.1. Let L be a dcs-language over the alphabet Σ. Then there is a DLB-automaton $M'=(\Gamma', K', \delta', q_0', H')$ such that

i) $L=L(M')$,

ii) $\delta'\colon K'\times\Gamma'\to K'\times((\Gamma'-\Sigma)\cup\{R, L\})$ is the mapping satisfying the following condition: for arbitrary $q\in K'$ and $a\in\Sigma$, there is a $z\in\Gamma'-\Sigma$ such that $\delta'(q, a)=$ $=(p, z)$, i.e., there are no forms $\delta(q, a)=(p, R)$ or $\delta(q, a)=(p, L)$.

Proof. Let $L=L(M)$ for a DLB-automaton $M=(\Gamma, K, \delta, q_0, H)$. We now construct a DLB-automaton M' from M as follows. Let $M'=(\Gamma', K', \delta', q_0', H')$, where $\Gamma'=\Gamma\cup\{a'/a\in\Sigma\}$, $K'=K, q_0'=q_0$, $H'=H$ and δ' is defined so that

1) for arbitrary $a\in\Sigma$ and $q\in K, \delta'(q, a)=(q, a')$,

2) for arbitrary $x\in\Gamma-\Sigma$ and $q\in K$

2.a) if $\delta(q, x)=(p, i)$ for an $i\in\{R, L\}$ then $\delta'(q, x)=(p, i)$,

2.b) if $\delta(q, x)=(p, y)$ then $\delta'(q, x)=(p, \bar{y})$, where

$$\bar{y}=\begin{cases} y & \text{if } y\in\Gamma-\Sigma,\\ y' & \text{if } y\in\Sigma,\end{cases}$$

3) for arbitrary $q\in K$ and $a\in\Sigma$.

3.a) if $\delta(q, a)=(p, i)$ for an $i\in\{R, L\}$ then $\delta'(q, a')=(p, i)$,

3.b) if $\delta(q, a)=(p, y)$ then $\delta'(q, a')=(p, \bar{y})$, where

$$\bar{y}=\begin{cases} y & \text{if } y\in\Gamma-\Sigma,\\ y' & \text{if } y\in\Sigma.\end{cases}$$

It is clear that $L(M')=L(M)$ and the condition ii) is satisfied. □

Definition 5.1. a) A simple deterministic linear bounded machine (abbreviated SDLB-machine) is a 6-tuple $M=(\Gamma, K, \Sigma, \delta, q_0, H)$, where Γ is the tape alphabet, K is the set of states, Σ is the input alphabet for $\Sigma\cap\Gamma=\emptyset$, $q_0\in K$ is the initial state, $H\subseteq K$ is a set of final states, and $\delta\colon K\times(\Gamma\cup\Sigma)\to K\times(\Gamma\cup\{R, L\})$ is the mapping satisfying the following conditions

i) for arbitrary $q\in K-H$ and $x\in\Gamma\cup\Sigma$: $\delta(q, x)$ contains exactly one element,

ii) for every $p\in H$: $\delta(p, a)$ contains exactly one element if $a\in\Gamma$ and it is undefined if $a\in\Sigma$.

b) An instantaneous configuration and the relation $\overset{*}{\underset{M}{\vdash}}$ are defined as in the case of a DLB-automaton. We define the language accepted by a SDLB-machine M to be

$$L(M) = \{w\in\Sigma^*/q_0w\overset{*}{\underset{M}{\vdash}}\alpha p \quad\text{for some}\quad p\in H\}.$$

A language L is said to be simple deterministic context-sensitive (abbreviated sdcs-language) if $L=L(M)$ for some SDLB-machine M. The family of all sdcs-languages is denoted by $\mathscr{L}_{\text{sdl}}$.

Theorem 5.2. Let L be any language over the alphabet Σ. L is an sdcs-language if and only if L deterministic context-sensitive and prefix-free.

Proof. Part 1. $L\in\mathscr{L}_{\text{sdl}}\to L\in\mathscr{L}_{dl}\cap\mathscr{L}_p$.

Let $L=L(M)$ for an SDLB-machine $M=(\Gamma, K, \Sigma, \delta, q_0, H)$. By condition ii) of δ, we can easily see that if M is an SDLB-machine then $L(M)$ must be prefix-free. We now prove that $L\in\mathscr{L}_{dl}$. Construct a DLB-automaton M' from M as follows.

Let $M'=(\Gamma', K', \delta', q_0, H)$, where $\Gamma'=\Gamma\cup\Sigma$, $K'=K\cup\{\bar{q}\}$ for $\bar{q}\notin K$, and δ' is defined so that

1) for every $q\in K-H$: $\delta'(q,x)=\delta(q,x)$ for all $x\in\Gamma\cup\Sigma$,

2) for every $p\in H$

$$\delta'(p,a)=\begin{cases}\delta(p,a) & \text{if } a\in\Gamma,\\ (\bar{q},R) & \text{if } a\in\Sigma,\end{cases}$$

3) $\delta'(\bar{q},x)=(\bar{q},R)$ for all $x\in\Gamma\cup\Sigma$.

It is clear that $L(M')=L(M)$. Thus $L\in\mathscr{L}_{dl}\cap\mathscr{L}_p$.

Part 2. $L\in\mathscr{L}_{dl}\cap\mathscr{L}_p\to L\in\mathscr{L}_{\text{sdl}}$.

Without loss of generality, we may assume that $L=L(M)$ for a DLB-automaton $M=(\Gamma, K, \delta, q_0, H)$ satisfying condition ii) of Lemma 5.1. We now construct an SDLB-machine M' from M as follows. Let $M'=(\Gamma', K, \Sigma, \delta', q_0, H)$, where $\Gamma'=\Gamma-\Sigma$, δ' is defined so that

1) for every $q\in K-H$: $\delta'(q,x)=\delta(q,x)$ for all $x\in\Gamma'\cup\Sigma$,

2) for every $p\in H$

$$\delta'(p,a)=\begin{cases}\delta(p,a) & \text{if } a\in\Gamma',\\ \text{undefined} & \text{if } a\in\Sigma.\end{cases}$$

We now prove that $L(M')=L(M)$.

($\subseteq$). Let $w\in L(M')$, i.e., $q_0w\overset{*}{\underset{M'}{\vdash}}\alpha p$ for some $p\in H$. By the definition of δ', we obtain: $q_0w\overset{*}{\underset{M}{\vdash}}\alpha p$ for $p\in H$. Thus $w\in L(M)$.

($\supseteq$). We shall prove that if $w\notin L(M')$ then $w\notin L(M)$. There are two cases to consider:

Case a) Let $w=w_1aw_2$ for $a\in\Sigma$, $w_1, w_2\in\Sigma^*$ and $q_0w_1aw_2\overset{*}{\underset{M'}{\vdash}}\alpha paw_2$ for some $p\in H$.

It is clear that $q_0w_1\overset{*}{\underset{M}{\vdash}}\alpha p$ for $p\in H$. Since L is prefix-free and $y=aw_2\neq\lambda$, we obtain: $w=w_1y\notin L(M)$.

Case b) Let $q_0w\overset{*}{\underset{M'}{\vdash}}\alpha q$ for some $q\in K-H$.

It is clear that $q_0w\overset{*}{\underset{M}{\vdash}}\alpha q$ for $q\notin H$. Thus $w\notin L(M)$. □

Theorem 5.3. a) For every SDP-machine M, there is an SDLB-machine M' such that $L(M')=L(M)$.

b) There is an SDLB-machine M_1 such that $L(M_1)\notin\mathscr{L}_{\mathrm{sd}2}$.

Proof. a) Part a) holds, due to Theorem 5.2 and the following statement (Theorem 3 in [5]) "A context-free language is accepted by a deterministic linear bounded automaton".

b) To prove part b), we reconsider the non context-free language

$$L = \{a^n b^n c^n / n \geqq 1\}$$

(which is not an sdc-language either), first seen in [1], [4], [7].

We now construct an SDLB-machine M_1 which accepts this language. Let $M_1= =(\Gamma_1, K_1, \{a, b, c\}, \delta_1, q_0, \{q_h\})$, where

$$\Gamma_1=\{A, B, C, X, Y\},\ K_1 = \{q_0, q_1, q_2, \dots, q_{11}, q_{12}, \bar{q}, q_h\},$$

δ_1 is defined so that:

1) $\delta_1(q_0, a) = (q_0, A)$, $\delta_1(q_0, A) = (q_1, R)$, $\delta_1(q_1, a) = (q_1, X)$,
$\delta_1(q_1, X) = (q_1, R)$, $\delta_1(q_1, b) = (q_1, B)$, $\delta_1(q_1, B) = (q_2, R)$,
$\delta_1(q_2, b) = (q_2, Y)$, $\delta_1(q_2, Y) = (q_2, R)$.

2) $\delta_1(q_2, c) = (q_2, C)$, $\delta_1(q_2, C) = (q_3, L)$, $\delta_1(q_3, Z) = (q_3, L)$,
for $Z\in\{Y, B, X\}$.

3) $\delta_1(q_3, A) = (q_4, R)$, $\delta_1(q_4, X) = (q_4, A)$, $\delta_1(q_4, A)=(q_5, R)$,
$\delta_1(q_4, B) = (q_{12}, R)$.

4) $\delta_1(q_5, X) = (q_6, R)$, $\delta_1(q_5, B) = (q_7, R)$.

5) $\delta_1(q_6, Y) = (q_3, X)$, $\delta_1(q_6, Z) = (q_6, R)$ for $Z\in\{B, X\}$.

6) $\delta_1(q_7, X) = (q_7, R)$, $\delta_1(q_7, Y) = (q_8, X)$.

7) $\delta_1(q_8, X) = (q_8, L)$, $\delta_1(q_8, B) = (q_9, R)$, $\delta_1(q_9, X) = (q_9, B)$,
$\delta_1(q_9, B) = (q_{10}, R)$.

8) $\delta_1(q_{10}, X) = (q_{11}, R)$, $\delta_1(q_{10}, C) = (q_{10}, R)$.

9) $\delta_1(q_{11}, c) = (q_8, C)$, $\delta_1(q_8, C) = (q_8, L)$, $\delta_1(q_{11}, Z) = (q_{11}, R)$
for $Z\in\{X, C\}$.

10) $\delta_1(q_{10}, c) = (q_{12}, C)$, $\delta_1(q_{12}, C) = (q_h, R)$.

11) $\delta_1(\bar{q}, Z) = (\bar{q}, A)$ for $Z\in(\Gamma_1-\{A\})\cup\{a, b, c\}$,
$\delta_1(q_h, Z) = (\bar{q}, A)$ for $Z\in\Gamma_1$, $\delta_1(\bar{q}, A) = (\bar{q}, R)$.

12) In all other cases for arbitrary $q\in K_1-\{\bar{q}, q_h\}$ and $x\in\Gamma_1\cup\{a, b, c\}$,
$\delta_1(q, x) = (\bar{q}, A)$.

It is easy to see that if $w\in\{a, b, c\}^* - \{a^n b^n c^n/n\geqq 1\} - \{\lambda\}$, then

$$q_0 w \underset{M'}{\vdash} \alpha q', \quad \text{where} \quad \alpha\in\Gamma_1^*, \quad \text{and}$$

$$q' = \begin{cases} q_1 & \text{if} \quad w = a^n, \\ q_2 & \text{if} \quad w = a^n b^m, \\ \bar{q} & \text{otherwise.} \end{cases}$$

Consequently, $w\notin L(M_1)$.

We now check that $w=a^nb^nc^n\in L(M_1)$ for all $n\geqq 1$. Indeed, for $n=1$

$$q_0abc \overset{*}{\underset{M_1}{\vdash}} ABq_2C \overset{*}{\underset{M_1}{\vdash}} q_3ABC \underset{M_1}{\vdash} Aq_4BC \underset{M_1}{\vdash} ABq_{12}C \underset{M_1}{\vdash} ABCq_h.$$

Similarly, for all $n\geqq 1$

$$q_0a^{n+1}b^{n+1}c^{n+1} \overset{*}{\underset{M_1}{\vdash}} AX^nBY^nq_2Cc^n \overset{*}{\underset{M_1}{\vdash}} A^{n+1}BX^{n-1}q_8XCc^n$$

$$\overset{*}{\underset{M_1}{\vdash}} A^{n+1}B^{n+1}C^nq_{10}c \underset{M_1}{\vdash} A^{n+1}B^{n+1}C^nq_{12}C \underset{M_1}{\vdash} A^{n+1}B^{n+1}C^{n+1}q_h.$$

Consequently, $a^nb^nc^n\in L(M_1)$ for all $n\geqq 1$.

§ 6. Simple deterministic Turing-machines

In this section we investigate a special kind of deterministic Turing-machines known as simple deterministic Turing-machines, (abbreviated SDT-machines), and prove that the family of all languages accepted by SDT-machines is the intersection of the two classes $\mathscr{L}_0$ and $\mathscr{L}_p$. Furthermore, we can prove that this family is closed under concatenation, intersection, p-quotient, p-union, and homomorphism h_w; but it is not closed under complementation and union. In this paper we do not prove these statements.

Definition 6.1. a) A deterministic Turing-machine (abbreviated DT-machine) is a 6-tuple $M=(K,\Gamma,\Sigma,\delta,q_0,H)$, where K is the set of states, Γ is the set of tape symbols, one of these, usually denoted by B, is the blank, $\Sigma\subseteqq\Gamma-\{B\}$ is the set of input symbols, $q_0\in K$ is the initial state, $H\subseteqq K$ is the set of final states, and δ: $K\times\Gamma\to K\times(\Gamma-\{B\})\times\{R,L\}$ is the mapping satisfying the following condition: for arbitrary $q\in K$ and $z\in\Gamma$, $\delta(q,z)$ contains exactly one element.

b) We denote a configuration of the DT-machine M by w_1qw_2 or qBw for $w,w_1,w_2\in(\Gamma-\{B\})^*$ and $w_1w_2\neq\lambda$. Let $\underset{M}{\vdash}$ be the relation on configurations of M given as follows. For arbitrary $q\in K, x, y\in\Gamma-\{B\}$ and $w_1,w_2\in(\Gamma-\{B\})^*$

1) $w_1qxw_2 \underset{M}{\vdash} w_1zpw_2$ if $\delta(q,x)=(p,z,R)$,
2) $w_1yqxw_2 \underset{M}{\vdash} w_1pyzw_2$ if $\delta(q,x)=(p,z,L)$,
3) $qxw_2 \underset{M}{\vdash} pBzw_2$ if $\delta(q,x)=(p,z,L)$,
4) $qBw_2 \underset{M}{\vdash} zpw_2$ if $\delta(q,B)=(p,z,R)$,
5) $qBw_2 \underset{M}{\vdash} pBzw_2$ if $\delta(q,B)=(p,z,L)$.

Let $\overset{*}{\underset{M}{\vdash}}$ denote the transitive closure of $\underset{M}{\vdash}$. Finally, define the language accepted by the DT-machine M to be $L(M)=\{w\in\Sigma^*/q_0w\overset{*}{\underset{M}{\vdash}}\alpha p$ for some $p\in H, \alpha\in(\Gamma-\{B\})^*\}$.

Remark. In this definition the tape of the Turing-machine is infinitely extensible to the left, but is totally bounded to the right by the end of the tape. By a carry forward algorithm to the left (M. DAVIS, 1958, [6]), we can prove that this is the equivalent of Turing-machine definition in [1] such that its tape is totally bounded to the left by the end of the tape.

Lemma 6.1. Let L be a type-0 language over the alphabet Σ. Then there is a DT-machine $M'=(K', \Gamma', \Sigma, \delta', q_0', H')$ such that

i) $L=L(M')$,

ii) the mapping δ' satisfies the following condition: for arbitrary $q\in K'$ and $x\in\Gamma'$ if $\delta'(q, x)=(p, z, i)$ for an $i\in\{R, L\}$ then $z\notin\Sigma'$.

Proof. By the Theorem 6.3 in [1], we may assume that $L=L(M)$ for a DT-machine $M=(K, \Gamma, \Sigma, \delta, q_0, H)$.

We now construct a DT-machine M' from M as follows. Let $M'=(K', \Gamma', \Sigma, \delta', q_0', H')$, where $K'=K, \Gamma'=\Gamma\cup\{a'/a\in\Sigma\}, q_0'=q_0, H'=H$, and δ' is defined so that

1) for arbitrary $q\in K$ and $x\in\Gamma$: if $\delta(q, x)=(p, z, i)$ for $i\in\{R, L\}$ then $\delta'(q, x)= =(p, \bar{z}, i)$, where

$$\bar{z}=\begin{cases} z & \text{if} \quad z\in\Gamma-\Sigma-\{B\}, \\ z' & \text{if} \quad z\in\Sigma, \end{cases}$$

2) for arbitrary $q\in K$ and $a\in\Sigma$: $\delta'(q, a')=\delta'(q, a)$.

It is clear that $L(M')=L(M)$ and the condition ii) is satisfied. □

Definition 6.2. a) A simple deterministic Turing-machine (abbreviated SDT-machine), is a 6-tuple $M=(K, \Gamma, \Sigma, \delta, q_0, H)$, where K is the set of states, Γ is the set of tape symbols; one of these, usually denoted by B, is the blank, Σ is the set of input symbols for which $\Sigma\cap\Gamma=\emptyset$, and $\delta: K\times(\Gamma\cup\Sigma)\to K\times(\Gamma-\{B\})\times\{R, L\}$ is the mapping satisfying the following conditions

i) for arbitrary $q\in K-H$ and $x\in\Gamma\cup\Sigma$: $\delta(q, x)$ contains exactly one element,

ii) for each $p\in H$: $\delta(p, a)$ is undefined if $a\in\Sigma$, and it contains exactly one element for all $a\in\Gamma$.

b) The relation $\overset{*}{\underset{M}{\vdash}}$ is defined as in the case of a DT-machine. Finally we define the language accepted by an SDT-machine M to be $L(M)=\{w\in\Sigma^*/q_0w\overset{*}{\underset{M}{\vdash}}\alpha p$ for some $p\in H, \alpha\in(\Gamma-\{B\})^*\}$. A language L is said to be simple deterministic type-0 (abbreviated sd0-language) if $L=L(M)$ for some SDT-machine M. The family of all sd0-languages is denoted by $\mathscr{L}_{\text{sd0}}$.

Theorem 6.2. Let L be any language over the alphabet Σ. L is an sd0-language if and only if L is prefix-free and $L\in\mathscr{L}_0$.

Proof. Part 1. $L\in\mathscr{L}_{\text{sd0}}\to L\in\mathscr{L}_0\cap\mathscr{L}_p$.

Let $L=L(M)$ for an SDT-machine $M=(K, \Gamma, \Sigma, \delta, q_0, H)$. By the definition of δ, we can easily see that $L\in\mathscr{L}_p$. On the other hand, it is easy to see that an SDT-machine is a Turing-machine. Consequently, $L(M)\in\mathscr{L}_0\cap\mathscr{L}_p$.

Part 2. $L\in\mathscr{L}_0\cap\mathscr{L}_p\to L\in\mathscr{L}_{\text{sd0}}$.

By Lemma 6.1, we may assume that $L=L(M)$ for a DT-machine $M=(K, \Gamma, \Sigma, \delta, q_0, H)$ satisfying the condition ii) of Lemma 6.1, i.e., for arbitrary $q \in K$ and $x \in \Gamma$ if $\delta(q, x)=(p, z, i)$ then $z \notin \Sigma$, where $i \in \{R, L\}$. We now construct the SDT-machine M' from M as follows. Let $M'=(K, \Gamma', \Sigma, \delta', q_0, H)$, where $\Gamma'=\Gamma-\Sigma, \delta'$ is defined as

1) for arbitrary $q \in K-H$ and $x \in \Gamma' \cup \Sigma$: $\delta'(q, x)=\delta(q, x)$,

2) for each $p \in H$

$$\delta'(p, a)=\begin{cases}\delta(p, a) & \text{if } a \in \Gamma', \\ \text{undefined} & \text{if } a \in \Sigma.\end{cases}$$

We prove that $L(M')=L(M)$.

($\subseteq$). Let $w \in L(M')$, i.e., $q_0 w \overset{*}{\underset{M'}{\vdash}} \alpha p$ for some $p \in H$. It is clear that: $q_0 w \overset{*}{\underset{M}{\vdash}} \alpha p$ for $p \in H$. Thus, $w \in L(M)$.

($\supseteq$). We shall prove that if $w \notin L(M')$ then $w \notin L(M)$. We have two cases to consider:

Case 1. Let $w=w_1 a w_2$, for $a \in \Sigma, w_1, w_2 \in \Sigma^*$, and $q_0 w_1 a w_2 \overset{*}{\underset{M'}{\vdash}} \alpha p a w_2$ for some $p \in H$.

It is easy to see that: $q_0 w_1 \overset{*}{\underset{M}{\vdash}} \alpha p$ for $p \in H$, i.e., $w_1 \in L(M)$. Since $L(M)$ is prefix-free and $y=a w_2 \neq \lambda$, $w=w_1 y \notin L(M)$.

Case 2. Let $q_0 w \overset{*}{\underset{M'}{\vdash}} \alpha p$ for some $q \in K-H$.

It is clear that: $q_0 w \overset{*}{\underset{M}{\vdash}} \alpha q$ for $q \notin H$. Thus, $w \notin L(M)$. □

Theorem 6.3. a) For every SDLB-machine M, there is an SDT-machine M' such that $L(M')=L(M)$.

b) There is an SDT-machine M_1 such that $L(M_1)$ is not an sdcs-language.

Proof. a) Part a) is implied, by Theorems 5.2., 6.2 and the following statement "A context-sensitive language is of type-0".

b) By Theorem III/9.4 in [7], there is a type-0 language $L \in \{a, b\}^*$ which is not context-sensitive. Without loss of generality, we may assume that $\lambda \notin L$.

First, we can easily check that $L_1=L \cdot \{c\}=\{wc / w \in L\} \in \mathscr{L}_0 \cap \mathscr{L}_p$. Consequently, $L_1 \in \mathscr{L}_{\text{sd0}}$. We now prove that $L_1 \notin \mathscr{L}_1$ (i.e., L_1 is not an sdcs-language either). Assume on the contrary that $L_1 \in \mathscr{L}_1$. We consider the following homomorphism h: $\{a, b, c\}^* \to \{a, b\}^*$ such that

$$h(\lambda)=\lambda, \quad h(a)=a, \quad h(b)=b, \quad h(c)=\lambda.$$

It is clear that if $x \in L_1=L \cdot \{c\}$, then $\lg(h(x))=\lg(x)-1$ (where $\lg(x)$ denotes the length of x). On the other hand, it can be easily seen that if $x \in L_1$ then $\lg(x) \geqq 2$. Consequently, for all $x \in L_1$: $2\lg(h(x))=2(\lg(x)-1) \geqq \lg(x)$, i.e., h is termed a 2-linear erasing with respect to L_1 (this definition can be found in [7]). By Theorem III/10.4 in [7], if $L_1 \in \mathscr{L}_1$ then $L=h(L_1) \in \mathscr{L}_1$, and the contradiction arises. Thus, L is an sd0-language which is not an sdcs-language. □

In the final part we wish to deal with the two memory simple machine that is the equivalent of an SDT-machine.

Definition 6.3. a) A two memory simple machine (abbreviated TS-machine) is a 6-tuple $M=(\Sigma, \Gamma, \Gamma', \delta, z_0, z_0')$, where Σ is the set of input symbols, Γ and Γ' are two sets of pushdown symbols, $z_0 \in \Gamma$, $z_0' \in \Gamma'$ are two initial symbols of two pushdown stores, and the mapping $\delta: \Gamma \times (\Sigma \cup \{\lambda\}) \times \Gamma' \to \Gamma^* \times \Gamma'^*$ satisfies the following conditions: for arbitrary $z \in \Gamma$ and, $z' \in \Gamma'$ either (i) $\delta(z, \lambda, z')$ is undefined and $\delta(z, a, z')$ contains exactly one element for all $a \in \Sigma$; or (ii) $\delta(z, \lambda, z')$ contains exactly one element and $\delta(z, a, z')$ is undefined for all $a \in \Sigma$.

b) A configuration of M is a triplet (α, w, α'), where $w \in \Sigma^*, \alpha \in \Gamma^*, \alpha' \in \Gamma'^*$. We define the operator $\underset{M}{\vdash}$ on configurations of M as follows. For arbitrary $a \in \Sigma \cup \{\lambda\}$, $w \in \Sigma^*$, $z \in \Gamma$, $z' \in \Gamma'$, $\alpha, \beta \in \Gamma^*$ and $\alpha', \beta' \in \Gamma'^*$: $(\alpha z, aw, \alpha' z') \underset{M}{\vdash} (\alpha\beta, w, \alpha'\beta')$ if $\delta(z, a, z')=(\beta, \beta')$. Let $\overset{*}{\underset{M}{\vdash}}$ denote the transitive closure of $\underset{M}{\vdash}$. Finally, we shall be concerned with the acceptance of an input tape by empty pushdown stores. Accordingly, we define the language accepted by a TS-machine M to be

$$L(M) = \{w \in \Sigma^* / (z_0, w, z_0') \overset{*}{\underset{M}{\vdash}} (\lambda, \lambda, \lambda)\}.$$

Theorem 6.4. Let L be any language over the alphabet Σ. L is an sd0-language if and only if L is accepted by some TS-machine M.

Proof. Part 1. Let $L=L(M)$ for an SDT-machine $M=(K, \Gamma, \Sigma, \delta, q_0, H)$. Without loss of generality, we may assume again that: for arbitrary $q \in K$, and $a \in \Sigma \cup \{\lambda\}$ if $\delta(q, a)=(p, z, i)$ then $p \neq q_0$, where $i \in \{R, L\}$. We now construct the TS-machine M_1 from M as follows. Let $M_1=(\Sigma, \Gamma_1, \Gamma_1', \delta_1, q_0, \$)$, where $\Gamma_1=K \cup \Gamma \cup \{\$\}$, $\Gamma_1'=(K-\{q_0\}) \cup \Gamma \cup \{\$\}$ for $\$ \notin K \cup \Gamma$, and δ_1 is defined so that:

1) For arbitrary $q \in K-H-\{q_0\}$ and $a \in \Sigma$:
 a) if $\delta(q_0, a) = (p, x, R)$ then $\delta_1(q_0, a, \$) = (Bxp, \$)$,
 b) if $\delta(q_0, a) = (p, x, L)$ then $\delta_1(q_0, a, \$) = (B, \$xp)$,
 c) if $\delta(q, a) = (p, x, R)$ then $\delta_1(q, a, \$) = (xp, \$)$,
 d) if $\delta(q, a) = (p, x, L)$ then $\delta_1(q, a, \$) = (\lambda, \$xp)$.
2) For arbitrary $p \in H$ and $y \in \Gamma-\{B\}$:
 a) $\delta_1(p, \lambda, \$) = (\lambda, \$)$,
 b) $\delta_1(y, \lambda, \$) = (\lambda, \$)$,
 c) $\delta_1(B, \lambda, \$) = (\lambda, \lambda)$.
3) For arbitrary $q \in K-\{q_0\}$ and $z \in \Gamma$:
 a) if $\delta(q, z) = (p, x, R)$ then $\delta_1(q, \lambda, z) = (xp, \lambda)$,
 b) if $\delta(q, z) = (p, x, L)$ then $\delta_1(q, \lambda, z) = (\lambda, xp)$,
 c) $\delta_1(q_0, \lambda, y) = (\$, \$)$ for all $y \in \Gamma$.
4) For arbitrary $y \in \Gamma-\{B\}$ and $q \in K-\{q_0\}$:
 a) $\delta_1(y, \lambda, q) = (q, y)$,
 b) $\delta_1(B, \lambda, q) = (Bq, B)$.
5) For arbitrary $y_1 \in \Gamma$ and $y_2 \in \Gamma$: $\delta_1(y_1, \lambda, y_2)=(\$, \$)$.
6) For arbitrary $q_1 \in K$ and $q_2 \in K-\{q_0\}$: $\delta_1(q_1, \lambda, q_2)=(\$, \$)$.
7) For each $z \in \Gamma_1'=(K-\{q_0\}) \cup \Gamma \cup \{\$\}$: $\delta_1(\$, \lambda, z)=(\$, \$)$.

It is easy to see that

$$w \in L(M) \leftrightarrow q_0 w \overset{*}{\underset{M}{\vdash}} \alpha p \quad \text{for some} \quad p \in H$$

$$\leftrightarrow \{(q_0, w, \$) \overset{*}{\underset{M_1}{\vdash}} (B\alpha p, \lambda, \$) \quad \text{for} \quad p \in H$$

$$\underset{M_1}{\vdash} (B\alpha, \lambda, \$) \overset{*}{\underset{M_1}{\vdash}} (B, \lambda, \$) \underset{M_1}{\vdash} (\lambda, \lambda, \lambda)\}$$

$$\leftrightarrow w \in L(M_1).$$

Thus, $L = L(M_1)$ for the TS-machine M_1.

Part 2. Let $L = L(M)$ for a TS-machine M.

By the acceptance of an input tape by empty pushdown stores, it can be easily seen that L is prefix-free. On the other hand, by the Church's thesis, $L \in \mathscr{L}_0$. Consequently, $L \in \mathscr{L}_{\mathrm{sd0}}$. □

Finally, we prove that every sd0-language equals to a homomorphic image of the intersection of two simple deterministic context-free languages.

Theorem 6.5. Every sd0-language L can be expressed in the form $L = h(L_1 \cap L_2)$, where h is a homomoprhism and L_1, L_2 are simple context-free languages.

Proof. By Theorem 6.4, we may assume that $L = L(M)$, where $M = (\Sigma, \Gamma, \Gamma', \delta, z_0, z_0')$ is a TS-machine. First, we set:

$$\Sigma_1 = \{x_{[z,z']} / z \in \Gamma, z' \in \Gamma'\}, \bar{\Gamma} = \{[z, z'] / z \in \Gamma, z' \in \Gamma'\}.$$

We now construct two simple machines M_1 and M_2 from M in the following way. Let $M_1 = (\Sigma', \Gamma_1, \delta_1, z_0)$, $M_2 = (\Sigma', \Gamma_2, \delta_2, z_0)$, where $\Sigma' = \Sigma \cup \Sigma_1, \Gamma_1 = \Gamma \cup \bar{\Gamma} \cup \{\$\}$, $\Gamma_2 = \Gamma' \cup \bar{\Gamma} \cup \{\$\}$, and δ_1, δ_2 are defined as follows:

1) For arbitrary $y, z \in \Gamma$ and $y', z' \in \Gamma'$:

a) $$\delta_1(x_{[y,z']}, z) = \begin{cases} [z, z'] & \text{if} \quad y = z \\ \$ & \text{if} \quad y \neq z, \end{cases}$$

b) $$\delta_2(x_{[z\ y']}, z') = \begin{cases} [z, z'] & \text{if} \quad y' = z', \\ \$ & \text{if} \quad y' \neq z', \end{cases}$$

c) $\delta_1(a, z) = \$$, $\delta_2(a, z') = \$$ for all $a \in \Sigma$.

2) For arbitrary $z \in \Gamma$ and $z' \in \Gamma'$:

a) The case where $\delta(z, \lambda, z')$ is defined.

If $\delta(z, \lambda, z') = (\alpha, \alpha')$ then $\delta_1(\lambda, [z, z']) = \alpha$, $\delta_2(\lambda, [z, z']) = \alpha'$.

b) The case where $\delta(z, \lambda, z')$ is undefined.

For every $a \in \Sigma$, if $\delta(z, a, z') = (\alpha, \alpha')$ then $\delta_1(a, [z, z']) = \alpha$, $\delta_2(a, [z, z']) = \alpha'$, and $\delta_1(b, [z, z']) = \$$, $\delta_2(b, [z, z']) = \$$ for all $b \in \Sigma_1$.

3) $\delta_1(\lambda, \$) = \$$, $\delta_2(\lambda, \$) = \$$.

Let h be the homomorphism of Σ' into Σ defined by

$$h(a) = \begin{cases} a & \text{if} \quad a \in \Sigma, \\ \lambda & \text{if} \quad a \in \Sigma_1. \end{cases}$$

We now prove that $L(M) = h(L_1 \cap L_2)$ for $L_1 = L(M_1)$ and $L_2 = L(M_2)$. First, we can easily check that for arbitrary $a \in \Sigma$, $z \in \Gamma$, $z' \in \Gamma'$, α, $\alpha_1 \in \Gamma^*$, β, $\beta_1 \in \Gamma'^*$,

$$(\alpha z, a, \beta z') \overset{*}{\underset{M}{\vdash}} (\alpha_1, \lambda, \beta_1) \quad \text{iff} \quad \begin{cases} \text{there is } u \in \Sigma_1^* \text{ such that} \\ (ua, \alpha z) \overset{*}{\underset{M_1}{\vdash}} (\lambda, \alpha_1) \quad \text{and} \\ (ua, \beta z') \overset{*}{\underset{M_2}{\vdash}} (\lambda, \beta_1). \end{cases} \tag{6.5.1}$$

Then, we prove by induction on the length of $w = a_1 \dots a_n \in \Sigma^*$ that

$$(z_0, a_1 \dots a_n, z_0') \overset{*}{\underset{M}{\vdash}} (\alpha, \lambda, \beta) \quad \text{iff} \quad \begin{cases} \text{there are } u_1, \dots, u_n \in \Sigma_1^* \text{ such that} \\ (u_1 a_1 \dots u_n a_n, z_0) \overset{*}{\underset{M_1}{\vdash}} (\lambda, \alpha) \quad \text{and} \\ (u_1 a_1 \dots u_n a_n, z_0') \overset{*}{\underset{M_2}{\vdash}} (\lambda, \beta). \end{cases} \tag{6.5.2}$$

Indeed, the case where $w = a \in \Sigma$ is trivial.

Assume that statement (6.5.2) is valid for all $w \in \Sigma^*$ with $\lg(w) < n$. We now consider the word $w = a_1 \dots a_{n-1} a_n$, and let $w_1 = a_1 \dots a_{n-1}$. Since $\lg(w_1) < n$, statement (6.5.2) is true and we have

$$(z_0, w_1, z_0') \underset{M}{\vdash} (\alpha_1 z, \lambda, \beta_1 z') \quad \text{iff} \quad \begin{cases} \text{there are } u_1, \dots, u_{n-1} \in \Sigma_1 \text{ such that} \\ (u_1 a_1 \dots u_{n-1} a_{n-1}, z_0) \overset{*}{\underset{M_1}{\vdash}} (\lambda, \alpha_1 z) \\ (u_1 a_1 \dots u_{n-1} a_{n-1}, z_0') \overset{*}{\underset{M_2}{\vdash}} (\lambda, \beta_1 z'). \end{cases}$$

On the other hand, by statement (6.5.1), we can easily see that

$$(\alpha_1 z, a_n, \beta_1 z') \overset{*}{\underset{M}{\vdash}} (\alpha, \lambda, \beta) \quad \text{iff} \quad \begin{cases} \text{there is } u_n \in \Sigma_1^* \text{ such that} \\ (u_n a_n, \alpha_1 z) \overset{*}{\underset{M_1}{\vdash}} (\lambda, \alpha), \quad \text{and} \\ u_n a_n, \beta_1 z') \overset{*}{\underset{M_2}{\vdash}} (\lambda, \beta). \end{cases}$$

Thus, statement (6.5.2) holds. Finally, for $w=a_1\dots a_n\in\Sigma^*$

$$w = a_1\dots a_n\in L(M) \quad \text{iff} \quad (z_0, a_1\dots, a_n, z_0') \overset{*}{\underset{M}{\vdash}} (\lambda, \lambda, \lambda)$$

$$\text{iff} \quad \begin{cases} \text{there are } u_1, \dots, u_n\in\Sigma_1 \text{ such that} \\ (u_1a_1\dots u_na_n, z_0) \overset{*}{\underset{M_1}{\vdash}} (\lambda, \lambda) \\ (u_1a_1\dots u_na_n, z_0') \overset{*}{\underset{M_2}{\vdash}} (\lambda, \lambda) \end{cases}$$

$$\text{iff} \quad \begin{cases} \text{there are } u_1, \dots, u_n\in\Sigma_1^* \text{ such that} \\ u_1a_1\dots u_na_n\in L_1\cap L_2 \text{ and} \\ h(u_1a_1\dots u_na_n) = a_1\dots a_n\in h(L_1\cap L_2). \quad \square \end{cases}$$

Corollary 6.6. Every sd0-language can be expressed in the form $L=h(L_1\cap L_2)$, where h is a homomorphism, and L_1, L_2 are two sdc-languages.

Acknowledgement. The author would like to thank J. Demetrovics, I. Peák and Gy. Révész forth eir careful readings of several versions of the manuscript and their many helpful suggestions for its revision. This paper is very much improved by their contributions.

COMPUTER AND AUTOMATION INSTITUTE
HUNGARIAN ACEDEMY OF SCIENCE
KENDE U. 13-17.
BUDAPEST, HUNGARY
H-1502

References

[1] Hopcroft, J., J. Ullman, Formal languages and their relation of automata, Addison-Wesley, 1969.
[2] Ginsburg, S., A. Greibach, Deterministic context-free languages, *Inform. and Control,* v. 9, 1966, pp. 620—648.
[3] Friedman, E. P., Simple context-free languages and free monadiç recursion schemes, *Math. Systems Theory,* v. 11, 1971, pp. 9—28.
[4] Révész, Gy., Bevezetés a formális nyelvek elméletébe, Akadémiai Kiadó, Budapest, 1979.
[5] Kuroda, S. Y., Classes of languages and linear-bounded automata, *Inform. and Control,* v. 7, 1964, pp. 207—223.
[6] Davis, M., Computability and unsolvability, McGraw-Hill, New York, 1958.
[7] Salomaa, A., Formal languages, Academic Press, 1973.

(Received Oct. 31, 1980)

Maximal families of restricted subsets of a finite set

By H.-D. O. F. GRONAU and CHR. PROSKE

1. Introduction

Let R be the set of the first r natural numbers, i.e. $R=\{1, 2, \ldots, r\}$. Furthermore, let a and b be integers with $0 \leqq a \leqq b \leqq r$, $a \neq r$, $b \neq 0$ [1]. Finally let $\mathcal{F}$ be an n-tuple $(X_1, X_2, \ldots, X_n)$ of subsets of R satisfying $a \leqq |X_i| \leqq b$ $(i=1, 2, \ldots, n)$.

An ordered pair (X, Y) of subsets of R has the property

— **A**: if and only if there is a $v \in R$: $v \notin X, v \in Y$,
— **B**: if and only if there is a $v \in R$: $v \in X, v \notin Y$,
— **C**: if and only if there is a $v \in R$: $v \notin X, v \notin Y$,
— **D**: if and only if there is a $v \in R$: $v \in X, v \in Y$.

Let $\mathbf{P}=\mathbf{P}(\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{D})$ be an arbitrary Boolean expression of **A**, **B**, **C**, **D**. $\mathcal{F}$ is said to be a **P**-family if and only if all ordered pairs (X_i, X_j), $1 \leqq i < j \leqq n$, satisfy the condition **P**. If there is a maximal value of n, we will denote this by $n_{a,b}(\mathbf{P}, r)$. Many well-known results in extremal set theory can be expressed in our concept. We will only mention the following two classical theorems.

1) SPERNER's theorem [13]: $\quad n_{0,r}(\mathbf{AB}, r)=\binom{r}{[r/2]}$, [2]

2) ERDŐS-KO-RADO-Theorem [3]: $\quad n_{0,k}(\mathbf{ABD}, r)=\binom{r-1}{k-1}$ if $k \leqq r/2$.

In [5] the first-named author considered all 2^{16} possible Boolean expressions **P**, found those **P**'s for which $n_{0,r}(\mathbf{P}, r)$ [3] eixsts, and determined in all these cases $n_{0,r}(\mathbf{P}, r)$ exactly [4]. In the present paper we consider the same problem for all **P**'s and $n_{a,b}(\mathbf{P}, r)$.

The results in Sections 2 and 3 are close to the corresponding results for $n(\mathbf{P}, r)$. Thus, the proofs are sketched only or are omitted.

[1] For simplification we exclude the pathological cases $a=r$ resp. $b=0$.
[2] **AB** will be used in place of $\mathbf{A} \wedge \mathbf{B}$ and $\overline{\mathbf{A}}$ denotes *non* **A**.
[3] In [5] the notation $n(\mathbf{P}, r)$ is used for $n_{0,r}(\mathbf{P}, r)$.
[4] With exception of only one case, where bounds and the asymptotic are found.

2. Existence of $n_{a,b}(\mathbf{P}, r)$

We set $n_{a,b}(\mathbf{0}, r)=1$ for all a, b, where $\mathbf{0}$ denotes the empty condition. In all what follows let $\mathbf{P}\neq\mathbf{0}$.

Then there is a nonempty canonical alternative normal form $CANF(\mathbf{P})$ of $\mathbf{P}$. If $\mathbf{A}'$ is an elementary conjunction of $\{\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{D}\}$, then $\mathbf{A}'\in CANF(\mathbf{P})$ means that $\mathbf{A}'$ is one of the conjunctions of $CANF(\mathbf{P})$.

Since no pair (X, Y) satisfies $\bar{\mathbf{A}}\bar{\mathbf{B}}\bar{\mathbf{C}}\bar{\mathbf{D}}$ and the only pairs satisfying $\bar{\mathbf{A}}\bar{\mathbf{B}}\bar{\mathbf{C}}\mathbf{D}$ or $\bar{\mathbf{A}}\bar{\mathbf{B}}\mathbf{C}\bar{\mathbf{D}}$ are (R, R) or $(\emptyset, \emptyset)$, respectively, it follows

Lemma 1.
(i) $n_{a,b}(\bar{\mathbf{A}}\bar{\mathbf{B}}\bar{\mathbf{C}}\bar{\mathbf{D}}, r)=1$,
$n_{a,b}(\mathbf{P}\vee\bar{\mathbf{A}}\bar{\mathbf{B}}\bar{\mathbf{C}}\bar{\mathbf{D}}, r)=n_{a,b}(\mathbf{P}, r)$,
(ii) $n_{a,b}(\bar{\mathbf{A}}\bar{\mathbf{B}}\bar{\mathbf{C}}\mathbf{D}, r)=1$ *if* $b<r$,
$n_{a,b}(\mathbf{P}\vee\bar{\mathbf{A}}\bar{\mathbf{B}}\bar{\mathbf{C}}\mathbf{D}, r)=n_{a,b}(\mathbf{P}, r)$,
(iii) $n_{a,b}(\bar{\mathbf{A}}\bar{\mathbf{B}}\mathbf{C}\bar{\mathbf{D}}, r)=1$ *if* $0<a$,
$n_{a,b}(\mathbf{P}\vee\bar{\mathbf{A}}\bar{\mathbf{B}}\mathbf{C}\bar{\mathbf{D}}, r)=n_{a,b}(\mathbf{P}, r)$.

Theorem 1. $n_{a,b}(\mathbf{P}, r)$ *does not exist if and only if*
(i) $\bar{\mathbf{A}}\bar{\mathbf{B}}\mathbf{C}\mathbf{D}\in CANF(\mathbf{P})$ *or*
(ii) $\bar{\mathbf{A}}\bar{\mathbf{B}}\bar{\mathbf{C}}\mathbf{D}\in CANF(\mathbf{P})$ *and* $b=r$ *or*
(iii) $\bar{\mathbf{A}}\bar{\mathbf{B}}\mathbf{C}\bar{\mathbf{D}}\in CANF(\mathbf{P})$ *and* $a=0$.

Hence, if $n_{a,b}(\mathbf{P}, r)$ exists, $\bar{\mathbf{A}}\bar{\mathbf{B}}\bar{\mathbf{C}}\bar{\mathbf{D}}$, $\bar{\mathbf{A}}\bar{\mathbf{B}}\bar{\mathbf{C}}\mathbf{D}$, and $\bar{\mathbf{A}}\bar{\mathbf{B}}\mathbf{C}\bar{\mathbf{D}}$ can be omitted in $CANF(\mathbf{P})$.

3. Some reductions

The following table gives an equivalent description of some conditions $\mathbf{P}$ in terms of ordered pairs (X, Y).

$\mathbf{P}$	$\leftrightarrow$	(X, Y)
$\mathbf{A}\bar{\mathbf{B}}\bar{\mathbf{C}}\bar{\mathbf{D}}$		$(\emptyset, R)$
$\bar{\mathbf{A}}\mathbf{B}\bar{\mathbf{C}}\bar{\mathbf{D}}$		$(R, \emptyset)$
$\mathbf{A}\bar{\mathbf{B}}\mathbf{C}\bar{\mathbf{D}}$		$(\emptyset, Z)$
$\bar{\mathbf{A}}\mathbf{B}\mathbf{C}\bar{\mathbf{D}}$		$(Z, \emptyset)$
$\mathbf{A}\bar{\mathbf{B}}\bar{\mathbf{C}}\mathbf{D}$		(Z, R)
$\bar{\mathbf{A}}\mathbf{B}\bar{\mathbf{C}}\mathbf{D}$		(R, Z)

(1)

where $Z\subseteq R$, $Z\neq\emptyset$, $Z\neq R$. The remaining 6 conditions $\mathbf{ABCD}$, $\mathbf{ABC}\bar{\mathbf{D}}$, $\mathbf{AB}\bar{\mathbf{C}}\mathbf{D}$, $\mathbf{A}\bar{\mathbf{B}}\mathbf{CD}$, $\bar{\mathbf{A}}\mathbf{BCD}$, $\mathbf{AB}\bar{\mathbf{C}}\bar{\mathbf{D}}$ are conditions for pairs (X, Y) with $\{X, Y\}\cap\{\emptyset, R\}=\emptyset$.

If $a=0$ and $b=r$ we refer to [5]. Let $a>0$ or $b<r$.

Then no pair (X, Y) can satisfy $\mathbf{A}\bar{\mathbf{B}}\bar{\mathbf{C}}\bar{\mathbf{D}}$ or $\bar{\mathbf{A}}\mathbf{B}\bar{\mathbf{C}}\bar{\mathbf{D}}$ and we may omit these conjunctions in $\mathbf{P}$. Let $\mathbf{P}=\mathbf{P}'\vee\mathbf{P}''$, where $\mathbf{P}''$ contains exactly those conjunctions which are in (1).

Theorem 2.

	$\mathbf{P}'\equiv 0$	$\mathbf{P}'\not\equiv 0$
$a>0$ $b=r$	$n_{a,r}(\mathbf{P},r)=\begin{cases}2^{(*)}\\ 1 \textit{ otherwise}\end{cases}$	$n_{a,r}(\mathbf{P},r)=\begin{cases}n_{a,r-1}(\mathbf{P}',r)+1 & \textit{if } \mathbf{A\overline{B}\,\overline{C}D}\in CANF(\mathbf{P}'') \textit{ or } \mathbf{\overline{A}B\overline{C}D}\in CANF(\mathbf{P}'')\\ n_{a,r-1}(\mathbf{P}',r) & \textit{otherwise}\end{cases}$
$b<r$ $a=0$	$n_{0,b}(\mathbf{P},r)=\begin{cases}2^{(**)}\\ 1 \textit{ otherwise}\end{cases}$	$n_{0,b}(\mathbf{P},r)=\begin{cases}n_{1,b}(\mathbf{P}',r)+1 & \textit{if } \mathbf{A\overline{B}C\overline{D}}\in CANF(\mathbf{P}'') \textit{ or } \mathbf{\overline{A}BC\overline{D}}\in CANF(\mathbf{P}'')\\ n_{1,b}(\mathbf{P}',r) & \textit{otherwise}\end{cases}$
$a>0$ $b<r$	$n_{a,b}(\mathbf{P},r)=1$	$n_{a,b}(\mathbf{P},r)=n_{a,b}(\mathbf{P}',r)$

(*) if $\mathbf{A\overline{B}\,\overline{C}D}\in CANF(\mathbf{P}'')$ or $\mathbf{\overline{A}B\overline{C}D}\in CANF(\mathbf{P}'')$;
(**) if $\mathbf{A\overline{B}C\overline{D}}\in CANF(\mathbf{P}'')$ or $\mathbf{\overline{A}BC\overline{D}}\in CANF(\mathbf{P}'')$.

Hence, we have to consider only alternatives **P** over $\{\mathbf{ABCD}, \mathbf{ABC\overline{D}}, \mathbf{AB\overline{C}D}, \mathbf{A\overline{B}CD}, \mathbf{\overline{A}BCD}, \mathbf{AB\overline{C}\,\overline{D}}\}$ and we may assume $a>0$, $b<r$.

Lemma 2.

(i) $n_{a,b}(\mathbf{P}(\mathbf{A},\mathbf{B},\mathbf{C},\mathbf{D}),r)=n_{a,b}(\mathbf{P}(\mathbf{B},\mathbf{A},\mathbf{C},\mathbf{D}),r)$,
(ii) $n_{a,b}(\mathbf{P}(\mathbf{A},\mathbf{B},\mathbf{C},\mathbf{D}),r)=n_{r-b,r-a}(\mathbf{P}(\mathbf{A},\mathbf{B},\mathbf{D},\mathbf{C}),r)$,
(iii) $n_{a,b}((\mathbf{A\overline{B}}\vee\mathbf{\overline{A}B})\mathbf{P}'(\mathbf{C},\mathbf{D}),r)=n_{a,b}(\mathbf{A\overline{B}P}'(\mathbf{C},\mathbf{D}),r)$,
(iv) $n_{a,b}((\mathbf{A}\vee\mathbf{B})\mathbf{P}'(\mathbf{C},\mathbf{D}),r)=n_{a,b}(\mathbf{AP}'(\mathbf{C},\mathbf{D}),r)$,
(v) $n_{a,b}(\mathbf{P}''\vee\mathbf{A\overline{B}CD}\vee\mathbf{\overline{A}BCD},r)=n_{a,b}(\mathbf{P}''\vee\mathbf{A\overline{B}CD},r)$,
(vi) $n_{a,b}(\mathbf{P}''\vee\mathbf{\overline{A}BCD},r)=n_{a,b}(\mathbf{P}''\vee\mathbf{A\overline{B}CD},r)$,
(vii) $n_{a,b}(\mathbf{P}''',r)=n_{r-b,r-a}(\mathbf{P}''',r)$,
(viii) $n_{a,b}(\mathbf{P}'''\vee\mathbf{AB\overline{C}D},r)=n_{r-b,r-a}(\mathbf{P}'''\vee\mathbf{ABC\overline{D}},r)$,
(ix) $n_{a,b}(\mathbf{P}'''\vee\mathbf{AB\overline{C}D}\vee\mathbf{ABC\overline{D}},r)=n_{r-b,r-a}(\mathbf{P}'''\vee\mathbf{AB\overline{C}D}\vee\mathbf{ABC\overline{D}},r)$,

where $\mathbf{P}'$ is an arbitrary Boolean function in 2 arguments,
$\mathbf{P}''$ is any alternative over $\{\mathbf{ABCD}, \mathbf{AB\overline{C}D}, \mathbf{ABC\overline{D}}, \mathbf{AB\overline{C}\,\overline{D}}\}$, and
$\mathbf{P}'''$ is any alternative over $\{\mathbf{ABCD}, \mathbf{AB\overline{C}\,\overline{D}}, \mathbf{A\overline{B}CD}\}$.

4. $n_{a,b}(\mathbf{P},r)$ for the reduced P's

For simplification we use $\mathbf{MN}\vee\mathbf{M\overline{N}}=\mathbf{M}$ and $\mathbf{M}\vee\mathbf{MN}=\mathbf{M}$. Now we consider the three general cases:

1) $a\leqq b<r/2$,
2) $a\leqq r/2\leqq b$,
3) $r/2<a\leqq b$.

The third case may be reduced to the first one using Lemma 2, (ii). If $a \leqq b < r/2$, then obviously no pair (X, Y) of $\mathcal{F}$ can satisfy $\mathbf{AB\bar{C}D}$ or $\mathbf{AB\bar{C}\bar{D}}$, i.e., these two conjunctions may be omitted, or if $CANF(\mathbf{P})$ has only conjunctions of these ones, $n_{a,b}(\mathbf{P}, r) = 1$ follows immediately.

Case 1. $a \leqq b < r/2$. $CANF(\mathbf{P})$ contains only conjunctions of $\{\mathbf{ABCD}, \mathbf{A\bar{B}CD}, \mathbf{ABC\bar{D}}\}$. Thus, only 7 $\mathbf{P}$'s are possible ($\mathbf{P} \equiv \mathbf{0}$ was excluded at the beginning).

No.	P	$n_{a,b}(\mathbf{P}, r)$	reference/remark
1.1	$\mathbf{A\bar{B}CD}$	$b-a+1$	$\mathcal{F}$ forms a chain.
1.2	$\mathbf{ABCD}$	$\binom{r-1}{b-1}$	Erdős, Ko, Rado [3] or Greene, Katona, Kleitman [4].
1.3	$\mathbf{ABC\bar{D}}$	$\left[\frac{r}{a}\right]$	The sets of $\mathcal{F}$ have to be disjoint.
1.4	$\mathbf{ACD}$	$\sum_{i=a}^{b}\binom{r-1}{i-1}$	Hilton [7].
1.5	$\mathbf{A\bar{B}CD} \vee$ $\vee \mathbf{ABC\bar{D}}$	$2r-]r/b[$ if $a=1$ $r-(a-1)]r/b[$ if $a \geqq 2$, $a \leqq r-b(]r/b[-1)$ $(b-a+1)(]r/b[-1)$ if $a \geqq 2$, $a > r-b(]r/b[-1)$	see Section 5.
1.6	$\mathbf{ABC}$	$\binom{r}{b}$	Lubell [9], Meshalkin [10], Yamamoto [14].
1.7	$\mathbf{ABC} \vee$ $\vee \mathbf{ACD}$	$\sum_{i=a}^{b}\binom{r}{i}$	Every pair (X, Y), $a \leqq \lvert X \rvert \leqq \lvert Y \rvert \leqq b$, satisfies the condition.

Case 2. $a \leqq r/2 \leqq b$.

Using the statements of Lemma 2 we may reduce all possible conditions to 23 types. More precisely, by Lemma 2 (v) and (vi), we may omit $\mathbf{\bar{A}BCD}$ if $\mathbf{A\bar{B}CD} \in CANF(\mathbf{P})$ or replace $\mathbf{\bar{A}BCD}$ by $\mathbf{A\bar{B}CD}$ if $\mathbf{A\bar{B}CD} \notin CANF(\mathbf{P})$. Furthermore, if $\mathbf{AB\bar{C}D} \in CANF(\mathbf{P})$ and $\mathbf{ABC\bar{D}} \notin CANF(\mathbf{P})$, $\mathbf{AB\bar{C}D}$ can be replaced by $\mathbf{ABC\bar{D}}$ according to Lemma 2 (viii). This procedure is the same as in [5]. We also use this notation. Many results are well-known, others are very simple. But there are some really new problems.

Most of them have been solved. The proofs are given in Sections 5 and 6. Finally some open problems are presented in Section 7.

No.	P	$n_{a,b}(\mathbf{P},r)$	reference/remark
2.1	$\mathbf{ACD}$	?	see Section 7.
2.2	$\mathbf{AB}$	$\binom{r}{[r/2]}$	SPERNER [13], LUBELL [9], MESHALKIN [10], or YAMAMOTO [14].
2.3	$\mathbf{ABC}$	$\binom{r}{[(r-1)/2]}$	MILNER [11] or GREENE, KATONA, KLEITMAN [4].
2.4	$\mathbf{ABCD}$	$\binom{r-1}{[(r-2)/2]}$	KATONA [8], SCHÖNHEIM [12], or GRONAU [6].
2.5	$\mathbf{ABC}\vee$ $\vee\mathbf{ABD}$	$\binom{r-1}{[(r-1)/2]}$	CLEMENTS, GRONAU [1].
2.6	$\mathbf{ABC}\vee$ $\vee\mathbf{ACD}\vee$ $\vee\mathbf{AB\bar{D}}$	$\begin{cases}\sum\limits_{i=a}^{r/2}\binom{r}{i} & \text{if } r \text{ is even}\\ \sum\limits_{i=a}^{(r-1)/2}\binom{r}{i}+\binom{r-1}{(r+1)/2} & \text{if } r \text{ is odd}\end{cases}$ *if* $a \leqq r-b$; ? *if* $a > r-b$	see Section 6. see Section 7.
2.7	$\mathbf{AB\bar{C}\bar{D}}$	2	clear.
2.8	$\mathbf{A\bar{B}CD}$	$b-a+1$	$\mathscr{F}$ forms a chain.
2.9	$\mathbf{A\bar{B}CD}\vee$ $\vee\mathbf{AB\bar{C}\bar{D}}$	$\begin{cases}2 & \text{if } a=b=r/2\\ b-a+1 & \text{otherwise}\end{cases}$	It follows by 2.7 and 2.8.
2.10	$\mathbf{AB\bar{D}}$	$[r/a]$	The sets of $\mathscr{F}$ are disjoint.
2.11	$\mathbf{ABC\bar{D}}$	$\begin{cases}[r/a] & \text{if } a \neq r/2\\ 1 & \text{if } a = r/2\end{cases}$	
2.12	$\mathbf{AB\bar{C}}\vee$ $\vee\mathbf{AB\bar{D}}$	$[r/c]$, $c = \min(a, r-b)$	In [5] it was proved that $\mathscr{F}$ satisfies $\mathbf{AB\bar{C}}$ ($\mathbf{AB\bar{C}D}$) or $\mathscr{F}$ satisfies $\mathbf{AB\bar{D}}$($\mathbf{ABC\bar{D}}$). 2.10 resp. 2.11 implies the result.
2.13	$\mathbf{AB\bar{C}D}\vee$ $\vee\mathbf{ABC\bar{D}}$	$\begin{cases}1 \text{ if } a=b=r/2\\ [r/c],\ c=\min(a,r-b) \text{ otherwise}\end{cases}$	
2.14	$\mathbf{ABC}\vee$ $\vee\mathbf{ABD}\vee$ $\vee\mathbf{ACD}$	$\begin{cases}\sum\limits_{i=a}^{r-b-1}\binom{r}{i}+\frac{1}{2}\sum\limits_{i=r-b}^{b}\binom{r}{i} & \text{if } a \leqq r-b\\ \frac{1}{2}\sum\limits_{i=a}^{r-a}\binom{r}{i}+\sum\limits_{i=r-a+1}^{b}\binom{r}{i} & \text{if } a > r-b\end{cases}$	$\mathscr{F}$ contains no set and its complement.

No.	P	$n_{a,b}(\mathbf{P},r)$	reference/remark
2.15	$\mathbf{ABC}\vee$ $\vee\mathbf{ACD}$	$\begin{cases} \sum_{i=a}^{[\frac{r-1}{2}]}\binom{r}{i}+\begin{cases}0 \text{ if } r \text{ is odd}\\ \binom{r-1}{r/2-1} \text{ if } r \text{ is even}\end{cases} & \text{if } a \leqq r-b\\ \sum_{i=a}^{b}\binom{r-1}{i} & \text{if } a > r-b\end{cases}$	see Section 6. HILTON [7].
2.16	$\mathbf{AB}\vee$ $\vee\mathbf{ACD}$	$\sum_{i=a}^{b}\binom{r}{i}$	Every pair satisfies this condition.
2.17	$\mathbf{ABC}\vee$ $\vee\mathbf{AB\bar{D}}$	$\binom{r}{[r/2]}$	$\mathscr{F}$ satisfies **AB** too. Indeed, $\mathscr{F}=\{X\colon X\subseteqq R,$ $\lvert X\rvert=[r/2]\}$ has that cardinality.
2.18	$\mathbf{ABCD}\vee$ $\vee\mathbf{AB\bar{C}\bar{D}}$	$2\binom{r-1}{[(r-2)/2]}$	Omitting complements $\mathscr{F}$ satisfies **ABCD** (see 2.4). $\mathscr{F}=\{X\colon X\subseteqq R, v\in X,$ $\lvert X\rvert=[r/2]\}\cup\{X\colon X\subseteqq R,$ $v\notin X, \lvert X\rvert=\{[(r+1)/2]\},$ where $v\in R$ is fixed, has the desired cardinality.
2.19	$\mathbf{A\bar{B}CD}\vee$ $\vee\mathbf{AB\bar{D}}$	$2r-2$ *if* $a=1$; $r-2a+2$ *if* $2\leqq a\leqq r-b$; $b-a+1$ *if* $2\leqq a>r-b$	see Section 5.
2.20	$\mathbf{A\bar{B}CD}\vee$ $\vee\mathbf{ABC\bar{D}}$	$2r-3$ *if* $a=1$; $r-2a+1$ *if* $2\leqq a<r-b$; $b-a+1$ *if* $2\leqq a\geqq r-b$	see Section 5.
2.21	$\mathbf{A\bar{B}CD}\vee$ $\vee\mathbf{AB\bar{C}D}\vee$ $\vee\mathbf{ABC\bar{D}}$	$2r-3$ *if* $a=1$ *or* $b=r-1$; $r-2a+1$ *if* $a\geqq 2,$ $b\leqq r-2,$ $a\leqq r-b$; $2b-r+1$ *if* $a\geqq 2,$ $b\leqq r-2,$ $a\geqq r-b$	see Section 5.
2.22	$\mathbf{A\bar{B}CD}\vee$ $\vee\mathbf{AB\bar{C}}\vee$ $\vee\mathbf{AB\bar{D}}$	$4r-6$ *if* $a=1, b=r-1$; $r+2b-2$ *if* $a=1, b<r-1$; $3r-2a-2$ *if* $a>1, b=r-1$; $2(b-a+1)$ *if* $a>1, b<r-1$	see Section 5.
2.23	$\mathbf{ACD}\vee$ $\vee\mathbf{AB\bar{C}\bar{D}}$	?	see Section 7.

5. Proofs of 1.5, 2.19, 2.20, 2.21, 2.22

In order to give examples of maximal families we use the following notations

$$\mathscr{D}_1=\{X\colon X=\{t\},\ 1\leqq t\leqq r\},$$
$$\mathscr{D}_2=\{X\colon R\setminus X\in\mathscr{D}_1\},$$
$$\mathscr{D}_3(p,q,s)=\{X\colon X=\{p+1,p+2,\dots,p+t\},\ q\leqq t\leqq s,\ p+t\leqq r\},$$
$$\mathscr{D}_4(p,q)=\{X\colon X=\{t+1,t+2,\dots,r\},\ t=p,p-1,p-2,\dots,q\}.$$

For all these conditions we have

$$n_{1,b}(\mathbf{P}, r) \leqq n_{2,b}(\mathbf{P}, r)+r. \tag{2}$$

Thus, $n_{a,b}(\mathbf{P}, r)$, $a \geqq 2$, implies an upper bound for $n_{1,b}(\mathbf{P}, r)$.

5.1. Let $\mathbf{P}=\mathbf{A\bar{B}CD} \vee \mathbf{AB\bar{D}}$ (2.19, 1.5).

Denote by $\mathscr{F}$ a maximal $(\mathbf{A\bar{B}CD} \vee \mathbf{AB\bar{D}})$-family with $a \leqq |X| \leqq b$ for all $X \in \mathscr{F}$. Then for every pair (X, Y) of $\mathscr{F}$ we have $X \subset Y$ or $X \cap Y = \emptyset$. Hence, there is a unique subfamily $\mathscr{G}(\mathscr{F}) \subseteqq \mathscr{F}$ satisfying

— $X \cap Y = \emptyset$ for all pairs (X, Y) with $X, Y \in \mathscr{G}(\mathscr{F})$,
— for all $X \in \mathscr{F} \setminus \mathscr{G}(\mathscr{F})$ there is an element $Y \in \mathscr{G}(\mathscr{F})$ with $X \subset Y$.

If $X \in \mathscr{G}(\mathscr{F})$, then $\mathscr{H}(X)=\{Y\colon Y \in \mathscr{F}, Y \subset X\}$. Thus,

— $\mathscr{F}=\mathscr{G}(\mathscr{F}) \cup \bigcup_{X \in \mathscr{G}(\mathscr{F})} \mathscr{H}(X)$,
— $\mathscr{H}(X)$ satisfies $\mathbf{A\bar{B}CD} \vee \mathbf{AB\bar{D}}$, and
— all $Y \in \mathscr{H}(X)$ satisfy $a \leqq |Y| \leqq |X|-1$.

Then

$$|\mathscr{F}| \leqq |\mathscr{G}(\mathscr{F})|+\sum_{X \in \mathscr{G}(\mathscr{F})} n_{a,|X|-1}(\mathbf{A\bar{B}CD} \vee \mathbf{AB\bar{D}}, |X|). \tag{3}$$

Now we prove

Lemma 3. $n_{a,r-1}(\mathbf{A\bar{B}CD} \vee \mathbf{AB\bar{D}}, r)=r-a$ *for* $3 \leqq a+1 \leqq r$.

Proof. The proof is given by induction on r for arbitrary, but fixed a, $r \geqq a+1$.
1. $r=a+1$. The statement is true, clearly.
2. We obtain for every maximal family $\mathscr{F}$, by (3),

$$|\mathscr{F}| \leqq |\mathscr{G}(\mathscr{F})|+\sum_{X \in \mathscr{G}(\mathscr{F})} (|X|-a).$$

If $|\mathscr{G}(\mathscr{F})|=1$, then $\sum_{X \in \mathscr{G}(\mathscr{F})} |X| \leqq r-1$ and $|\mathscr{F}| \leqq r-a$.

If $|\mathscr{G}(\mathscr{F})| \geqq 2$, then $\sum_{X \in \mathscr{G}(\mathscr{F})} |X| \leqq r$ and

$$|\mathscr{F}| \leqq r-(a-1)|\mathscr{G}(\mathscr{F})| \leqq r-2(a-1) \leqq r-a.$$

Indeed, $\mathscr{D}_3(0, a, r-1)$ is a $(\mathbf{A\bar{B}CD} \vee \mathbf{AB\bar{D}})$-family with the desired cardinality. □

Now we return to the general a, b-case.
Lemma 3 and (3) yield for a maximal $(\mathbf{A\bar{B}CD} \vee \mathbf{AB\bar{D}})$-family $\mathscr{F}$

$$|\mathscr{F}| = \sum_{X \in \mathscr{G}(\mathscr{F})} |X|-(a-1)|\mathscr{G}(\mathscr{F})|.$$

If $|\mathscr{G}(\mathscr{F})| \leqq]r/b[-1$, then $|X| \leqq b$ implies

$$|\mathscr{F}| \leqq (b-a+1)|\mathscr{G}(\mathscr{F})| \leqq (b-a+1)(]r/b[-1). \tag{4}$$

If $|\mathscr{G}(\mathscr{F})| \geqq]r, b[$, then $\sum_{X \in \mathscr{G}(\mathscr{F})} |X| = \left|\bigcup_{X \in \mathscr{G}(\mathscr{F})} X\right| \leqq r$

and

$$|\mathscr{F}| \leqq r-(a-1)|\mathscr{G}(\mathscr{F})| \leqq r-(a-1)]r/b[. \tag{5}$$

Simple verification shows that the upper estimation of (4) is not smaller than that one of (5) iff

$$a > r-b(]r/b[-1).$$

Indeed, $\mathcal{D}_5(a,b) = \bigcup_{t=0,1,\dots} \mathcal{D}_3(bt,a,b)$ confirms in both cases that these upper bounds are the desired results. Thus, 1.5 is proven if $a \geqq 2$ remarking that **C** is always satisfied. The case $a=1$ follows by (2) and the example $\mathcal{D}_1 \cup \mathcal{D}_5(1,b)$. Moreover, 2.19 is proven, note $]r/b[=2$ for $b \geqq r/2$. Also here the case $a=1$ follows by (2) and the example $\mathcal{D}_1 \cup \mathcal{D}_5(1,b)$.

5.2. Let $\mathbf{P} = \mathbf{A\bar{B}CD} \vee \mathbf{ABC\bar{D}}$ (2.20).

In analogy to the preceding case a special subfamily $\mathcal{G}(\mathcal{F})$ exists and we obtain for a maximal family $\mathcal{F}$

$$|\mathcal{F}| \leqq |\mathcal{G}(\mathcal{F})| + \sum_{X \in \mathcal{G}(\mathcal{F})} n_{a,|X|-1}(\mathbf{A\bar{B}CD} \vee \mathbf{AB\bar{D}}, |X|). \tag{6}$$

We remark that $\mathcal{H}(X)$ satisfies $\mathbf{A\bar{B}CD} \vee \mathbf{AB\bar{D}}$, not necessarily $\mathbf{A\bar{B}CD} \vee \mathbf{ABC\bar{D}}$.

Lemma 3 implies

$$|\mathcal{F}| \leqq \sum_{X \in \mathcal{G}(\mathcal{F})} |X| - (a-1)|\mathcal{G}(\mathcal{F})|.$$

If $|\mathcal{G}(\mathcal{F})| = 1$, then $\sum_{X \in \mathcal{G}(\mathcal{F})} |X| \leqq b$ and $|\mathcal{F}| \leqq b-a+1$.

If $|\mathcal{G}(\mathcal{F})| = 2$, then $\sum_{X \in \mathcal{G}(\mathcal{F})} |X| \leqq r-1$ (since $\mathcal{G}(\mathcal{F})$ contains no complementary sets) and $|\mathcal{F}| \leqq r-1-2(a-1) = r-2a+1$.

If $|\mathcal{G}(\mathcal{F})| \geqq 3$, then $\sum_{X \in \mathcal{G}(\mathcal{F})} |X| \leqq r$ and

$$|\mathcal{F}| \leqq r-3(a-1) \leqq r-2a+1.$$

Hence,

$$|\mathcal{F}| \leqq \max(b-a+1, r-2a+1) = \begin{cases} b-a+1 & \textit{if} \quad a \geqq r-b, \\ r-2a+1 & \textit{if} \quad a < r-b. \end{cases}$$

Indeed,

$$\mathcal{D}_6(a,b) = \begin{cases} \mathcal{D}_3(0,a,b) & \textit{if} \quad a \geqq 2, a \geqq r-b, \\ \mathcal{D}_3(0,a,b) \cup \mathcal{D}_3(b,a,r-b-1) & \textit{if} \quad a \geqq 2, a < r-b \end{cases}$$

is an example which confirms that the upper bound is the desired result for $a \geqq 2$. (2) and $\mathcal{D}_1 \cup \mathcal{D}_6(1,b)$ yield the result for $a=1$.

5.3. Let $\mathbf{P} = \mathbf{A\bar{B}CD} \vee \mathbf{AB\bar{C}D} \vee \mathbf{ABC\bar{D}}$ (2.21).

5.3.1. $a \leqq r-b$.

If $\mathcal{F}$ is a maximal family, consider

$$\mathcal{F}' = \left\{ X : \begin{cases} X & \textit{if } X \in \mathcal{F}, |X| < r/2, \\ X & \textit{if } X \in \mathcal{F}, |X| = r/2, 1 \notin X, \\ R \setminus X & \textit{if } X \in \mathcal{F}, |X| = r/2, 1 \in X, \\ R \setminus X & \textit{if } X \in \mathcal{F}, |X| > r/2. \end{cases} \right\}.$$

Since $\mathcal{F}$ contains no complementary sets, $|\mathcal{F}'|=|\mathcal{F}|=n_{a,b}(\mathbf{P}, r)$. Obviously, $\mathcal{F}'$ satisfies $\mathbf{\bar{A}BCD}\vee\mathbf{A\bar{B}CD}\vee\mathbf{AB\bar{C}D}\vee\mathbf{ABC\bar{D}}$. No pair of $\mathcal{F}'$ satisfies $\mathbf{AB\bar{C}D}$. By Lemma 2 (v), we have now that $\mathcal{F}'$ satisfies $\mathbf{A\bar{B}CD}\vee\mathbf{ABC\bar{D}}$. Thus a maximal family of 2.20 is also a maximal family here. Hence, 2.21 follows by 2.20 if $a\leqq r-b$, $a=1$ or $a\geqq 2$.

5.3.2. $a>r-b$. Then $r-b<r-(r-a)$.

We apply Lemma 2 (ix) and the results of 5.3.1, and get

$$n_{a,b}(\mathbf{A\bar{B}CD}\vee\mathbf{AB\bar{C}D}\vee\mathbf{ABC\bar{D}}, r) =$$

$$= n_{r-b,r-a}(\mathbf{A\bar{B}CD}\vee\mathbf{ABC\bar{D}}, r) = \begin{cases} 2r-3 & \textit{if} \quad r-b=1, \\ r-2(r-b)+1 & \textit{if} \quad r-b\geqq 2. \end{cases}$$

5.4. Let $\mathbf{P}=\mathbf{A\bar{B}CD}\vee\mathbf{AB\bar{C}}\vee\mathbf{AB\bar{D}}$ (2.22).

If $\mathcal{F}$ is a maximal family, we split $\mathcal{F}$ into two subfamilies $\mathcal{F}_1$ and $\mathcal{F}_2$ by

if $X\in\mathcal{F}$, $R\setminus X\notin\mathcal{F}$, then $X\in\mathcal{F}_1$,

if $X\in\mathcal{F}$, $R\setminus X\in\mathcal{F}$, then $X\in\mathcal{F}_1$, $R\setminus X\in\mathcal{F}_2$ or $X\in\mathcal{F}_2$, $R\setminus X\in\mathcal{F}_1$.

Then $\mathcal{F}_1$ and $\mathcal{F}_2$, respectively, satisfy $\mathbf{A\bar{B}CD}\vee\mathbf{AB\bar{C}D}\vee\mathbf{ABC\bar{D}}$. Since $X\in\mathcal{F}$, $R\setminus X\in\mathcal{F}$ can hold only if $c\leqq|X|\leqq r-c$, $c=\max(a, r-b)$. We obtain immediately

$$|\mathcal{F}_1| \leqq n_{a,b}(\mathbf{A\bar{B}CD}\vee\mathbf{AB\bar{C}D}\vee\mathbf{ABC\bar{D}}, r)$$

and

$$|\mathcal{F}_2| \leqq n_{c,r-c}(\mathbf{A\bar{B}CD}\vee\mathbf{AB\bar{C}D}\vee\mathbf{ABC\bar{D}}, r).$$

Hence,

$$|\mathcal{F}| \leqq \begin{cases} r-2a+1+r-2(r-b)+1 & \textit{if} \quad a\leqq r-b, \\ 2b-r+1+r-2a+1 & \textit{if} \quad a>r-b, \end{cases}$$

i.e.

$$|\mathcal{F}| \leqq 2(b-a+1).$$

Similarly, it follows by (2)

$$|\mathcal{F}| \leqq \begin{cases} r+2b-2 & \textit{if} \quad a=1, \quad b\leqq r-2, \\ 3r-2a-2 & \textit{if} \quad a\geqq 2, \quad b=r-1, \\ 4r-6 & \textit{if} \quad a=1, \quad b=r-1. \end{cases}$$

Finally, we complete the proof by following examples

$\mathcal{D}_7(a, b)=\mathcal{D}_3(0, a, b)\cup\mathcal{D}_4(r-a, r-b)$	*if* $a\geqq 2$, $b\leqq r-2$,
$\mathcal{D}_1\cup\mathcal{D}_7(1, b)$	*if* $a=1$, $b\leqq r-2$,
$\mathcal{D}_7(a, r-1)\cup\mathcal{D}_2$	*if* $a\geqq 2$, $b=r-1$,
$\mathcal{D}_1\cup\mathcal{D}_7(1, r-1)\cup\mathcal{D}_2$	*if* $a=1$, $b=r-1$.

Remark. A family satisfying $\mathbf{A\bar{B}CD}\vee\mathbf{AB\bar{C}}\vee\mathbf{AB\bar{D}}$ may be interpreted as a family without qualitatively independent sets (see also KATONA [8]).

6. Proofs of 2.6 and 2.15

6.1. Let $\mathbf{P}=\mathbf{ABC}\vee\mathbf{ACD}$ (2.15) and let $a\leqq r-b$.

Let $\mathscr{F}$ be an arbitrary maximal family. Then $\mathscr{F}$ contains no complementary sets, i.e.

$$|\mathscr{F}|\leqq\sum_{i=a}^{r-b-1}\binom{r}{i}+\frac{1}{2}\sum_{i=r-b}^{b}\binom{r}{i}.$$

An example for a maximal family is

$$\{X\colon X\subseteqq R,\ a\leqq|X|<r/2\quad \textit{or}\quad (|X|=r/2\quad \textit{and}\quad 1\notin X)\}.$$

6.2. Let $\mathbf{P}=\mathbf{ABC}\vee\mathbf{ACD}\vee\mathbf{AB\bar{D}}$ (2.6) and let $a\leqq r-b$.

If $\mathscr{F}$ is a maximal family, then we split $\mathscr{F}$ into two subfamilies $\mathscr{F}_1$ and $\mathscr{F}_2$ by the same procedure as in Section 5, 4. Thus, $\mathscr{F}_1$ satisfies $\mathbf{ABC}\vee\mathbf{ACD}$, i.e.

$$|\mathscr{F}_1|\leqq n_{a,b}(\mathbf{ABC}\vee\mathbf{ACD},r).$$

$\mathscr{F}_2$ satisfies $\mathbf{ABC}\vee\mathbf{ACD}$. Moreover, for arbitrary sets $X,Y\in\mathscr{F}_2$ also $(R\setminus X,Y)$, $(X,R\setminus Y)$, and $(R\setminus X,R\setminus Y)$ satisfy $\mathbf{ABC}\vee\mathbf{ACD}=\mathbf{ABCD}\vee\mathbf{A\bar{B}CD}\vee\mathbf{ABC\bar{D}}$. Hence, (X,Y) satisfies $\mathbf{ABCD}\vee\mathbf{A\bar{B}CD}\vee\mathbf{ABC\bar{D}}$ as well as $\mathbf{ABCD}\vee\mathbf{\bar{A}BCD}\vee\mathbf{AB\bar{C}D}$, i.e. $\mathbf{ABCD}$. 2.15 implies

$$n_{a,b}(\mathbf{ABC}\vee\mathbf{ACD}\vee\mathbf{AB\bar{D}},r)=\sum_{i=a}^{\left[\frac{r-1}{2}\right]}\binom{r}{i}+\binom{r-1}{[r/2]-1}+\begin{cases}0 & \textit{if } r \textit{ is odd},\\ \binom{r-1}{r/2} & \textit{if } r \textit{ is even}.\end{cases}$$

Indeed, $\{X\colon X\subseteqq R,\ a\leqq|X|\leqq r/2$ *and, if r is odd,* $|X|=[r/2]+1,\ 1\notin X\}$ is a maximal family.

7. Open problems

In this section we give explicitely the open problems in usual notation. Also some estimations are presented.

1. *Problem* (2.1) $n_{a,b}(\mathbf{ACD},r)=?$

Remember that $\mathbf{ACD}$ means $(X\cap Y\neq\emptyset)\wedge(X\cup Y\neq R)$ for all $X,Y\in\mathscr{F}$. It is known only that

$$\sum_{i=a}^{b}\binom{r-2}{i-1}\leqq n_{a,b}(\mathbf{ACD},r)\leqq\begin{cases}\sum\limits_{i=a}^{b}\binom{r-1}{i} & \textit{if}\quad a\leqq r-b,\\ \sum\limits_{i=a}^{b}\binom{r-1}{i-1} & \textit{if}\quad a>r-b\end{cases}$$

by 2.15 and Lemma 2 (viii).

Equality occurs, for example, in the left hand side if $a=r-b=1$, and in the right hand side if $a=b=r/2$.

2. *Problem* (2.6, $a>r-b$) $n_{a,b}(\mathbf{ABC}\vee\mathbf{ACD}\vee\mathbf{AB\bar{D}},r)=?$ *if* $a>r-b$.

Remember that this condition means that $\mathcal{F}$ contains no not-complementary sets with union R. The investigations in the case $a \leqq r-b$ yield immediately

$$n_{a,b}(\mathbf{ABC} \vee \mathbf{ACD}, r) \leqq n_{a,b}(\mathbf{ABC} \vee \mathbf{ACD} \vee \mathbf{AB\overline{D}}, r)$$

$$\leqq n_{a,b}(\mathbf{ABC} \vee \mathbf{ACD}, r) + \binom{r-1}{[r/2]-1}.$$

3. *Problem* (2.23) $n_{a,b}(\mathbf{ABC} \vee \mathbf{AB\overline{C}\overline{D}}, r) = ?$

In this case also $n_{1,r-1}(\mathbf{P}, r)$ is unknown. Bounds are in analogy to [5] given by

$$n_{a,b}(\mathbf{ACD}, r) \leqq n_{a,b}(\mathbf{ACD} \vee \mathbf{AB\overline{C}\overline{D}}, r) \leqq n_{a,b}(\mathbf{ACD}, r) + \binom{r-1}{[r/2]-1}.$$

WILHELM-PIECK-UNIVERSITÄT
SEKTION MATHEMATIK
DDR—25 ROSTOCK
UNIVERSITÄTSPLATZ 1

References

[1] Clements, G. F. and H.-D. O. F. Gronau, On maximal antichains containing no set and its complement, *Discrete Math.*, v. 33, 1981, pp. 239—247.
[2] Daykin, D. E. and L. Lovász, The number of values of a Boolean function, *J. London Math. Soc.* (2), v. 12, 1976, pp. 225—230.
[3] Erdős, P., Chao Ko and R. Rado, Intersection theorems for systems of finite sets, *Quart. J. Math. Oxford Ser.* (2), v. 12, 1961, pp. 313—320.
[4] Greene, C., G. O. H. Katona and D. J. Kleitman, Extensions of the Erdős-Ko-Rado theorem, *Stud. Appl. Math.*, v. 55, 1976, pp. 1—8.
[5] Gronau, H.-D. O. F., On maximal families of subsets of a finite set, *Discrete Math.*, v. 34, 1981, pp. 119—130.
[6] Gronau, H.-D. O. F., Sperner type theorems and complexity of minimal disjunctive normal forms of monotone Boolean functions, *Period. Math. Hungar.*, v. 12, 1981, to appear.
[7] Hilton, A. J. W., Analogues of a theorem of Erdős-Ko-Rado on a family of finite sets, *Quart. J. Math. Oxford Ser.* (2), v. 25, 1974, pp. 19—28.
[8] Katona, G. O. H., Two applications of Sperner type theorems (for search theory and truth functions), *Period. Math. Hungar.*, v. 3, 1973, pp. 19—26.
[9] Lubell, D., A short proof of Sperner's lemma, *J. Combin, Theory Ser. A*, v. 1, 1966, pp. 299.
[10] Meshalkin, L. D., A generalization of Sperner's theorem on the number of subsets of a finite set, *Teor. Verojatnost. i Primenen.*, v. 8, 1963, pp. 219—220 (in Russian).
[11] Milner, E. C., A combinatorial theorem on systems of sets, *J. London Math. Soc.*, v. 43, 1968, pp. 204—206.
[12] Schönheim, J., On a problem of Purdy related to Sperner systems, *Canad. Math. Bull.*, v. 17, 1974, pp. 135—136.
[13] Sperner, E., Ein Satz über Untermengen einer endlichen Menge, *Math., Z*, v. 27, 1928, pp. 544—548.
[14] Yamamoto, K., Logarithmic order of free distributive lattices, *J. Math. Soc. Japan*, v. 6, 1954, pp. 343—353.

(Received June 11, 1980)

A note on the interconnection structure of cellular networks

By I. H. DEFEÉ

Using the concept of the structure automaton it is proved that every cellular automaton may be simulated by a cellular automaton realized by a cellular network of semigroup-type.

1. Introduction

The interconnection structures of infinite cellular automata, called also tesselation automata [3], are usually taken to be networks based on direct sum of infinite cyclic groups. Such networks have a great degree of uniformity [4]. Realizations of finite and infinite cellular automata by various types of uniform networks were described in [2]. It was shown there that such realizations may be described by the use of the theory of groups. The structure of cellular automata realized by nonuniform cellular networks has not been investigated because of the lack of the proper description method for such networks. In this paper a step in this direction is presented using the concept of the structure automaton. It is proved that every cellular automaton may be simulated by a cellular automaton realized by a cellular network of semigroup-type.

2. Preliminaries

Definition. *A cellular network* is a system $\mathcal{N}=(C, S, \delta, f)$ where C — is a count able *set of cells,* S — is a finite set of *cell-states,* $\delta\colon S^k \to S$ — is a *cell transition function,* $f\colon C \to C^k$ — is the *neighbourhood function.*

Definition. *The cellular automaton* (CA) realized by a cellular network $\mathcal{N}$ is a pair $\mathcal{A}(\mathcal{N})=(S^C, F)$ where $S^C=\{h|h\colon C\to S\}$ — is *the set of* CA *configurations* $F\colon S^C\to S^C$ is the *global map* defined by $\underset{c\in C}{\forall}\, F(h(c))=\delta\cdot h^k\cdot f(c)$ where $h^k(c_1, c_2, \ldots, c_k)=(h(c_1), h(c_2), \ldots, h(c_k))$.

Let $\mathcal{N}_1=(C_1, S_1, \delta_1, f_1)$ and $\mathcal{N}_2=(C_2, S_2, \delta_2, f_2)$ be two cellular networks with $f_1\colon C_1\to C_1^k$ and $f_2\colon C_2\to C_2^k$. A network $\mathcal{N}_1$ is a *realization* of the network $\mathcal{N}_2$ when there exists a pair of functions (φ, ψ), $\varphi\colon C_1\to C_2$, $\varphi(C_1)=C_2$, $\psi\colon S_1\to S_2$ such that $\underset{c_1\in C_1}{\forall}\ \varphi^k(f_1(c_1))=f_2(\varphi(c_1))$ and $\underset{s_1, s_2, \ldots s_k\in S_1}{\forall}\ \psi\cdot\delta_1(s_1, s_2, \ldots, s_k)=\delta_2(\psi(s_1), \psi(s_2), \ldots, \psi(s_k))$. These equalities mean that (φ, ψ) is a *homomorphism* of $\mathcal{N}_1$ onto $\mathcal{N}_2$.

In [2] the following theorem was proved.

Theorem 1. *If a cellular network $\mathcal{N}_1$ realizes the cellular network $\mathcal{N}_2$ then the cellular automaton $\mathcal{A}(\mathcal{N}_1)$ simulates the cellular automaton $\mathcal{A}(\mathcal{N}_2)$ i.e. a function H from $S_1^{C_1}$ onto $S_2^{C_2}$ exists such that*

$$\forall_{s^c \in S_1^{C_1}} H(F_1(s^c)) = F_2(H(s^c)).$$

Simulations of CA realized by various types of cellular networks having a high degree of uniformity were described in [2]. It was shown there that the simulation of CA realized by such networks is essentially a problem of the group homomorphism and, in some cases, a problem of permutation groups generators.

No attempts were reported on the simulation of CA realized by nonuniform networks. Particularly interesting question is whether there is an algebraic structure for the description similarily as the theory of groups in the case of uniform networks. The answer for this question is given here. It states that simulations of all CA may be described by the use of the theory of semigroups.

3. Results

Let $\mathcal{N}=(C, S, \delta, f)$ be a cellular network defined as above. For notational convenience we label cell inputs $1, 2, ..., k$ of the cells in $\mathcal{N}$ by $x_1, x_2, ..., x_k$.

Definition. *A structure automaton* of the cellular network $\mathcal{N}$ is a triple $\mathcal{N}_A = (X, C, \omega)$ where X — is the *imput alphabet of cell input labels, C* — is a countable *set of cells of $\mathcal{N}$ ω: $X \times C \to C$* is a *transition function* defined by $\forall_{x_i \in X} \forall_{c_k \in C} \omega(x_i, c_k) = c_l \Leftrightarrow$ the i-th component of the neighbourhood function value $f(c_l)$ is equal to c_k.

It is easy to see that every cellular network may be described by some structure automaton. Classical results [1] obtained in the theory of autcmata may be now applied to the description of cellular networks. For example we can generalize the classification of networks as follows: (For the notions below [1] may be consulted).

1. *Connected networks* described by connected structure automata.
2. *Strongly connected networks* described by strongly connected structure automata.
3. *Balanced networks* [2] described by connected permutation automata.
4. *Uniform networks* [2] described by quasi-perfect automata.
5. *Arrays* [2] described by perfect automata.

The enumeration above is done according to the generality of specific class of networks. The first two classes are important from the point of information flow in CA. In the cellular network described by the connected structure automaton there are some parts from (or to) which information flows in only one direction, and there are no such parts in the cellular network described by the strongly connected structure automaton.

Definition. *A cellular network $\mathcal{N}$* is of *semigroup-type* if there is ono-to-one correspondence α between the set of cells C and certain semigroup J with operation

$*$ such that for some subset $L=\{l_1, l_2, ..., l_k\}\subset J$,

$$\underset{c\in C}{\forall}\, \alpha^k(f(c)) = \big(l_1*\alpha(c), l_2*\alpha(c), ..., l_k*\alpha(c)\big).$$

Theorem 2. If $\mathcal{N}$ is a cellular network described by the strongly connected structure automaton $\mathcal{N}_A$ then there exist a cellular network $\mathcal{M}$ of semigroup-type such that the CA $\mathcal{A}(\mathcal{M})$ simulates the CA $\mathcal{A}(\mathcal{N})$.

Proof. By Theorem 1, it is sufficient to consider cellular network realizations. Let $\mathcal{N}_A=(X, C, \omega)$. With every imput symbol x_i we can associate a transformation $\omega_{x_i}\colon C\to C$ by taking $\omega(x_i, c)$ for all $c\in C$. Let J be the transformation semigroup generated by all ω_{x_i} for $x_i\in X$.

We define the following structure automaton $\mathcal{M}_J=(\Omega, J, m)$, where $\Omega=\{\omega_{x_i}|x_i\in X\}$ — input alphabet, $m(\omega_{x_i}, j)=\omega_{x_i}\cdot j$ — transition function defined as a composition of mappings in J.

Let $H\colon J\to C$ be a function defined by $\underset{j\in J}{\forall} H(j)=j(c_0)$ for some fixed $c_0\in C$. H is onto C because the semigroup J is transitive. We shall prove that H is a homomorphism of the structure automaton $\mathcal{M}_J(\Omega, J, m)$ onto the structure automaton $\mathcal{N}_A=(X, C, \omega)$. We have

$$\underset{\omega_{x_i}\in\Omega}{\forall}\ \underset{j\in J}{\forall}\ H\big(m(\omega_{x_i}, j)\big) = H(\omega_{x_i}\cdot j) = \omega_{x_i}\cdot j(c_0) = \omega\big(x_i, H(j)\big).$$

From Theorem 1 it follows that the CA realized by the semigroup-type network $\mathcal{M}$ described by the structure automaton $\mathcal{M}_J$ simulates the CA realized by the network $\mathcal{N}$. □

Now, we will extend Theorem 2 to cellular networks described by connected structure automata. In this case the transformation semigroup J is not transitive.

An *extension of the semigroup* J will be defined in two steps. First, when there is no identity, we add an identity e to the semigroup J obtaining the semigroup $J\cup e=J_e$. Let $C_g\subset C$ be the set (possibly with the smallest cardinality) such that

$$J_e(C_g) = C.$$

In the second step, let the elements of C_g be numbered $c_1, c_2, ..., c_i, ...$. For each element $c_i\in C_g$ a set of vectors is constructed

$$[J_e]_i = \{[0, 0, ..., 0, j_e, 0, ...]\} \quad \text{for all} \quad j_e\in J$$

where j_e is on the i-th position and 0 is an element such that $0\cdot 0=0\cdot j_e=j_e\cdot 0=0$.

Let $[J_e]=\bigcup\limits_i [J_e]_i$. It is easy to see that the set $[J_e]$ together with componentwise multiplication forms a semigroup.

Let H_e be a function $H_e\colon J_e\to C$ such that for each $c_i\in C_g$, $H([J_e]_i)$ is defined as

$$H([0, 0, ..., 0, j_e, 0, ...]) = j(c_i).$$

H is onto C because of the definition of the set C_g and vector semigroup $[J_e]$.

From these constructions we finally have

Theorem 3. Any cellular automaton may be simulated by a cellular automaton realized by a cellular network of semigroup-type.

4. Conclusion

It was provèd that semigroupe-type cellular networks are universal in the sense that any cellular automaton may be simulated by a network of such type. This result may compared with the simulation power of the group-type and Abelian group-type networks [2]. Note, that in the case of connected networks with infinite number of cells the semigroup for simulation may be not finitely generated, which gives a new level of complexity in the theory of cellular automata. Further investigation is needed in two directions. First, on the computational capability of the cellular automata realized by semigroup-type networks comparing to tesselation automata. Second, in the finite case, on the algebraic characterization of the structure of finite cellular automata using finite structure automata.

DEPARTMENT OF ELECTRICAL ENGNG.
TECHNICAL UNIVERSITY OF SZCZECIN
SIKORSKI AV. 37.
70-313 SZCZECIN, POLAND

References

[1] GÉCSEG, F., I. PEÁK, *Algebraic theory of automata*, Akadémiai Kiadó, Budapest, 1972.
[2] JUMP, J. R., J. S. KIRTANE, On the interconnection structure of cellular networks, *Inform. and Control*, v. 24, 1974, pp. 74—91.
[3] SMITH, A. R., Introduction to and survey of polyautomata theory, *Automata, Languages, Development*, A. LINDENMAYER, G. ROZENBERG eds., North Holland, 1976, pp. 405—422.
[4] WAGNER, E. G., On connecting modules together uniformly to form a modular computer, *IEEE Trans. Electr. Comput.*, EC—15, 1966, pp. 864—973.

(Received April 24, 1981)

Binary addition and multiplication in cellular space

By E. KATONA

Cellular automata are highly parallel bitprocessors, so they are suitable for the bitparallel execution of distinct computational tasks. In this paper powerful bitparallel algorithms are given for fixed point binary addition and multiplication, taking into account the cellprocessor architecture developed by T. LEGENDI [1]. For this architecture there have been constructed more then 100 cellular algorithms solving different computational tasks [6]. In a large cellular space a high number of cellular adders, multipliers and other processing elements may be embedded, and more complex tasks may be computed in parallel, as matrix multiplication [4], certain data processing tasks [5], etc.

1. Introduction

A cellular automaton is a highly parallel processor, but the economical programming of such a processor is not an easy task. If *macro-cells* are applied (a cell works as a microprocessor), then the programming of the cellular structure is somewhat easier [7], but the architecture has lower flexibility (fixed operations, fixed word length, etc.) and in general the bitparallel execution of the operations is impossible.

If *micro-cells* are applied (having maximum 16 states) with variable transition functions, then the cellprocessor has high flexibility and a totally bitparallel processing is possible. In [3], [4], [5], [6] and in this paper it is shown that a cellprocessor consisting of micro-cells is economically programmable, and the speed of the cellular algorithms is *wordlength-independent* in most cases.

The cellprocessor architecture proposed in [1] is based on the micro-cell conception, and has — from the point of view of this paper — the following characteristic properties:

(i) The cellular space is a *two-dimensional rectangle-form cell-matrix* which is bounded by *dummy-cells* (the dummy cells have no transition funtion, but their states can be set from the outside world). In the cellular net the *von Neumann neighbourhood* is assumed.

(ii) *The cells do not have a fixed transition function,* but receive commands (microinstructions) from a central control (CCPU), and arbitrary local transition function may be realized by the execution of a certain sequence of microinstructions. This implies that the cellprocessor can work with an *arbitrary local transition function,* and — moreover — it can work with *time-varying transition function.*

(iii) The cellular space is *inhomogeneous,* that is, the individual cells may work with different transition functions at the same time. To ensure this property, each cell has an *internal state.* The cells having different internal states may work with different transition functions. So, if there are n different internal states, then maximum n different transition functions may work in parallel. The internal states are set at $t=0$, and during the working of the cellprocessor they are unchanged.

The transition functions will be defined according to [2] by *microconfiguration terms.* A microconfiguration term has the form:

$$\boxed{\begin{array}{c}\text{the state of a group}\\ \text{of cells at time } t\end{array}} \rightarrow \boxed{\begin{array}{c}\text{the required state of (another)}\\ \text{group of cells at time } t+1\end{array}}$$

Each cell on the right side occurs on the left side, too, and is marked by double frame for the identification. Because of the inhomogeneity a microconfiguration term may describe more transition functions together.

The notation $[x_k x_{k-1} \dots x_1]$ will be used often in the text, which means a k-digit binary number having the digits $x_k, x_{k-1}, \dots, x_1$ $(x_i \in \{0, 1\})$.

2. Binary addition

Binary addition is the most fundamental arithmetic operation. The cellular algorithm described below is applied in many further cellular processing elements (see the cellular multiplier in this paper, and [4], [5], [6]).

The cellular binary addition is based on the "carry save" addition algorithm. Let $x=[x_k \dots x_1]$, $y=[y_k \dots y_1]$ and $z=[z_k \dots z_1]$ be binary numbers of k digits to be added. In the first step x and y are added *in a parallel way:* a (partial) sum $s=[s_k \dots s_1]$ and a carry vector $c=[c_k \dots c_1]$ is computed as follows

$$[c_i s_i] := x_i + y_i \qquad \text{for any } i. \tag{1}$$

In the second step the number z can be added to s and c by the formula

$$[c_i' s_i'] := z_i + s_i + c_{i-1} \qquad \text{for any } i. \tag{2}$$

(The sign $'$ serves for the distinction between the old and new values of s and c.)

If there are more numbers to be added, then they can be added to s and c also by formula (2). The complete sum of the operands should be computed from the last s and c in $k-1$ steps applying the formula

$$[c_i' s_i'] := s_i + c_{i-1} \qquad \text{for any } i. \tag{3}$$

On the basis of the described parallel addition algorithm it is easy to construct a cellular automaton for binary addition. It consists of k adder cells, each containing a sum bit S and a carry bit C (4-state cells). A dummy cell is connected to each adder cell as upper neighbour (Fig. 1).

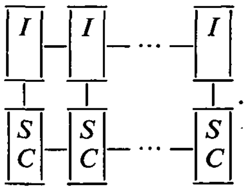
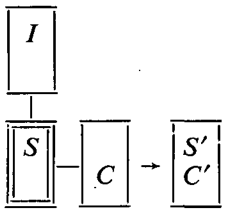


Fig. 1

At $t=0$ the bits S and C are 0, and the bits I contain the first number to be added. In any further step a new number will be written into the bits I and the adder cells work with the transition function:



where $[C'S'] = S+C+I$.

After the input of the last operand the dummy cells are set into 0 and after $k-1$ steps the complete sum of the operands is computed in the bits S of the cell-row. (In this way the above transition function includes the formulas (1), (2), (3).)

The addition of n numbers each consisting of k bits, needs $n+k-1$ *steps*, so the parallel addition algorithm is economical for many operands.

Remark. To prevent the overflow, for n operands a cellular adder consisting of $k+\log_2 n$ cells should be used. If only k cells are applied, then the leftmost cell needs a special overflow-watching transition function (inhomogeneity).

The above cellular adder has many simple applications, as the binary counter, the computation of certain number-rows (e.g. Fibonacci-numbers), vector addition, etc. [6]; but the most important application is the binary multiplication discussed in the next point.

3. The multiplication of two binary numbers

The cellular multiplication algorithm is based, as usual, on the addition: the partial products will be generated in a special cell-row, and another cell-row under it works as an adder (Fig. 2).

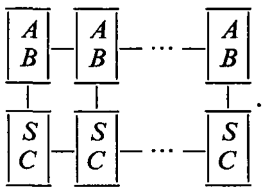


Fig. 2

The partial products are generated in an overlapped manner. Between the digits of the multiplicand $a=[a_k...a_1]$ and the multiplier $b=[b_k...b_1]$ zero digits are inserted, and in such a form they move step by step one against another in the upper cell-row (Fig. 3).

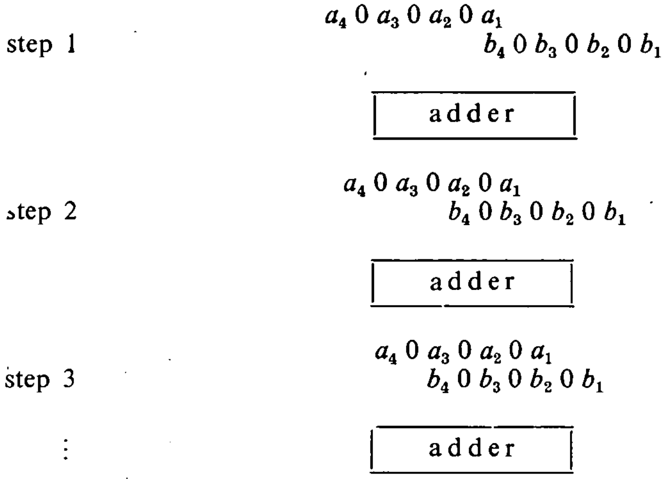
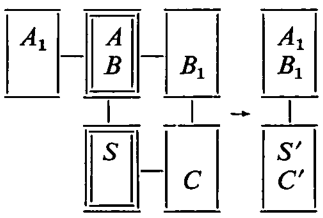

step 1
$a_4\ 0\ a_3\ 0\ a_2\ 0\ a_1$
$b_4\ 0\ b_3\ 0\ b_2\ 0\ b_1$
adder
step 2
$a_4\ 0\ a_3\ 0\ a_2\ 0\ a_1$
$b_4\ 0\ b_3\ 0\ b_2\ 0\ b_1$
adder
step 3
$a_4\ 0\ a_3\ 0\ a_2\ 0\ a_1$
$b_4\ 0\ b_3\ 0\ b_2\ 0\ b_1$
adder
⋮

Fig. 3
Cellular algorithm for binary multiplication in the case $k=4$.

The products of the operand digits staying on the same position are summed by the adder (on Fig. 3 in the first step a_1b_4, in the second step a_2b_4 and a_1b_3 are summed). Fig. 3 shows well that in steps 1, 2, 3 and 4 the bit b_4 is multiplied by a_1, a_2, a_3 and a_4, thus the partial product $[a_4a_3a_2a_1]\cdot b_4$ is generated for the adder. The partial products corresponding to b_3, b_2 and b_1 are computed in a similar way, and each is created on the appropriate position.

The two rows of the cellular multiplier have distinct transition functions, which may be defined together as follows:



where $[C'S'] = S+C+A\cdot B$.

If k-bit numbers are multiplied, then the product has $2k$ bits, therefore an adder of length $2k$ should be used. Thus the multiplier needs $4k$ *4-state cells.*

If at $t=0$ the configuration of Fig. 3 (step 1) is assumed, then at $t=2k-1$ all the partial products are generated. It is easy to see that at $t=2k$ the rightmost k cells of the adder have zero carry bits. Therefore to compute the complete product further k steps are needed, thus the whole multiplication process uses $3k$ steps.

Remark. If between the digits of a and b the digits of further two k-bit numbers x and y are written (instead of the zeros), then the multiplier computes the expression $a \cdot b + x \cdot y$! The cellular multiplier may be used for vector-multiplication in a similar way [4].

4. Multiplication of more then two numbers

In this section a cellular algorithm is given to compute the product $x_1 \dots x_n$ where x_i is a k-bit number and $0 \leqq x_i < 1$ holds for any i (the leftmost digit of x_i has the positional value 2^{-1}). To solve this task the cellular multiplier of section 3 will be modified: 3-bit cells (i.e. 8-state cells) will be used where the third bits in the adder cells serve for control (Fig. 4).

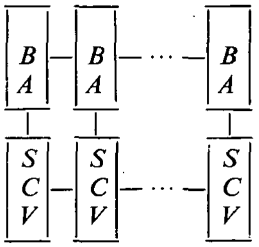
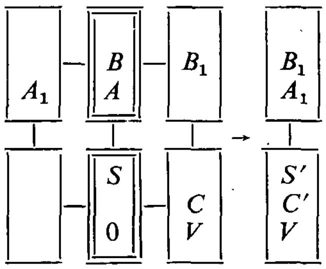


Fig. 4
Cellular multiplier for more then two numbers. The control bits are marked by V.

At $t=0$ the number x_1 is stored in the bits "S" of the adder. The numbers $x_2, \dots, x_n$ come from the outside world and go left on the bits "B". Before each number x_i a control signal of value 1 is sent, which goes left on the control bits and copies the bits "S" into the bits "A" (at the same time the adder is cleared). Thus the number x_i coming from the outside world is multiplied by the product $x_1 \cdot \dots \cdot x_{i-1}$, and the process may be repeated until it is necessary.

According to the above principle, the transition functions of section 3 should be modified as follows.

If the adder cell contains a control signal 0:

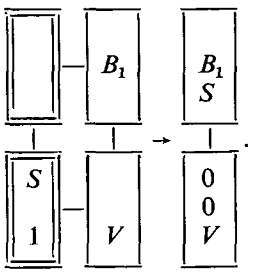


where $[C' S'] = S + C + A \cdot B$.

Listing

```
STEP  0:  . . . . . . . .      STEP 12:  1 . 0 . 1 . . .
          . . . . . . . .                . . . . 1 . 0 .
          1 0 0 1 0 0 0 0                0 1 1 0 1 1 0 0
          . . . . . . . <                . . . . . < . .

STEP  1:  . . . . . . . .      STEP 13:  . 0 . 1 . . . 1
          . . . . . . . 0                . . . . . 1 . 0
          1 0 0 1 0 0 0 0                0 1 1 0 2 0 0 0
          . . . . . . < .                . . . . < . . .

STEP  2:  . . . . . . . .      STEP 14:  0 . 1 . . . 1 .
          . . . . . . 0 .                . . . . 0 . 1 .
          1 0 0 1 0 0 0 0                0 1 1 1 0 0 0 0
          . . . . . < . .                . . . < . . . .

STEP  3:  . . . . . . . 1      STEP 15:  . 1 . . . 1 . 1
          . . . . . 0 . 0                . . . 1 . 0 . 1
          1 0 0 1 0 0 0 0                0 1 1 0 0 0 1 0
          . . . . < . . .                . . < . . . . .

STEP  4:  . . . . . . 1 .      STEP 16;  1 . . . 1 . 1 .
          . . . . 0 . 0 .                . . 1 . 1 . 0 .
          1 0 0 1 0 0 0 0                0 1 0 0 0 0 1 1
          . . . < . . . .                . < . . . . . .

STEP  5:  . . . . . 1 . 1      STEP 17:  . . . 1 . 1 . 1
          . . . 1 . 0 . 0                . 1 . 1 . 1 . 0
          1 0 0 0 0 0 0 0                0 0 0 0 1 0 1 1
          . . < . . . . .                < . . . . . . .

STEP  6:  . . . . 1 . 1 .      STEP 18:  . . 1 . 1 . 1 .
          . . 0 . 1 . 0 .                0 . 1 . 1 . 1 .
          1 0 0 0 0 0 0 0                0 0 0 1 1 1 1 1
          . < . . . . . .                . . . . . . . .

STEP  7:  . . . 1 . 1 . 0      STEP 19:  . 1 . 1 . 1 . 0
          . 0 . 0 . 1 . 0                . 0 . 1 . 1 . 1
          1 0 0 0 1 0 0 0                0 0 1 1 2 1 2 1
          < . . . . . . .                . . . . . . . .

STEP  8:  . . 1 . 1 . 0 .      STEP 20:  1 . 1 . 1 . 0 .
          1 . 0 . 0 . 1 .                . . 0 . 1 . 1 .
          0 0 0 0 1 1 0 0                0 0 1 3 0 3 0 1
          . . . . . . . .                . . . . . . . .

STEP  9:  . 1 . 1 . 0 . 1      STEP 21:  . 1 . 1 . 0 . .
          . 1 . 0 . 0 . 1                . . . 0 . 1 . 1
          0 0 0 0 1 1 0 0                0 0 2 1 2 1 0 1
          . . . . . . . .                . . . . . . . .

STEP 10:  1 . 1 . 0 . 1 .      STEP 22:  1 . 1 . 0 . . .
          . . 1 . 0 . 0 .                . . . . 0 . 1 .
          0 1 0 0 1 1 0 1                0 1 0 2 0 1 0 1
          . . . . . . . <                . . . . . . . .

STEP 11:  . 1 . 0 . 1 . .      STEP 23:  . 1 . 0 . . . .
          . . . 1 . 0 . 1                . . . . . 0 . 1
          0 1 1 0 1 1 0 0                0 1 1 0 0 1 0 1
          . . . . . . < .                . . . . . . . .
```

If the adder cell contains a control signal 1:



The multiplication process is demonstrated on a simulation example (see Listing). The product of $x_1=0.1001$, $x_2=0.1101$ and $x_3=0.1110$ will be computed by an 8-bit multiplier. The multiplier is displayed in 4 rows, according to Fig. 4, but in the third row the bits S and C are printed together in the form $[CS]$ (that is, for example the value 2 means $C=1$ and $S=0$). The points meán insignificant zeros in each row.

At $t=0$, x_1 is stored in the adder, and a control signal marked by "<" starts on the right end of the multiplier. Between $t=1$ and $t=8$ the number x_1 is copied into the bits "A" and it is shifted right (hereby zeros are inserted between the digits). The number x_2 comes from outside and will be multiplied by x_1. At $t=10$ the rightmost digit of x_1x_2 is computed. Already at this moment a new control signal may be started which ensures the multiplication of x_1x_2 by x_3, thus an overlapping is possible between the consecutive multiplications.

For the multiplication of n numbers $(2k+2)(n-1)+2k \approx 2kn$ *steps* are required, and the modified multiplier consists of $4k$ *8-state cells*. The product contains $2k$ digits (the leftmost digit has the positional value 2^{-1}) and the first $?k-\log_2 k-\log_2 n$ bits are always correct.

5. Concluding remarks

In this paper three fundamental *cellular processing elements* have been discussed, *each designed for the same cellprocessor architecture* [1]. Each processing element is based on a *bitparallel cellular algorithm* where nearly all cells work effectively in each time-step. By the interconnection of such simple processing elements more complex tasks may be solved in bitparallel by a cellprocessor.

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

References

]1] Legendi, T., Cellprocessors in computer architecture, *Computational Linguistics and Computer Languages*, v. 11, 1977, pp. 147—167.

[2] Legendi, T., A 2D transition function definition language for a subsystem of the CELLAS cellular processor simulation language, *Computational Linguistics and Computer Languages*, v. 13, 1979, pp. 169—194.

[3] Katona, E., T. Legendi, Cellular algorithms for fixed point decimal addition and muítiplication, *Elektron. Informationsverarb. Kybernet.*, v. 17, 1981, pp. 637—644.

[4] Katona, E., Cellular algorithms for fixed point vector- and matrix-multiplication, *Proceedings of the Conference Programming Systems*' 81, pp. 262—280, in Hungarian.

[5] Katona, E., The application of cellprocessors in conventional data processing, *Proceedings of the Third Hungarian Computer Science Conference,* Publishing House of the Hungarian Academy of Sciences, Budapest, 1981, pp. 295—306.

[6] Katona, E., Cellular algorithms (Selected results of the cellprocessor team led by T. Legendi), Von Neumann Society, Budapest, 160 pages in Hungarian, 1981

[7] Domán, A., A 3-dimensional cellular space, *Sejtautomaták,* Gondolat Kiadó, Budapest, 1978, in Hungarian.

[8] Vollmar, R., *Algorithmen in Zellularautomaten,* B. G. Teubner, Stuttgart, 1979.

[9] Nishio, H., Real time sorting of binary numbers by 1-dimensional cellular automaton, *Proceedings of the International Symposium on Uniformly Structured Automata and Logic,* Tokyo, 1975, pp. 153—162.

(Received April 10, 1981)

Алгебраический подход к операциям на элементах базы данных

С. Лебедева

База данных представлена в качестве формализированной системы; представлено постулаты этой системы, подано определения операций на данных и примеры применения этих операций. Показано практическое применение представленной системы для конкретной задачи: многостепенной идентификации объекта. Указывается связь между алгебраическими операциями на данных и инструкциями Языка Манипулирования Данными. Операции на данных сравниваются с операциями на отношениях в реляционных базах данных.

1. Введение

Во время разработки базы данных для многостепенного эксперимента [1] появилась необходимость формализации некоторых проблем, связанных с базой данных и манипулированием данными. Бава данных для многостепенного эксперимента и способ её использования имеют ряд особенностей. Вопервых, база данных имеет численный характер: элементами базы данных являются двумерные матрицы, векторы, системы векторов и отдельные числа. Во-вторых, элементами базы данных являются регулярные структуры [5] или же структуры, полученные при помощи операций на данных из регулярных структур. В-третьих, в процессе эксперимента экспериментатор может принять решение об увеличении числа измерений, или же может появитъся необходимость' использования измерений, полученных в других лабораториях, что повлечёт за собой не только увеличение количества данных, но также необходимость объединения данных, находящихся в разных физических областях.

Полученные результаты имеют довольно общий характер и справедливы для любых баз данных с иерархическими и сетевыми структурами.

2. Язык. Первичные понятия и постулаты

Обозначим символом N счётное множество *имён* данных базы данных. Элементы множества имён будем обозначать символами $n_1, n_2, n_3, \ldots$, элементы множества значений-символами $v, v_1, v_2, \ldots$. В множестве V выделим некоторое подмножество V_S *структурных значений*. Предполагаем, что на множестве N определено отображение f, значения которого принадлежат множеству V. Множество всех упорядоченных пар (n, v) выполняющих условие $n \in N, v \in V$ и $v = f(n)$ будем называть *данными*, множество всех данных обозначим символом D. Элементы множества D будем обозначасть символами $d, d_1, d_2, \ldots$. Символами $\cdot, /, \#$ обозначим некоторые операции, определённые на множестве имён. Операции $\cdot, /, \#$ позволяют получать новые имена из имён, принадлежащих множеству N. Предполагаем, что на множестве имён определено отношение частичного упорядочения $<$. Символами $\sim$, $\wedge$ и $\Rightarrow$ будем обозначать логические связки: негацию, конъюнкцию и импликацию соответственно, символ $\forall$ обозначает квантор общности, символ $\exists$ — квантор существования. Угловые скобки $\langle$ и $\rangle$ обозначают последовательность данных. Свойства базы данных описывают постулаты P1—P14 [3].

P1 $\forall (n \in N) \exists (v \in V) \{f(n) = v\}$

P2 $\forall (n_1, n_2) \quad \{n_1 = n_2 \Rightarrow f(n_1) = f(n_2)\}$

P3 $\forall (n_1, n_2 \in N) \quad \{n_1 \cdot n_2 \in N\}$

P4 $(n_1 \cdot n_2) \cdot n_3 = n_1 \cdot (n_2 \cdot n_3)$

P5 $\forall (n_1, n_2 \in N) \quad \{n_1 \# n_2 \in N\}$

P6 $\forall (n_1, n_2 \in N) \quad \{n_1 / n_2 \in N\}$

P7 $f(n_1 \cdot n_2) = \langle f(n_1), f(n_2) \rangle$

P8 $f(n_1 \# n_2) = \langle (n_1, f(n_1)), (n_2, f(n_2)) \rangle$

P9 $(n, v) \in D \Leftrightarrow n \in N \wedge v \in V \wedge f(n) = v$

P10 $d \in V_S$

P11 $v_1, v_2 \in V_S \Rightarrow \langle v_1, v_2 \rangle \in V_S$

P12 $d = (n, v) \wedge v = \langle d_1, \ldots, d_k \rangle \wedge d_i \in v \Rightarrow d_i < d$

P13 $d_1 < d_2 \wedge d_2 < d_3 \Rightarrow d_1 < d_3$

P14 $\sim (d < d)$

В силу постулатов P1—P2 на множестве имён определено отображение f значения которого принадлежат множеству значений V. Постулаты P3—P6 определяют некоторые отношения на именах. Операцию $\cdot$ будем называть *соединением имён*, операцию $/$ — *вычитанием имён*, операцию $\#$ — *конструкцией имён*.

Постулаты P7 и P8 определяют отображение f для соединения и конструкции имён соответственно. Постулат P9 является необходимым и достаточным условием принадлежности данного множеству данных базы данных. Постулаты P10—P11 описывают свойства множества структурных значений, из постулатов P12—P14 следует, что на элементах данных базы данных определено отношение частичного упорядочения $<$. Отношение $<$ будем называть отношением предшествования. Из постулатов P9, P1 и P2 сейчас же следует.

Следствие 2.1. $d_1 = (n_1, v_1) \wedge d_2 = (n_2, v_2) \Rightarrow (n_1 = n_2 \Rightarrow v_1 = v_2)$.

Следствие 2.1 гарантирует, что в базе данных нет двух данных, имена которых идентичны, а значения разные.

Данные, значения которых принадлежат множеству структурных значений V_S, будем называть сложными данными, данные, значения которых принадлежат множеству V/V_S — данными элементарными. Множество сложных данных обозначим символом D_S, множество элементарных данных — символом D_E.

3. Операции на элементах базы данных

Основными операциями на элементах базы данных являются следующие операции: операция β *извлечения значения данного*, операция δ *экстракции или извлечения данного из некоторого подмножества данных* операция $\hat{\#}$ *конструкции данных* [5], операция $\odot$ *конкатенации* (соединения данных), операция ζ *ограниченного вычитания данных*, операция γ *изменения значения данного* [3, 4].

Определение 3.1. Пусть $d=(n, v)$ — данное, где n — имя, а v — значение данного d. Тогда

$$\beta(n, v) = v.$$

Операция извлечения значения данного β ставит в соответствие каждому данному его значение. Выполнимость операции извлечения значения гарантирует постулат P1, однозначность операции следует из следствия 2.1. Заметим, что результат операции извлечения значения, вообще говоря, не данное, а элементарное значение или последовательность данных. Например, допустим, что данные $d_1=$(измерение 3,1.05) и $d_2=$(координаты точки a, $\langle$(координата x, 3.5), (координата y, 5.0)$\rangle$). Тогда $\beta(d_1)=1{,}5$; $\beta(d_2)=\langle$(координата x, 3.5), (координата y, 5,0)$\rangle$.

Операция δ экстранции или извлечения данного из некоторого подмножества данных базы данных позволяет получить данное $d=(n, v)$, если нам известно имя этого данного и имя подмножества данных, элементом которого является данное d. Операция δ зависит от двух аргументов: первым аргументом является имя n данного d, вторым аргументом — имя подмножества X.

Определение 3.2. Пусть n — имя данного $d=(n, v)$, X — имя подмножества данных базы данных, которому принадлежит данное d. Тогда

$$\delta(n, X) = (n, v).$$

Операция δ определена на основании постулата P1, однозначность операции следует из следствия 2.1. Пусть X будет именем множества векторов $в_1=$ =(век 1, $\langle(x_1, 1), (y_1, 2)\rangle$), $в_2=$(век 2, $\langle(x_2, 3), (y_2, 1.5)\rangle$), $в_3=$(век 3, $\langle(x_3, 5), (y_3, 2.5)\rangle$), имена этих векторов соответственно век 1, век 2, век 3, значениями являются последовательности данных $\langle(x_1, 1), (y_1, 2)\rangle$, $\langle(x_2, 3), (y_2, 1.5)\rangle$, $\langle(x_3, 5), (y_3, 2.5)\rangle$ где x_i и y_i-имена компонент вектора век$_i$, $i=1, 2, 3$. Тогда

δ (век 1, X)=(век 1, $\langle(x_1, 1), (y_1, 2)\rangle$)
δ (век 2, X)=(век 2, $\langle(x_2, 3), (y_2, 1.5)\rangle$)
δ (век 3, X)=(век 3, $\langle(x_3, 5), (y_3, 2.5)\rangle$).

Суперпозиция операций экстракции и извлечения значения данного даёт возможность получить значение любого данного из любого множества данных. Для описанного выше примера мы получаем значения данных принадлежащих множеству данных с именем X:

$$\beta(\delta(\text{век } 1, X)) = \langle(x_1, 1), (y_1, 2)\rangle$$
$$\beta(\delta(\text{век } 2, X)) = \langle(x_2, 3), (y_2, 1.5)\rangle$$
$$\beta(\delta(\text{век } 3, X)) = \langle(x_3, 5), (y_3, 2.5)\rangle$$
$$\beta(x_1, 1) = 1,\ \beta(x_1, 2) = 2,\ \beta(x_2, 3) = 3,\ \beta(y_2, 1.5) = 1.5 \text{ и т. д.}$$

Определим операции, позволяющие создавать новые данные из элементов множества данных D: операцию $\odot$ соединения данных (конкатенации), операцию $\underset{\smile}{\overset{\frown}{\#}}$ конструкции данных и операцию $\wr$ — ограниченного вычитания данных.

Определение 3.3. Если $d_1=(n_1, v_1)$ и $d_2=(n_2, v_2)$, то

$$d_1 \odot d_2 = \begin{cases} (n_1 \cdot n_2, \langle v_1, v_2\rangle) & \text{для } d_1, d_2 \in D_S \\ (n_1 \cdot n_2, \langle d_1, d_2\rangle) & \text{для } d_1, d_2 \in D_E \\ (n_1 \cdot n_2, \langle v_1, d_2\rangle) & \text{для } d_1 \in D_S,\ d_2 \in D_E \\ (n_1 \cdot n_2, \langle d_1, v_2\rangle) & \text{для } d_1 \in D_E, d_2 \in D_S. \end{cases}$$

Выполнимость операции $\odot$ гарантируют постулаты P3, P7, P10 и P11, однозначность операции следует из следствия 2.1. Из постулатов P7—P10 следует, что результатом операции $\odot$ является данное. Операция $\odot$ ассоциативна.

Приведём примеры операции соединения данных. Пусть данные $d_1 = (x, 2)$ и $d_2=(y, 3)$ будут элементарными данными. В результате соединения данных d_1 и d_2 мы получим некоторый вектор $d=(x \cdot y, \langle(x, 2), (y, 3)\rangle)$, где x и y — имена, а 2 и 3 — значения компонент этого вектора. Пусть данные (серия 1, $\langle$(измерение 1, $\langle(x_1, 1), (y_1, 3), (z_1, 5)\rangle$), (измерение 2, $\langle(x_2, 1), (y_2, 5), (z_2, 8)\rangle$)$\rangle$) и (серия 2, $\langle$(измерение 3, $\langle(x_3, 2), (y_3, 4), (z_3, 8)\rangle$), (измерение 4, $\langle(x_4, 3), (y_4, 1), (z_4, 7)\rangle$), (измерение 5, $\langle(x_5, 4), (y_5, 8), (z_5, 1)\rangle$)$\rangle$) — двумерные матрицы, число строк первой матрицы — два, второй — три. Назовём результат соединения имён «серия 1» $\cdot$ «серия 2» именем «серия измерений», т. е. «серия 1» и «серия 2»=«серия измерений». Тогда в результате конкатенации мы получим данное (серия измерений,

$$\begin{aligned}&\langle(\text{измерение } 1, \langle(x_1, 1), (y_1, 3), (z_1, 5)\rangle),\\ &(\text{измерение } 2, \langle(x_2, 1), (y_2, 5), (z_2, 8)\rangle),\\ &(\text{измерение } 3, \langle(x_3, 2), (y_3, 4), (z_3, 8)\rangle),\\ &(\text{измерение } 4, \langle(x_4, 3), (y_4, 1), (z_4, 7)\rangle),\\ &(\text{измерение } 5, \langle(x_5, 4), (y_5, 8), (z_5, 1)\rangle)\rangle),\end{aligned}$$

которое представляет собой матрицу, состоящую из пяти строк.

Определение 3.4. Если $d_1=(n_1, v_1)$ и $d_2=(n_2, v_2)$, то

$$d_1 \underset{\smile}{\overset{\frown}{\#}} d_2 = (n_1 \# n_2, \langle d_1, d_2\rangle).$$

Выполнимость операции $\overset{\frown}{\underset{\smile}{\#}}$ гарантируют постулаты P5, P10 и P11, однозначность — следствие 2.1. Предположим, что в базе данных хранятся матрицы являющиеся входными и выходными измерениями некоторого эксперимента. Имена этих матриц соответственно — «входные измерения» и «выходные измерения». В результате операции конструкции данных мы получим данное, имя которого — «входные измерения # выходные измерения», а значение — последовательность матриц входных и выходных измерений. Многократное применение операции конкатенации и конструкции к элементам базы данных делает возможным создание новых множеств данных. Множество данных можно интерпретировать как данное, значением которого является последовательность элементов этого множества. Операция конструкции сохраняет отношение предшествования.

Операцией обратной относительно операции конструкции является операция ограниченного вычитания $\wr$. Перед определением этой операции введём понятие отношения непосредственного предшествования $\overset{*}{<}$.

Определение 3.5. $d_1 \overset{*}{<} d_2 \Leftrightarrow d_1 \overset{*}{<} d_2 \wedge \sim \exists (d)\{d_1 < d \wedge d < d_2\}$ например, если $d = (n, \langle d_1, d_2 \rangle)$, то $d_1 \overset{*}{<} d$ и $d_2 \overset{*}{<} d$.

Теперь можно определить операцию ограниченного вычитания данных:

Определение 3.6. Если $d = (n, v), v = \langle d_1, d_2, \ldots, d_n \rangle$ и $d_i = (n_i, v_i), i = 1, 2, \ldots, n$, то

$$d \wr d_i = (n/n_i, \langle d_1, d_2, \ldots, d_{i-1}, d_{i+1}, \ldots, d_n \rangle).$$

Заметим, что операция $\wr$ определена не для всех пар данных, а только для таких пар d_k, d_i, для которых выполнено условие $d_i \overset{*}{<} d_k$. Выполнимость и однозначность операции гарантируют постулаты P1, P2, P7, P11 и P9. Покажем действие операции $\wr$ на данных из предыдущего примера. Пусть именами данных d, d_1 и d_2 будут соответственно имена «входные измерения # выходные измерения», «входные измерения», «выходные измерения». Тогда результаты операции $d \wr d_2$ будет данное d_1, имя которого «входные измерения» а значение — матрица значений входных измерений. Операция $\wr$ является избыточной операцией, данное d_1, может быть получено из данного d при помощи операции экстракции. Вообще, каждое данное полученное при помощи операции ограниченного вычитания может быть получено при помощи суперпозиции операции экстракции, конструкции и конкатенации. Эта операция введена только для удобства пользователя. Операция γ изменения значения данного присваивает данному $d = (n, v)$ новое значение v_1.

Определение 3.7. Если $(n, v) \in D$ и $v, \in V$ то

$$\gamma((n, v), v_1) = (n, v_1).$$

Операция изменения значения может изменить структуру базы данных. Чтобы этого избежать, можно потребовать, чтобы старое и новое значения были одинаковы в структурном отношении, в случае, когда оба значения определены. Постулаты P1—P14 не предусматривают случая, когда значение данного не определено. Но такая ситуация может иметь место. Например, пользователь вводит в базу данных имена матриц, информацию относительно числа строк и столбцов. Система управления базой данных резервирует место для записи значений. До момента введения данного в базу данных, значение данного не определено. Чтобы учесть эту ситуацию допустим существование выделенного элемента Φ множества значений V.

P15 $\Phi \in V$.

Если $f(n) = \Phi$, то значение данного не определено. Результат операций конкатенации и конструкции для случая, когда значение одного из аргументов не определено, определяют формулы (3.1)—(3.2)

$$(n_1, v_1) \odot (n_2, \Phi) = (n_2, \Phi) \odot (n_1, v_1) = (n_1, v_1) \tag{3.1}$$

$$(n_1, v_1) \overset{\frown}{\underset{\smile}{\#}} (n_2, \Phi) = (n_2, \Phi) \overset{\frown}{\underset{\smile}{\#}} (n_1, v_1) = (n_1, v_1) \tag{3.2}$$

Результатом конкатенации или конструкции любого данного d с данным, значение которого не определено, является данное d, следовательно данное, значение которого не определено является нейтральным элементом относительно операций конкатенации и конструкции. Частным случаем операции изменения значения является операция *присваивания значения* данным γ^*. Операция γ^* определяется формулой

$$\gamma^*((n, \Phi), v) = (n, v). \tag{3.3}$$

В результате операции присваивания значения значение данного становится определённым.

4. Практическое применение

Система базы данных, удовлетворяющая постулатам P1—P15 была разработана для специальной задачи: многостепенного эксперимента (многостепенной идентификации объекта). Отличительной чертой базы данных для многостепенного эксперимента является её динамический характер: число данных в базе данных постоянно возрастает [1, 2]. Экспериментатору может понадобиться матрица измерений, проводимых в разное время и записанных в разных физических областях, эту потребность удовлетворяют операции конкатенации и конструкции. Экспериментатору может понадобиться только часть матрицы входных (или выходных) измерений, это требование выполняется при помощи операции ограниченного вычитания или суперпозиции операций экстракций и конструкции. Заметим, что операция ограниченного вычитания была введена только для удобства пользователя. Система Управления Базой Данных (СУБД) резервирует место для данных пользователя и содержит процедуры, являющиеся реализациями перечисленных операций.

Доступ до базы данных и операции на данных и множествах данных реализует Язык Манипулирования Данными (ЯМД). Инструкции ЯМД можно поделить на две группы: инструкции типа WRITE осуществляющие запись информации в базу данных и инструкции типа READ, осуществляющие введение информации, находящейся в базе данных, в оперативную память [4]. Инструкции типа WRITE являются реализациями операций присваивания значения данным и изменения значения данных. Инструкции типа READ реализуют операции извлечения значения данного, суперпозицию операций экстракции и извлечение значения, операции конкатенации, конструкции, ограниченного вычитания и суперпозиции этих операций. Описанные операции гарантируют независимость данных. Пользователь должен знать только имя данного и имя множества, которому принадлежит это данное, физическая организация базы данных и физические адреса данных пользователю неизвестны.

Описанная система базы данных является действующей системой, она внедрена в Лаборатории Технической Механики Вроцлавского политехнического института на ЭВМ ODRA-1325.

5. Заключительные замечания

Представленные операции не исчерпывают всех операций, которые можно производить на элементах базы данных. Важный класс операций составляют операции выборки данных, выполняющие некоторое логическое условие, например, нужно выбрать все измерения, значения которых находятся в определённом интервале и т. п. Логические условия обычно зависят от конкретного применения. В настоящем сообщении операции, реализующие выборку данных по заданному критерию, не обсуждаются.

В заключение несколько слов о сравнении представленной модели базы данных с реляционной моделью Кодда. Известно, что базы данных с древовидными и простыми сетевыми структурами можно преобразовать в реляционную базу данных (с некоторой избыточностью). Заметим, что при таком преобразовании в общем случае не сохраняется отношение порядка. Нетрудно заметить, что существует и обратный переход от реляционной модели к сетевой структуре. Действительно, элементами реляционной базы данных являются двумерные таблицы, поэтому реляционные базы данных выполняют постулаты P1—P14 при интерпретации отношения (таблицы) и строки таблицы как сложного данного и элемента таблицы (A, g) где A — имя атрибута, g — конкретное значение атрибута) как данного элементарного. Основные операции реляционной базы данных — *объединение отношений* и *проекция* выполнимы в представленной модели. Операция объединения отношений выполняется при помощи операции конкатенации, операция проекции — при помощи экстракции (если нам нужен один столбец отношения) или же при помощи суперпозиции операций экстракции, конструкции и конкатенации.

Представленные постулаты непротиворечивы, существует простая интерпретация этих постулатов в теории множеств. Представленная модель базы данных однозначна (результаты операций однозначны), полна в том смысле, что при помощи конечного числа операций может быть получено любое элементарное или структурное данное, а также замкнута. Замкнутость состоит в том, что результаты всех операций на элементах базы данных принадлежат базе данных.

ВРОЦЛАВСКИЙ ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ
ИНСТИТУТ ТЕХНИЧЕСКОЙ КИБЕРНЕТИКИ
ЛАБОРАТОРИЯ СИСТЕМ УПРАВЛЕНИЯ
ПОЛЬЩА

Литература

[1] Bubnicki, Z., On the multistage identification, *Systems Sci.*, v. 3, No 2, 1977.
[2] Lebiediewa, S., Rola bazy danych w procesie sterowania eksperymentem wielostopniowym, Prace Naukowe ICT Politechniki Wrocławskiej, Studia i Materiały, NR 46, 1978.
[3] Лебедева, С., Аксиоматический подход и операциям на элементах базы данных, 2 Symposium Grundlagen und Anwendung der Informationsverarbeitung, Technische Hochschule Karl-Marx-Stadt, Tagungsberichte, 1978.
[4] Lebiediewa, S., Rola bazy danych i Języka Manipulacji Danymi w procesie sterowania eksperymentem wielostopniowym, ICT Politechniki Wrocławskiej, Raport 279, Wrocław, 1978.
[5] Turski, W., Struktury danych, WNT, Warszawa, 1971.

(Поступило 3-ого декабря 1981 г.)

Описание одного класса предельных распределений в одноканальных приоритетных системах

Э. А. Даниелян

1°. Бурное развитие вычислительной техники предъявляет к современной теории массового обслуживания новые требования. Дело в том, что математические модели прохождения программ на ЭВМ являются грубыми приближениями и поэтому не могут целиком опысывать реальные процессы, возникающие при обслуживании вычислительной техники. К тому же, точные результаты, получаемые даже для простых систем, порой настолько сложны, что часто малопригодны для практических применений.

В теории приоритетных систем почти все точные результаты получаются в терминах преобразований Лапласа—Стилтьеса (ПЛС). Однако на практике удобнее оперировать их обращениями, получение которых представляет собой трудную задачу.

Настоящая работа посвящена обращению точных формул для совместного предельного распределения времен ожидания в следующей приоритетной модели.

2°. В одноканальную систему массового обслуживания с ожиданием поступают независимые пуассоновские потоки 1-вызовов, ..., r-вызовов. При фиксированных функциях распределения (ФР) длительностей обслуживания с конечными первыми двумя моментами, в терминах ПЛС в условиях критической загрузки в [1] получен класс предельных распределений для вектора стационарных времен ожидания в случае дисциплин абсолютного и относительного приоритета. Работе [1] предшествовали работы [2, 3].

В [1] вопрос обращения многомерных предельных распределений решен полностью лишь при $r=3$.

Пусть $w_i(i=\overline{1, r})$-стационарное время ожидания i-вызова, ϱ_{i1}-загрузка системы $\overline{1, i}$-вызовами (1-вызовами, ..., i-вызовами) и существуют пределы:

$$\bar{c}_i = \lim_{\varrho_{r1}\uparrow 1} c_i\,(c_i = \varrho_i/\varrho_{i-1},\ \varrho_i = 1-\varrho_{i1},\ \varrho_0 = 1).$$

Из индексов $\overline{1, r}$ выделяем те и только те $1 \leqq p_1(=p) < p_2 < \ldots < p_m = r$, для которых $\bar{c}_{p_i}=0\ (i=\overline{1, m})$, и разобъем потоки на группы $P_i=\{j\colon p_{i-1} \leqq j < p_i\}$, $P_{m+1}=\{j\colon j > p_m\}$.

Тогда [1] существует предел ($\varrho_{r1}\uparrow 1$)

$$\lim P\{w_j^* < x_j(j=\overline{1,r})\} = \prod_{n=1}^{m+1} \lim P\{w_j^* < x_j(j\in P_n)\}, \qquad (1)$$

где $w_j^*=w_j\ (j\in P_1)$, $w_j^*=w_j/Mw_j\ (j\notin P_1)$, M-знак математического ожидания, а предельное распределение одно и то же для дисциплин относительного и абсолютного приоритета.

Настоящая работа посвящена описанию процедуры получения предельных распределений групп $P_i\ (i\geqq 2)$ и основана на том, что в [1] $\lim P\{w_j^*<x_j(j\in P_i)\}$ зависит только от констант $\bar{c}_j$ группы P_i.

3°. В силу вышесказанного, с целью нахождения [1] предлагается изучить случай дисциплины абсолютного приоритета с дообслуживанием и упростить систему изменениями начальных данных, сохраняющими в пределе неизменными отношения «недогрузок» $\bar{c}_i$ для данной группы.

Пусть группа фиксирована и содержит k потоков с константами $\bar{c}_1=0$, $\bar{c}_2>0, \ldots, \bar{c}_k>0$. Программа упрощений такова.

1. Приравнять нулю параметры потоков из последующих групп.

2. Потоки предыдущих групп объединить с первым потоком нашей группы и считать первым потоком нашей группы, что не меняет константы нашей группы.

3. Длительности обслуживания всех вызовов считать показательно распределенными с единичным параметром.

4. Положить

$$a_i = (\tilde{c}_{i-1}\tilde{c}_i)\varrho(\tilde{c}_1=1,\ \tilde{c}_i=\bar{c}_2\ldots\bar{c}_i,\ i=\overline{2,k};\ \varrho=1-a_1),$$

где a_i-параметр i-го потока нашей группы.

Тогда предел отношений «недогрузок» равен ($\varrho\downarrow 0$)

$$\lim(1-\sigma_i)/(1-\sigma_{i-1})=\bar{c}_i \quad (\sigma_i=a_1+\ldots+a_i,\ i=\overline{1,k},\ \sigma_0=0,\ \sigma=\sigma_k).$$

Изучим полученную приоритетную систему.

4°. Пусть $p_k(n)$ $(n=(n_1,\ldots,n_k))$-стационарная вероятность наличия в системе в момент t n_1 1-вызовов, $\ldots$, n_k k-вызовов;

$$P_k(z)=\sum p_k(n)\,z_1^{n_1}\ldots z_k^{n_k} \quad z=(z_1,\ldots,z_k).$$

Введением дополнительного события выводим

$$P_k(z)\{[\sigma-az]_1^k+1-z_1^{-1}\}=(1-\sigma_k)(1-z_k^{-1})+\sum_{j=1}^{k-1}P_k(0^jz)(z_{j+1}^{-1}-z_j^{-1}), \qquad (2)$$

где

$$[\sigma-az]_i^j=\sum_{m=i}^{j}(a_m-a_mz_m), \quad (0^jz)=(\underbrace{0,\ldots,0}_{j},z_{j+1},\ldots,z_r).$$

Из (2) находим $\left(z_i^i = \underbrace{(z_i, \ldots, z_i)}_{i}\right)$ уравнение

$$P_k(z_i^i z) \cdot \{\sigma_i(1-z_i)+[a-az]_{i+1}^k+1-z_i^{-1}\}-P_k(0^i z)\cdot(z_{i+1}^{-1}-z_i^{-1}) =$$

$$= (1-\sigma_k)(1-z_k^{-1})+\sum_{j=i+1}^{k-1} P_k(0^j z)(z_{j+1}^{-1}-z_j^{-1}),$$

правая часть которого не зависит от z_i. Подставляем его в левую часть

$$z_i = \varphi_{ki} = (2\sigma_i)^{-1}\cdot\left\{\sigma_i+1+[a-az]_{i+1}^k-\sqrt{(\sigma_i+1+[a-az]_{i+1}^k)^2-4\sigma_i}\right\},$$

откуда следует уравнение для $P_k(0^i, z)$, которое позволяет из (2) вычислить $P_k(z)$:

$$P_k(z) = \frac{(z_k-1)(1-\sigma_k)}{z_1(\sigma-az+1)-1}\prod_{i=1}^{k-1}\frac{z_i-\varphi_{ki}}{z_{i+1}-\varphi_{ki}}. \tag{3}$$

Из (3) вытекает формула $(j=\overline{1, k-1})$

$$P_k(z_j^j z) = \frac{R_{j+1}}{R_j}\cdot\frac{z_j-\varphi_{kj}}{z_{j+1}-\varphi_{kj}} P_k(z_{j+1}^{j+1} z), \tag{4}$$

где

$$R_j = z_j(\sigma_j-\sigma_j z_j+[a-az]_j^k+1)-1 = R_j(\vec{z}_j).$$

5°. В дальнейшем вектор $(\cdot_i, \ldots, \cdot_k)$ обозначается $\vec{\cdot}_k$.

Положим $(i=\overline{1, k};\ s_i \geqq 0)$:

$$\bar{\omega}_{ki}(\vec{s}_i) = M\exp\left\{-\sum_{i=1}^{k} s_i \bar{w}_i\right\},\ \omega_{ki}(\vec{s}_i) = M\exp\left\{-\sum_{j=i}^{k} s_j w_j\right\},$$

где $\bar{w}_i$-условное стационарное время ожидания i-вызова при условии прекращения с момента ее отсчета поступлений, а w_i-безусловное.

Ясно, что $(i=\overline{1, k};\ s_i \geqq 0)$:

$$\bar{\omega}_{ki}(\vec{s}_i) = M\exp\left\{-s^{\{i\}}\bar{w}_i-\sum_{j=i+1}^{k} s^{\{j\}}(\bar{w}_j-\bar{w}_{j-1})\right\} = P_k(u_i^i u) \overset{\text{def}}{=} \tilde{\omega}_{ki}(\vec{s}^{\{i\}}),$$

где

$$s^{\{i\}} = s_i+\ldots+s_k,\ u_i = (1+s^{\{i\}})^{-1}.$$

В силу (4)

$$\bar{\omega}_{ki}(\vec{s}_i) = T_i(\vec{s}^{\{i\}})\cdot T_2(\vec{s}^{\{i\}})\cdot\tilde{\omega}_{ki+1}(\vec{s}_{i+1}), \tag{5}$$

где

$$T_1(\vec{s}^{\{i\}}) = R_{i+2}(\vec{u}_{i+1})/R_i(\vec{u}_i),$$

$$T_2(\vec{s}^{\{i\}}) = \big(u_i-\varphi_{ki}(\vec{u}_{i+1})\big)/\big(u_{i+1}-\varphi_{ki}(\vec{u}_{i+1})\big).$$

Можно показать, что $(i=\overline{1, k};\ s_i \geqq 0)\ \omega_{ki}(\vec{s}_i) = \tilde{\omega}_{ki}(\vec{\alpha}_i)$, где $\alpha_1, \ldots, \alpha_k$ задаются рекуррентно

$$\alpha_1 = s_1+\alpha_2,\ \alpha_i = y_1(\ldots(y_{i-1}(s_i+y_{i+1}^*)\ldots),\ y_{k+1}^* \equiv 0,\ y_i^* = y_{i+1}(s_i+y_{i+1}^*).$$

Здесь

$$y_i(s) = s+\frac{a_i}{2\sigma_i}\left\{\sigma_i-s-1+\sqrt{(s+\sigma_i+1)^2-4\sigma_i}\right\}.$$

Тогда на основе (5) получаем $(j=\overline{1, k-1})$

$$\omega_{kj}(\bar{s}_j) = T_1(\bar{\alpha}_j)\cdot T_2(\bar{\alpha}_j)\omega_{kj+1}(\bar{s}_{j+1}). \tag{6}$$

6°. Для вычисления пределов $\lim P\{w_j/Mw_j<x_j(j=\overline{1, k})\}$ вводим обозначения $(a\in(0, 1); j=\overline{2, k})$:

$$\Delta(t) = \sqrt{t+\frac{1}{4}}-\frac{1}{2},\ \Lambda_a(t) = at+(1-a)\Delta(t),\ s_j^* = \bar{c}_j s_j,$$

$$w^{\{1\}} = s_1+w^{\{2\}},\ w^{\{j\}} = \bar{c}_{j-1}\Delta(s_j^*+v_{j+1}),\ w^{\{k+1\}} = 0,\ w_j = w^{\{j\}}-w^{\{j+1\}},$$

где $v_m\,(m=\overline{3, k})$ определяются рекуррентно

$$v_m = \bar{c}_{m-1}\Lambda_{\bar{c}_{m-1}}(s_m^*+v_{m+1}),\ v_{k+1} = 0.$$

Полагая $s_i^*=s_i/Mw_i$, произведем при $\varrho\downarrow 0$ выкладки $(i=\overline{1, k}; j=\overline{2, k}; s_i\geqq 0)$:

$$Mw_1 \sim \varrho^{-1},\quad Mw_j \sim (\bar{c}_{j-1}\cdot\bar{c}_j\varrho^2)^{-1},$$

$$y_{j+1}^*(\bar{s}_{j+1}) = \bar{c}_{j-1}^2\varrho^2 v_{j+1}(1+o_\varrho(1)),\ \alpha_j^* = \alpha_j(\bar{s}_j^*) = \varrho w^{\{j\}}(1+o_\varrho(1)),$$

$$D_i(\varrho\cdot\vec{w}^{\{i\}}) = -\varrho^2\left\{c_i w^{\{i\}}+(w^{\{i\}})^2-\sum_{j=k+1}^{k}(\bar{c}_{j-1}-\bar{c}_j)w^{\{j\}}\right\}(1+o_\varrho(1)).$$

Приведенные асимптотические соотношения позволяют установить существование пределов: $\lim T_j(\vec{\alpha}_i^*)\ (j=1, 2)$, причем

$$\lim_{\varrho\downarrow 0} T_1(\alpha_i)\cdot T_2(\alpha_i^*) = \frac{w^{\{i+1\}}+(\bar{c}_i/2)+\sqrt{q_i+\frac{\bar{c}_i^2}{4}}}{w^{\{i\}}+(\bar{c}_i/2)+\sqrt{q_i+(\bar{c}_i^2/4)}} \overset{\mathrm{d}}{=} I_i(\bar{s}_i), \tag{7}$$

где

$$q_i = \sum_{j=i+1}^{k}\bar{c}_{j-1}(1-\bar{c}_j)\cdot w^{\{j\}}.$$

Наконец, обозначив $\hat{\omega}_{ki}(\bar{s}_i)=\lim\omega_{ki}(\bar{s}_i^*)$, получаем

$$\hat{\omega}_{kj}(s_j) = I_j(\bar{s}_j)\cdot\hat{\omega}_{kj+1}(\bar{s}_{j+1}).$$

Таким образом, вопрос получения предельного распределения группы сводится к вопросу обращения функций $I_j(\bar{s}_j)\ (j=\overline{1, k})$, выписанного в (7) в терминах величин $w^{\{j\}}$.

7°. Произведя переобозначения

$$\lambda_j = \frac{w^{(j+1)}}{\tilde{c}_j},\ \nu_j = \frac{\tilde{c}_{j-1}^2}{\tilde{c}_j^2}(1-\bar{c}_j),\ \mu_i = \frac{w^{(i)}-w^{(i+1)}}{\tilde{c}_i},$$

получаем

$$I_i = \int_0^\infty e^{-\mu_i x}\,dC_i(x), \tag{8}$$

где

$$C_i(x) = \exp\left\{-x\left(\lambda_i+\frac{1}{2}+\sqrt{\sum_{j=i}^{k-1}\nu_j\lambda_j+\frac{1}{4}}\right)\right\}.$$

Воспользовавшись формулой обращения (23.91) из [4]

$$e^{-u\sqrt{\mu}} = \int_0^\infty e^{-\mu v}\Psi(u, v)\,dv,\ \Psi(u, v) = \frac{u}{2v\sqrt{\pi v}}\exp(-u^2/4v),$$

имеем

$$C_i(x) = e^{-\lambda_i x}\int_0^\infty e^{-x/2}\exp\left\{-\left(\sum_{j=i}^{k-1}\nu_j\lambda_j+\frac{1}{4}\right)v\right\}\Psi(x, v)\,dv,$$

что путем преобразований сводится к многомерному интегралу

$$C_i(x) = \nu_i^{-1}\underbrace{\int_0^\infty\ldots\int_0^\infty}_{k-i}\exp\left\{-\sum_{j=i}^{k-1}\lambda_j t_j\right\}\chi(t_i > x)\,d_{t_i}\ldots d_{t_k}\Big\{e^{-x/2-\frac{v-x}{4\nu_i}}$$

$$\Psi\left(x, \nu_i^{-1}(v-x)\right)\chi\left(\nu_i\min_{i+1\leqq j<k+1}(\nu_j^{-1}t_j) > v-x\right)dv\Big\},$$

где

$$\chi(u > v) = \begin{cases}1, & u > v,\\ 0, & u \leqq v.\end{cases}$$

Подставляя последнее выражение для $C_i(x)$ в правую часть (8), после замен $t_i'=t_i+x$, $\bar{c}_{i-1}x=t_{i-1}'$ с использованием равенства $\bar{c}_i^{-1}\lambda_{i-1}=\mu_i+\lambda_i$, получаем

$$I_i = \underbrace{\int^\infty\ldots\int^\infty}_{k-i+1}\exp\left\{-\sum_{j=i-1}^{k-1}\lambda_j t_{j+1}\right\}d_{t_i}\ldots d_{t_k}\Phi(\vec{t}_i),$$

где

$$\Phi(\vec{t}_i) = -\int_0^{t_{i+1}}\nu_i^{-1}\exp\left(-\frac{\bar{c}_i t_i}{2}-\frac{v}{4\nu_i}\right)\Psi(\bar{c}_i t_i, \nu_i^{-1}v)\chi\left(\nu_i\min_{i+2\leqq j\leqq k}(\nu_{j-1}^{-1}t_j) > v\right)dv.$$

Произведено обращение I_j при условии, что параметрами ПЛС служат величины λ_j $(j=\overline{i, k})$.

8°. Обратим I_j, считая параметрами ПЛС $v_j\,(j=\overline{i+2, k})$.

$$d_{t_i}\dots d_{t_k} = -v_i^{-1}\left\{\exp\left(-\frac{t_{i+1}}{4v_i}\right)dt_{i+1}\right\}.$$

$$\left\{d_{t_i}[e^{-(\bar{c}_i\bar{t}_i/2)}\Psi(\bar{c}_it_i,\, v_i^{-1}t_{i+1})]\right\}\cdot\left\{\prod_{j=i+2}^{k} d_{t_j}\left[\chi\left(t_j>\frac{t_{i+1}v_{j-1}}{v_i}\right)\right]\right\}.$$

После несложных преобразований находим

$$I_i = \int_0^\infty\dots\int_0^\infty \exp\left\{-\sum_{j=i}^{k} s_{j-1}^* z_j\right\}\exp\left\{-\sum_{j=i}^{k} v_j z_j\right\}\Phi_1(\vec{z}_i)\,dz_i\dots dz_k,$$

где

$$\Phi_1(\vec{z}_i) = e^{-\frac{1}{4}\sum_{j=i}^{k} z_j}\left[\int_{x=0}^{\infty}\int_{y=0}^{\infty} e^{-\frac{y}{4v_i}+\frac{x+y}{2}}\prod_{j=i}^{k} e^{\frac{yv_{j-1}}{v_i}}\Psi\left(\frac{yv_{i-1}}{v_i},\, z_j\right)\right].$$

$$d_x\{e^{-(\bar{c}_i x/2)}\Psi(\bar{c}_i x,\, v_i^{-1}y)\}\,dy.$$

9°. Так как

$$v_j = \beta_{j-1}\cdot(s_j^*+v_{j+1})+\gamma_{j-1}\cdot\left\{\sqrt{s_j^*+v_{j+1}+\frac{1}{4}}-\frac{1}{2}\right\}\quad(\gamma_j=\bar{c}_j(1-\bar{c}_j),\ \beta_j=\bar{c}_j^2),$$

то

$$\exp\left\{-\sum_{j=i}^{k} v_j z_j\right\} = \underbrace{\int_0^\infty\dots\int^\infty}_{k-i+1}\exp\left\{\sum_{j=i+1}^{k} v_j t_{j-1}\right\}\Phi_2(\vec{z}_i,\, \vec{t}_i). \tag{9}$$

$$\exp\left\{-\sum_{j=i}^{k}(t_j+\beta_{j-1}z_j)s_j^*\right\}d_{t_i}\dots d_{t_k},$$

где

$$\Phi_2(\vec{z}_i,\, \vec{t}_i) = \prod_{j=i}^{k} e^{-t_j\left(z_j\beta_{j-1}+\frac{1}{4}\right)+\frac{\gamma_{j-1}z_j}{2}}\Psi(\gamma_{j-1}z_j,\, t_j).$$

В силу (9) просто проверяется, что

$$\exp\left\{-\sum_{j=i}^{k} v_j z_j\right\} = M\exp\left\{-\sum_{j=i}^{k} s_j^* W_j^i(\vec{z}_i)\right\},$$

где вектор-процесс

$$\vec{W}_i(\vec{z}_i) = (W_i^i(\vec{z}_i),\, \dots,\, W_k^i(\vec{z}_i))$$

определяется рекуррентно следующим образом

$$\vec{W}_i(\vec{z}_i) = (0,\, W_{i+1}(\vec{Y}_i^{\{0\}}(\vec{z}_i)))+\vec{E}_i(\vec{z}_i)+\vec{Y}_i(z_i)$$

Здесь $\vec{W}_{i+1}(\vec{t}_{i+1})$ не зависит от $\vec{W}_i(z_i)$,

$$\vec{E}_i(z_i) = (\beta_{i-1}z_i,\, \dots,\, \beta_{k-1}z_k),$$

а

$$\vec{Y}_i(\vec{z}_i) = (\vec{Y}_i^{\{0\}}(\vec{z}_i),\, Y_k^i(\vec{z}_i))\quad(\vec{Y}_i^{\{0\}}(\vec{z}_i) = (Y_i^i(\vec{z}_i),\, \dots,\, Y_{k-1}^i(\vec{z}_i)))$$

имеет плотность $\Phi_2(\vec{z}_i,\, \vec{t}_i)$

Последнее влечет за собой следующую рекуррентную связь

$$A_j(\vec{z}_j;\, \vec{x}_j) = \int\limits_{t_k=0}^{[x_k-\beta_{k-1} z_k]^+} dt_k \underbrace{\int\limits_0^\infty \dots \int\limits_0^\infty}_{k-j} A_{j+1}(t_j;\, \dots;\, t_{k-1};\, x_j, \dots, x_{k-1})$$

$$\Phi_2(\vec{z}_j;\, \vec{t}_j)\, dt_j \dots dt_{k-1},$$

где

$$A_i(\vec{t}_i;\, \vec{y}_i) = P\{\vec{W}_i(\vec{t}_i) < \vec{y}_i\},$$

откуда выводим

$$I_i = \int\limits_0^\infty \dots \int\limits_0^\infty \exp\left\{-\sum_{j=i}^{k} s_{j-1}^* z_j\right\} \Phi_3(\vec{z}_i)\, dz_i \dots dz_k.$$

Здесь

$$\Phi_3(\vec{z}_i) = \int\limits_0^\infty \dots \int\limits_0^\infty \Phi_1(\vec{z}_i - \vec{x}_i) d_{x_i} \dots d_{x_k} A_i(\vec{z}_i - \vec{x}_i;\, \vec{x}_i).$$

Процедура обращения полностью обрисована.

Description of a class of limit distributions in single server priority queues

E. A. Danielian

In a single server queuing system with waiting room r streams of customers are arriving. It is supposed that the first two moments of the serving distribution functions are finite.

Let $w_i(i=\overline{1,r})$ be the stationary waiting time of the i-th stream's customers, and ϱ_{i1} be the traffic intensity of customers of the first i streams.

It is known that in the case of FIFO and LIFO priority disciplines and $\varrho_{r1}\uparrow 1$ the joint distribution function of $w_i(i=\overline{1,r})$ under some normalization has a limit, which is found in terms of a multidimensional Laplace—Stiltjes transform.

In the paper a procedure for finding the corresponding multidimensional distribution function is described.

ВЫЧИСЛИТЕЛЬНЫЙ ЦЕНТР
АКАДЕМИИ НАУК АРМЯНСКОЙ ССР
УЛ. П. СЕВАКА, I
375044 ЕРЕВАН—44, СССР

Литература

[1] Даниелян, Э. А., Н. С. Земляной, Класс предельных распределений совместного стационарного распределения времен ожидания некоторых систем $\vec{M}_r|\vec{G}_r|1|\infty$ в условиях критической загрузки, *ДАН Арм. ССР*, том LXX, № I, 1980, стр. 3—10.

[2] Азларов, Т. А., Я. М. Хусаинов, Предельные теоремы для системы обслуживания, с абсолютным приоритетом в условиях большой загрузки, *Известия АН Узб. ССР* серия физ.-мат. наук, № 6, 1974, стр. 53—55.

[3] Azlarov, T. A., Ja. M. Husainov, Lecture notes in Mathematics, Springer-Verlag, 1976, pp. 575.

[4] Диткин, В. А., А. П. Прудников, Справочник по операционному исчислению М, Высшая школа, 1965.

(Поступило 5-ого декабря 1980 г.)

Definition of global properties of distributed computer systems by the analysis of system components method

By J. R. JUST

1. Introduction

A distributed computer system (abbr. DCS) consists of a number of distinct and logically connected communicating asynchronous sequential processes. A task realization in a DCS is the result of these process activities. During the task realization a user of a system creates a virtual network of processes. The virtual network of processes consists of a set of logically connected coprocesses. Each of the coprocesses for a given virtual process is executed in another processor of a DCS.

To gain a theoretical understanding of such systems, it is necessary to find a mathematical model which reflects essential features of these systems while abstracting irrelevant details away. Such the model allows problems to be stated precisely and make them amenable to mathematical analysis.

In the papers JUST [3, 4] it has been introduced a mathematical model of a distributed computer system and a mathematical model of their input/output behaviour. We use the concept process as a basic unit in our description of a DCS, and by a mathematical model of the process we shall mean a finite-control (FC-) algorithm of MAZURKIEWICZ, PAWLAK [5]. Formally, our model is based on a notion of so called vector of coroutines. This notion has been introduced by JANICKI [1, 2], in order to describe the semantics of programs with coroutines.

The main purpose of this paper is to define the global properties of distributed computer systems by the analysis of system components (coprocesses). We would like to answer the following questions. What can we say about all possible behaviours of the whole system, if we only know the local behaviour of all particular components of a DCS? Is it possible to analyse each component independently, and then to assemble all local properties in order to get the global semantics of the virtual process executed in a DCS?

To solve these problems, we extend the theory in JUST [4], and adapt some elements of the theory from JANICKI [1].

2. The model of a distributed computer system

The mathematical model of a distributed computer system has been introduced in the paper JUST [3]. In this chapter basic facts, important to the problem examined in this paper, will be presented. For more details the reader is advised to consult JUST [3, 4].

For every $n=1, 2, \dots,$ let $[n]=\{1, 2, \dots, n\}$. For each alphabet Σ let $\Sigma^* = \bigcup_{i=0}^{\infty} \Sigma^i$, $\Sigma^+ = \Sigma^*\Sigma$, $\Sigma^{\cdot} = \Sigma \cup \{\varepsilon\}$ where ε is the empty word. The remaining notation of the paper is standard.

By a model of a DCS we shall mean a 3-tuple

$$\mathrm{DCS} = (S, \mathrm{MP}, \mathrm{AL})$$

where

S is the structure of the system,
MP is the set of processes in the system,
AL is a mapping $\mathrm{AL}\colon \mathrm{MP} \to S$.

2.1. Structure of DCS. By the structure of DCS we mean a directed graph

$$S = (N, n_0, \mathrm{LT})$$

where

N is the set of nodes (interpreted as stations of computer network),
$n_0 \in N$ is the initial node,
$\mathrm{LT} \subseteq N \times N$ is the set of edges (interpreted as transmission lines).

2.2. Processes in DCS. In order to describe the set of processes in DCS we shall introduce a mathematical object, called a matrix of coprocesses.

2.2.1. Matrix of coprocesses. By a matrix of coprocesses we mean a system

$$\mathrm{MP} = (\mathscr{A}, I_0)$$

where

$$\mathscr{A} = \{A_{ij}\}_{\substack{i \in [m] \\ j \in [n]}}, \quad I_0 \in [m] \times [n].$$

A_{ij} is a coprocess, and I_0 indicates the start process. The set $\mathscr{A}$ can be interpreted as a matrix

$$\mathscr{A} = \begin{bmatrix} A_{11}, \dots, A_{1n} \\ \dots \\ A_{m1}, \dots, A_{mn} \end{bmatrix}.$$

Each line in the above matrix represents one process. A_{ij} is a 4-tuple which represents the j-th coprocess in the i-the process.

$$A_{ij} = (\Sigma_{ij}, V_{ij}, \sigma_{ij}, P_{ij}) \qquad \text{or} \qquad A_{ij} = (\emptyset, \emptyset, \{\varepsilon\}, \emptyset)$$

where

1) Σ_{ij} is an alphabet (of action symbols),
2) V_{ij} is an alphabet (of control symbols of A_{ij}),
3) $\sigma_{ij} \in V_{ij}$ is the start symbol of A_{ij},

4) P_{ij} is a finite subset of the set

$$\{(i,j)\}\times([m]\times[n])\times(V\times V^{\cdot})\times\Sigma_{ij}.$$

This means that P_{ij} is a finite set of 4-tuples of the form $(i\to r, j\to s, a\to b, R)$ where $i\to r\in\{i\}\times[m]$, $j\to s\in\{j\}\times[n]$, $a\to b\in V\times V^{\cdot}$, $R\in\Sigma_{ij}^{\cdot})$. P_{ij} is called the set of instructions of A_{ij}. Let $P=\bigcup_{i=1}^{m}\bigcup_{j=1}^{n}P_{ij}$.

Each instruction consists of four parts:

1) $i\to r$ indicates the process which will be active after the execution of the instruction (the r-th process will be active);

2) $j\to s$ indicates the coprocess which will be active after the execution of the instruction;

3) $a\to b$ indicates the way of execution of the component A_{ij}. This part of the instruction indicates the current and next point of the component A_{ij}.

4) R is the action of the instruction. It is an action name. R — because of its abstract character — will mean the program, the part of the program or an activity of the opertating system.

Every matrix of coprocesses can be represented graphically by means of graphs

$$a\cdot\xrightarrow[R]{}\cdot b \quad a\cdot\overset{i,j}{\underset{R}{\Longrightarrow}}\cdot b \quad a\cdot\xrightarrow[R]{i,j}\cdot b$$

to denote instructions $(i\to i, j\to j, a\to b, R)$, $(i\to i, j\to s, a\to b, R)$ and $(i\to r, j\to s, a\to b, R)$, respectively.

Put $=\bigcup_{i=1}^{m}\bigcup_{j=1}^{n}\Sigma_{ij}$. The set Σ is called the set of action names of the matrix MP.

Let $\mathrm{ms}=\bigtimes_{i=1}^{m}\bigtimes_{j=1}^{n}V_{ij}^{\cdot}$ ($\bigtimes$ is the cartesian product). The set $\mathrm{MS}=[m]\times[n]\times\mathrm{ms}$ is called the set of control states of MP.

Let $\mathrm{co}\colon[m]\times[n]\times\mathrm{ms}\to\bigcup_{i=1}^{m}\bigcup_{j=1}^{n}V_{ij}^{\cdot}$ be the function such that, for $\alpha\in\mathrm{ms}$ and $a_{ij}\in V_{ij}^{\cdot}$, $\mathrm{co}(i,j,\alpha)=a_{ij}$.

Each $(i\to r, j\to s, a\to b, R)$ can be regarded as a relation on the set Rel(MS) defined in the following way

$$y_1(i\to r, j\to s, a\to b, R)\,y_2\Leftrightarrow(\exists\alpha,\beta\in\mathrm{ms})\ y_1=(i,j,\alpha),\ y_2=(r,s,\beta)$$

and $\mathrm{co}(i,j,\alpha)=a$, $\mathrm{co}(r,s,\beta)=b$.

The set $\mathrm{MT}=\{(i,j,\alpha)\in\mathrm{MS}\mid \mathrm{co}(i,j,\alpha)=\varepsilon\}$ is called the set of terminal control states of MP. The set $\mathrm{ST}=\mathrm{MS}\times\Sigma^*$ is the set of states of MP.

Let $T\subseteq\mathrm{ST}\times\mathrm{ST}$ be the relation defined by the equivalence

$$(y_1,u_1)T(y_2,u_2)\Leftrightarrow\left[\left(\exists(i\to r, j\to s, a\to b, R)\in P\right)(y_1,y_2)\in\mathrm{MS}\ \&\ u_2=u_1R\right].$$

We put $y_0=(i_0,j_0,\alpha_0)$, where

$$\mathrm{co}(i,j,\alpha_0)=\begin{cases}\sigma_{ij} & \text{for } A_{ij}\neq\theta,\\ \varepsilon & \text{for } A_{ij}=\theta,\end{cases}$$

(θ denotes the empty coprocess of form $(\emptyset,\emptyset,\{\varepsilon\},\emptyset)$).

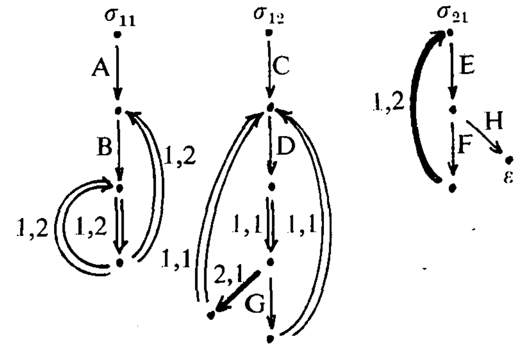
Put

$$L(\mathrm{MP}) = \{w \in \Sigma^* | (\exists y \in \mathrm{MT})(y_0, \varepsilon) T^*(y, w)\}.$$

$L(\mathrm{MP})$ is called the language generated by the matrix of coprocesses MP. The language $L(\mathrm{MP})$ represents the structure of a virtual network of processes, whereas each Σ_{ij} represents a set of names of actions (procedures) that should be executed in the framework of the (i, j)-th component of the system. This language is interpreted as a description of the semantics of the matrix MP.

Proving properties of the system of processes (in our model) is the same as proving properties of the language $L(\mathrm{MP})$. Properties of this language can be analysed by means of fixed-point methods (see Just [3]). These methods of analysis of the matrix of coprocesses need the knowlage about the form of all components prior to the analysis. All components must be analysed together. There is a question if it is possible to analyse each component independently, and then to get the global semantics of the matrix of coprocesses. We are going to discuss this main problem of our paper in section 3.

Example 1. Consider the system which consists of two processes, and the first process consists of two coprocesses. Let this system be represented by the following flowdiagram.



It can be proved that $L(\mathrm{MP}) = ABCD((EF \cup GB)D)^*EH$. (A method is given in Just [4]).

2.3. Mapping AL. The mapping AL specifies an allocation of a process.

3. From local to global properties of DCS

Proving properties of a system of processes (in our model) is the same as proving properties of the language $L(\mathrm{MP})$. But the language $L(\mathrm{MP})$ does not contain much information about the structure of the matrix of coprocesses. If we know this language only, we do not know anything about the number and the form of components. Now we define a language which gives $L(\mathrm{MP})$, the number of components, and sublanguages defined by components. Note that every component can be interpreted as certain right-linnear grammar.

Let $MP=(\mathscr{A}, I_0)$, where

$$\mathscr{A} = \{A_{ij}\}_{\substack{i\in[m]\\ j\in[n]}},\ I_0\in[m]\times[n]$$

and

$$A_{ij} = (\Sigma_{ij}, V_{ij}, \sigma_{ij}, P_{ij}) \quad (i = 1, \ldots, m, j = 1, \ldots, n)$$

is the matrix of coprocesses.

We define the following alphabets

$$\Lambda_k = \{\lambda_{kl_1}, \ldots, \lambda_{kl_{mn}}\} - \{\lambda_{kk}\},\ \Lambda_k' = \Lambda_k \cup \{\lambda_k\}$$

for

$$i, r\in[m], \quad j, s\in[n] \quad \text{and} \quad k = (i, j), \quad l = (r, s).$$

Let

$$\Lambda = \bigcup_{k\in[m]\times[n]} \Lambda_k \cup \{\lambda_{k_0}\}, \quad \Lambda' = \bigcup_{k\in[m]\times[n]} \Lambda_k' \quad (k_0 = (i_0, j_0)).$$

The set Λ in our model represents the set of names of actions of transmissions.

Let $\lambda(MP)$ be the matrix of coprocesses defined as follows

$$\lambda(PM) = (\mathscr{A}^\lambda, I_0), \quad \text{where} \quad \mathscr{A}^\lambda = \{A_{ij}\}_{\substack{i\in[m]\\ j\in[n]}},\ I_0\in[m]\times[n].$$

$$A_{ij}^\lambda = (\Sigma_{ij} \cup \Lambda_{ij}, V_{ij} \cup \{\sigma_{ij}'\}, \sigma_{ij}', P_{ij}^\lambda)$$

and

$$P_{ij}^\lambda = \{(i \to r, j \to s, a \to b, R\lambda_{kl}) \mid (i \to r, j \to s, a \to b, R)\in P_{ij} \,\&\, k \neq l\} \cup$$

$$\cup \{(i \to r, j \to s, a \to b, R) \mid (i \to r, j \to s, a \to b, R)\in P_{ij} \,\&\, k = l\} \cup$$

$$\cup \{(i \to i, j \to j, \sigma_{ij}' \to \sigma_{ij}, \mu) \ \text{ if } \ (i, j) = I_0 \ \text{ then } \ \mu = \lambda_{k_0} \ \text{ elsewhere } \ \mu = \varepsilon\}.$$

The language $L(\lambda(MP))$ contains all the necessary informations about the structure of the matrix MP.

Let $h_\Lambda\colon (\Sigma \cup \Lambda')^* \to \Sigma^*$ be the following homomorphism

$$(\forall R\in\Sigma\cup\Lambda')\, h_\Lambda(R) = \begin{cases} R & \text{if } R\in\Sigma, \\ \varepsilon & \text{if } R\notin\Sigma. \end{cases}$$

Corollary. $L(MP) = h_\Lambda(L(\lambda(MP)))$.

For arbitrary $i=1, \ldots, m, j=1, \ldots, n$ and $k=(i, j)$ let

$$\lambda(A_{ij}) = (\Sigma_{ij} \cup \Lambda_k', V_{ij}', \sigma_{ij}', Q_{ij}),$$

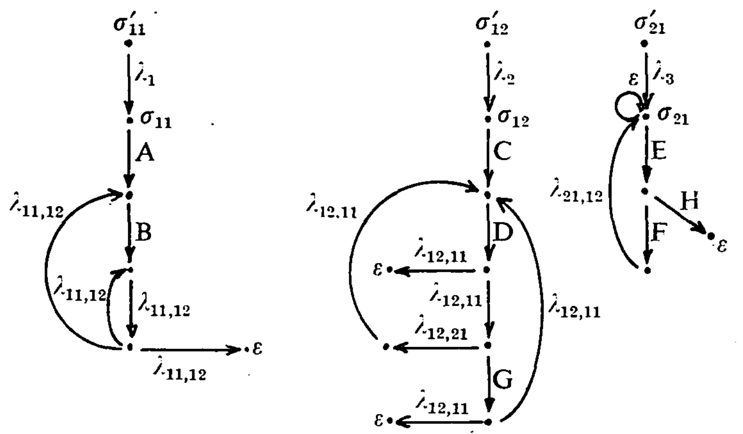
where

$$V_{ij}' = V_{ij} \cup \{\sigma_{ij}'\}, \quad \sigma_{ij}' \notin V_{ij},$$

$$Q_{ij} = \{a \to Rb \mid \exists (i \to r,\ j \to s,\ a \to b,\ R)\in P_{ij}^\lambda\} \cup \{\sigma_{ij}' \to \lambda_k \sigma_{ij}\}.$$

Note that $\lambda(A_{ij})$ is a right-linnear grammar.

Example 2. These grammars for our matrix of coprocesses (see Example 1) can be described by the following graphs.



Let $L(\lambda(A_{ij}))$ denote the language generated by these grammars. Note that the language $L(\lambda(A_{ij}))$ not only contains an information on "actions" (elements of Σ_{ij}) of the component A_{ij}, but also on points of resumptions of another components, and an information on "actions of transmissions" as well.

For arbitrary $i, r\in[m]$, $j, s\in[n]$ and $(i\to r, j\to s, a\to b, R)$ let num: $[m]\times[n]\to\to[m\cdot n]$ be defined in the following way: num $(i,j)=(i-1)\cdot m+j$ and for $\lambda_{kl}\in\Lambda$, $k=$num (i,j), $l=$num (r,s). Let $q=m\cdot n$.

Example 3. Languages $L(\lambda(A_{ij}))$ generated by grammars given in Example 2 are the following

$$L(\lambda(A_{11}))=\lambda_1AB\lambda_{12}((\lambda_{12}\cup\lambda_{12}B)\lambda_{12})^*\lambda_{12},$$
$$L(\lambda(A_{12}))=\lambda_2CD\lambda_{21}((G\lambda_{21}\cup\lambda_{23}\lambda_{21})D\lambda_{21})^*\lambda_{23},$$
$$L(\lambda(A_{21}))=\lambda_3(EF\lambda_{32}\cup\varepsilon)^*EH.$$

Let $\Delta_1, ..., \Delta_q, \Gamma_1, ..., \Gamma_q$ be sets defined in the following way

$$(\forall k\in[q])\quad \Delta_k=\{\lambda_{1k}, ..., \lambda_{k-1k}, \lambda_{k+1k}, ..., \lambda_q\},$$

$$\Gamma_k=\bigcup_{t\neq k, t=1}^{q}(\Sigma_t\cup\Lambda_t)-\Delta_k.$$

For arbitrary $k, l\in[q]$ and $k\neq l$ let Γ_{kl} be the following set

$$\Gamma_{kl}=\Sigma_l^*(\Lambda_l-\{\lambda_{lk}\})\Gamma_k(\Delta_k\cup\{\varepsilon\})\cup\Sigma_l^*\Delta_{lk}\cup\Sigma_l^*.$$

Let $\psi\colon(\Sigma\cup\Lambda')\to 2^{(\Sigma\cup\Lambda')^*}$ be the substitution of languages defined in the following way

$$(\forall R\in\Sigma\cup\Lambda')\psi(R)=\begin{cases}R & \text{if}\quad R\in\Sigma\cup\{\lambda_{k_0}\},\\ \lambda_{k_0}\Gamma_k^*\Delta_k & \text{if}\quad R=\lambda_k\ \&\ k\in[q]-\{k_0\},\\ R\lambda_{kl} & \text{if}\quad R=\lambda_{kl}\ \&\ k\neq l.\end{cases}$$

The function ψ is called the basic semantic function. This function has been defined by JANICKI [1] in order to describe the local semantics of a vector of coroutines.

A component A_{ij} of MP is called final if there exists an instruction $(i \to r, j \to s, a \to b, R)$ such that $b = \varepsilon$. The set of all final components of MP will be denoted by $\mathrm{FIN}_{\mathrm{MP}}$. We restrict our attension to the matrix of coprocesses with the property card $(\mathrm{FIN}_{\mathrm{MP}}) = 1$.

Theorem. For every matrix of coprocesses (of form defined in this paper)

$$L(\lambda(\mathrm{MP})) = \bigcap_{k=1}^{q} \psi(\lambda_k L(\lambda(A_k))),$$

where

$$A_k = A_{\mathrm{num}(i,j)} (A_{ij} \text{ is in MP}, \ i \in [m], j \in [n]).$$

The proof of the above theorem follows from considerations which have been described in [1, 2].

EXAMPLE 4. Let us consider the distributed computer system which consists of three processors, connected over a communication system. These processors execute particular parts (coprocesses) of the virtual process. We know these coprocesses only. In our model they are given in the form of components of the matrix of coprocesses (see Example 1), and can be interpreted as certain right-linear grammars (see Example 2). The languages generated by these grammars, are given in Example 3.

On the basis of these languages and by taking into consideration the Theorem, we can obtain the following language

$$L(\lambda(\mathrm{MP})) = \lambda_1 AB\lambda_{12} CD\lambda_{21} ((\lambda_{12}\lambda_{23} EF\lambda_{32}\lambda_{21} \cup \lambda_{12} G\lambda_{21} B)\lambda_{12} D\lambda_{21})^* \lambda_{12}\lambda_{23} EH.$$

This language describes all possible behaviours of our distributed computer system — both computations and transmissions.

From this and from Corollary 1 it follows that $L(\mathrm{MP}) = h_A(L(\lambda(\mathrm{MP}))) = ABCD((EF \cup GB)D)^* EH$. In our model this language is interpreted as a description of the semantics of the matrix of coprocesses (the semantics of the virtual network of processes).

4. Final comment

Treating distributed systems as the superposition of sequential subsystems is the natural way of analysis and synthesis of systems. This paper is an attempt to give a formal approach to this problem. Similar problems are considered in [1, 2, 3, 4], and from a different point of view in [6].

WARSAW TECHNICAL UNIVERSITY
DEPARTMENT OF ELECTRONICS, IPE
UL. NOWOWIEJSKA 15/19 p. 230 A
00-661 WARSAW, POLAND

References

[1] Janicki, R., Analysis of coroutines by means of vector of coroutines, *Fund. Inform.*, v. 2, 3, 1979.
[2] Janicki, R., Analysis of vector of coroutines by means of components, Proc. of 2-nd Symp. on Fundamentals of Computation Theory, Berlin, 1979.
[3] Just, J. R., An algebraic model of the distributed computer system, Proc. of 5-th Conf. on the Theory of Operating Systems, Visegrad, 1979.
[4] Just, J. R., Analysis and synthesis of distributed computer systems by algebraic means., Proc. of 6-th Conf. on the Theory of Operating Systems, Visegrad, 1980.
[5] Mazurkiewicz, A., Z. Pawlak, Mathematical Foundation of Computer Science, PWN, Warsaw, 1978.
[6] Winkowski J., An algebraic approach to distributed computations, ICS PAS Reports, 377, Warsaw, 1979.

(Received Dec. 3, 1981)

INDEX — TARTALOM

ISSN 0324—721 X

Felelős szerkesztő és kiadó: Gécseg Ferenc
A kézirat a nyomdába érkezett: 1981. december 28
Megjelenés: 1982. június 30
Példányszám: 1025. Terjedelem: 8,75 (A/5) ív
Készült monószedéssel, íves magasnyomással
az MSZ 5601 és az MSZ 5602—55 szabvány szerint
81-5569 — Szegedi Nyomda — F. v.: Dobó József igazgató