Tomus 7. Fasciculus 1.

ACTA CYBERNETICA

FORUM CENTRALE PUBLICATIONUM
CYBERNETICARUM HUNGARICUM

Szeged, 1985
Curat: Universitas Szegediensis de Attila József nominata

7. kötet 1. füzet

ACTA CYBERNETICA

A HAZAI KIBERNETIKAI KUTATÁSOK
KÖZPONTI PUBLIKÁCIÓS FÓRUMA

Szeged, 1985. január
A Szegedi József Attila Tudományegyetem gondozásában

Frontiers of one-letter languages

Stephen L. Bloom

Introduction

In a talk titled "Infinite words" given in the spring of '83, Dana Scott introduced the notion of a convergent sequence of "word" (i.e. elements of a finitely generated free monoid). A sequence is convergent if, for every regular set R of words, all but finitely many of the terms of the sequence belong to R or all but finitely many belong to the complement of R. Scott stated a number of properties of the collection of (equivalence classes of) convergent sequences, some of which showed that this structure had been known earlier in the guise of a free "profinite" monoid. This monoid has a natural topology and contains a copy of the original free monoid (as a certain discrete subspace). In the case that the words are elements of the one-generated free monoid (i.e. the additive monoid N of the nonnegative integers) an explicit description of all convergent sequences was stated (and is derived here). Further, he posed several problems connected with this structure and the current paper addresses one of these.

Scott asked whether an investigation of the "frontiers" of languages would lead to a useful classification of languages. In an effort to answer this question, we looked at the simplest case: frontiers of subsets of N. An explicit description of these frontiers has been obtained. It is seen that the cardinality of the frontier of a subset of N cannot distinguish regular from nonregular sets, and a necessary and sufficient condition is given that a subset of N have an uncountable frontier.

In order to obtain these facts we derived a number of the facts mentioned by Scott in his talk. We have tried to give very elementary proofs and keep the paper self-contained.

Preliminaries

N will denote both the set of nonnegative (or "natural") numbers as well as the additive monoid $(N, +, 0)$. An infinite sequence will be denoted by diagonal brackets:

$$\langle x\rangle = \langle x_1, x_2, \ldots\rangle;$$

the *terms* of $\langle x\rangle$ are the elements x_n, $n>0$.

The value of a function f on an argument x will be written as xf or $f(x)$.

For positive integers k and l, the 1-generated finite monoid which is the quotient of N by the least monoid congruence identifying $k+l$ and k is denoted $N_{k,l}$. When $k=l=n!$, this monoid will be denoted M_n. One may assume that the elements of $N_{k,l}$ are the integers

$$0, 1, \ldots, k, k+1, \ldots, k+l-1.$$

The closure of a subset A of a topological space is denoted A^-, and the *frontier* of A is $A^- - A$.

In this paper, we will use the word "uncountable" to mean the cardinality of the powerset of the natural numbers.

1. Convergent sequences

Let $\langle x\rangle$ and $\langle y\rangle$ be infinite sequences of nonnegative integers. For a subset S of N, the nonnegative integers, define $\langle x\rangle$ to be "S-equivalent" to $\langle y\rangle$ if, all but finitely many of the terms of $\langle x\rangle$ belong to S iff all but finitely many of the terms of $\langle y\rangle$ do.

1.1. Definition. $\langle x\rangle$ is equivalent to $\langle y\rangle$, written $\langle x\rangle \sim \langle y\rangle$, if for every *regular* set R, $\langle x\rangle$ is R-equivalent to $\langle y\rangle$.

It is easy to verify that $\sim$ is indeed an equivalence relation on the infinite sequences of natural numbes. The $\sim$ – equivalence class of $\langle x\rangle$ will be written $\langle x\rangle^\sim$.

1.2. Definition. An infinite sequence is *convergent* if, for each regular set R, either all but finitely many terms of $\langle x\rangle$ belong to R (we will say $\langle x\rangle$ is "eventually in R") or $\langle x\rangle$ is eventually in $N \setminus R$, the complement of R.

We let *Conv* denote the set of all convergent sequences. Note that if $\langle x\rangle$ is convergent and $\langle y\rangle$ is equivalent to $\langle x\rangle$, then $\langle y\rangle$ is also convergent.

The set of equivalence classes of convergent sequences is denoted N^-. This set has both a monoid structure and a topological structure, as will be shown below, and is one of the more fascinating objects in the mathematical universe. Our goal in this first section is to give an explicit representation of the members of N^-.

We will say that the sequence $\langle y\rangle$ is a subsequence of the sequence $\langle x\rangle$ if there is a strictly increasing function f: $N \to N$ such that for each $n \in N$,

$$y_n = x_{nf}.$$

1.3. Proposition. If $\langle y\rangle$ is a subsequence of the convergent sequence $\langle x\rangle$, then $\langle y\rangle$ is equivalent to $\langle x\rangle$ and is therefore convergent.

A sequence of integers is *eventually increasing* if for each number b, $x_n > b$ for all but finitely many n; similarly a sequence $\langle x\rangle$ is *eventually constant* with value b if $x_n = b$ for all but finitely many n.

1.4. Proposition. If $\langle x\rangle$ is convergent, either $\langle x\rangle$ is eventually constant or eventually increasing.

Proof. Let R_b denote the regular set of integers greater than b. Then either $\langle x\rangle$ is eventually in R_b for every b, in which case $\langle x\rangle$ is eventually increasing, or not. If not, since $\langle x\rangle$ is convergent, $\langle x\rangle$ is eventually in the finite union of the singleton sets $\{0\}\cup\{1\}\cup\ldots\cup\{b\}$, for some b. We now apply the following lemma.

1.5. Lemma. If $\langle x\rangle$ is convergent and $\langle x\rangle$ is eventually in a finite union of regular sets $A1\cup\ldots\cup An$, then for some i, $\langle x\rangle$ is eventually in Ai.

By the Lemma 1.5, $\langle x\rangle$ is eventually constant, proving 1.4. The easy proof of 1.5 is omitted.

1.6. Proposition. Let $\langle x\rangle$ be a convergent sequence. Then either there is a constant sequence $\langle y\rangle$ or a strictly increasing sequence $\langle y\rangle$ with $\langle x\rangle$ equivalent to $\langle y\rangle$.

Proof. By 1.4, $\langle x\rangle$ is either eventually constant, in which case $\langle x\rangle$ has a constant subsequence, or $\langle x\rangle$ is eventually increasing, in which case $\langle x\rangle$ has a strictly increasing subsequence. By 1.3, the proof is complete.

We will characterize those strictly increasing sequences which are convergent. But first we give an alternate formulation of the notion of convergent sequence.

1.7. Proposition. The sequence of integers $\langle x\rangle$ is convergent iff

(1.7a) for each homomorphism h from the additive monoid $(N, +, 0)$ to a finite monoid M, the sequence

$$\langle xh\rangle = \langle x_1 h, x_2 h, \ldots\rangle$$

is eventually constant.

Proof. Suppose first that $\langle x\rangle$ is convergent and that $h\colon N\to M$ is a monoid homorphism. If M consists of the elements $m1, \ldots, mk$, then the sets

$$Ri = h^{-1}(mi)$$

are disjoint regular sets and $\langle x\rangle$ is eventually in the union $R1\cup\ldots\cup Rk=N$. By lemma 1.5, $\langle x\rangle$ is eventually in one Ri; i.e. $\langle xh\rangle$ is eventually constant.

Now suppose that (1.7a) holds. Let R be any regular set and recall the following fundamental fact:

any regular subset R of N may be written as

$$R = h^{-1}(X)$$

for some monoid homomorphism $h\colon N\to M$, M finite, and some $X\subset M$.

Thus, $\langle x\rangle$ is eventually in R if $\langle xh\rangle$ is eventually in X; otherwise, $\langle x\rangle$ is eventually in the complement of R.

The preceding proposition may be rephrased: a convergent sequence is one whose terms eventually cannot be distinguished by a finite automaton.

The next fact is proved in exactly the same way.

1.8. Proposition. If $\langle x\rangle$ and $\langle y\rangle$ are convergent sequences, then $\langle x\rangle$ is equivalent to $\langle y\rangle$ iff for every monoid homomorphism $h\colon N\to M$, M finite, $\langle xh\rangle$ and $\langle yh\rangle$ are eventually equal.

1.9. Corollary. If $\langle x\rangle$ and $\langle y\rangle$ are convergent, so is "$\langle x\rangle+\langle y\rangle$", where the "sum" is obtained by pointwise addition; further, if $\langle x\rangle\sim\langle x'\rangle$ and $\langle y\rangle\sim\langle y'\rangle$, then $\langle x\rangle+\langle y\rangle\sim\langle x'\rangle+\langle y'\rangle$.

Both facts follow from 1.7 and 1.8.

1.10. Corollary. N^- is a monoid, where the operation is that in 1.9 and the identity element is the equivalence class of the constant zero sequence.

The next proposition is quite useful.

1.11. Proposition. Suppose that $\langle x\rangle$ is a strictly increasing sequence. Then $\langle x\rangle$ is convergent iff, for each $n>0$,

$$x_m \equiv x_p \,(\text{mod } n!) \tag{1.11a}$$

for all but finitely many m and p. If $\langle x\rangle$ and $\langle y\rangle$ are strictly increasing convergent sequences, $\langle x\rangle \sim \langle y\rangle$ iff, for each $n>0$,

$$x_m \equiv y_m \,(\text{mod } n!) \tag{1.11b}$$

for all but finitely many m.

Proof. If $\langle x\rangle$ is convergent, then (1.11a) holds, by 1.7 applied to the canonical homomorphism $N \to Z/n!$, where $Z/n!$ denotes the ring of integers modulo $n!$. Conversely assume that $\langle x\rangle$ satisfies the property (1.11a). We show that $\langle x\rangle$ is convergent by using Lemma 1.7 and the following facts.

1.12. Fact. Any monoid homomorphism h: $N \to M$, M finite, factors as a composite

$$\begin{tikzcd} N \arrow[r, "e"] \arrow[dr, "h"'] & N_{k,l} \arrow[d, "i"] \\ & M \end{tikzcd} \tag{1.13}$$

where e is a surjective homomorphism, i is injective and $N_{k,l}$ was defined in the Preliminary section.

1.14. Fact. Any homomorphism h: $N \to N_{k,l}$ factors as

$$\begin{tikzcd} N \arrow[r, "k_n"] \arrow[dr, "h"'] & N_{n!,n!} \arrow[d, "h\#"] \\ & N_{k,l} \end{tikzcd} \tag{1.15}$$

where n is any number greater than both k and l. Note that the map h (and $h\#$) satisfies:

$$\begin{aligned} xh &= x \quad \text{if} \quad x<k; \\ &= k+\text{Rem}\,(x-k, l) \quad \text{if} \quad x \geqq k. \end{aligned} \tag{1.16}$$

We now show that the property of (1.11a) implies that $\langle x\rangle$ is convergent. Suppose that $h\colon N\to M$ is a monoid homomorphism, M finite. Then by 1.12 and 1.14 we may as well assume that M is $N_{n!,n!}$ and h is defined by (1.16) when k and l are $n!$; thus we assume

$$\begin{aligned} zh &= z \quad \text{if} \quad z < n!; \\ &= n!+\operatorname{Rem}(z-n!,\, n!) \quad \text{if} \quad z \geqq n! \\ &= n!+\operatorname{Rem}(z,\, n!). \end{aligned}$$

It follows that $\langle xh\rangle$ is eventually constant, since $\langle x\rangle$ is strictly increasing, $x_m > n!$ for $m > n!$ and $x_m \equiv x_{m'}$ (mod $n!$), for sufficiently large m and m'. Thus, by (1.7), $\langle x\rangle$ is convergent. The proof is complete.

We now want to show that the monoid N^- is isomorphic to the (inverse) limit L of the diagram

$$\ldots \to M_n \xrightarrow{g_n} M_{n-1} \xrightarrow{g_{n-1}} \ldots \xrightarrow{g_2} M_1, \tag{1.17}$$

where $M_n = N_{n!,n!}$ and where the homomorphism $g_n\colon M_n \to M_{n-1}$ is defined, as one might expect by now, as

$$\begin{aligned} xg_n &= x \quad \text{if} \quad x < (n-1)!; \\ &= (n-1)!+\operatorname{Rem}\bigl(x,\, (n-1)!\bigr), \quad \text{if} \quad x \geqq (n-1)!. \end{aligned}$$

It is well known [G] that L may be described as the submonoid of πM_n consisting of all "compatible" sequences, i.e. all sequences $\langle z\rangle$, with $z_n \in M_n$, for each $n > 0$, and with $z_n g_n = z_{n-1}$, for $n > 1$. In order to prove N^- is isomorphic to L, we make two observations about the sequences in L.

1.18. Lemma. If $\langle z\rangle$ is a compatible sequence and for some n, $z_n < n!$, then $z_m = z_n$, for all $m > n$.

1.19. Corollary. If $\langle z\rangle$ is a compatible sequence in L and $\langle z\rangle$ is not eventually constant, then for all $n > 0$,

$$n! \leqq z_n (< 2n!).$$

Now let $k_n\colon N \to M_n$ be the canonical homomorphism (i.e. $1k_n = 1$ in M_n) and let $h_n\colon N^- \to M_n$ be the monoid homomorphism taking the equivalence class of the sequence $\langle x\rangle$ to the eventual value of the sequence $\langle xk_n\rangle$ in M_n.

1.20. Lemma. For each $n > 0$ the diagram below commutes.

$$\begin{tikzcd} N^- \arrow[r, "h_{n+1}"] \arrow[dr, "h_n"'] & M_{n+1} \arrow[d, "g_{n+1}"] \\ & M_n \end{tikzcd} \tag{1.21}$$

Proof. Since the diagram (1.22) commutes,

$$\begin{tikzcd} N \arrow[r, "k_{n+1}"] \arrow[dr, "k_n"'] & M_{n+1} \arrow[d, "g_{n+1}"] \\ & M_n \end{tikzcd} \tag{1.22}$$

if a is the eventual value of $\langle xk_{n+1}\rangle$ then ag_{n+1} is also the eventual value of $\langle xk_n\rangle$.

Now since the monoid L, equipped with the projection maps

$$p_n\colon \langle z\rangle \to z_n$$

is the limit of the diagram (1.17), there is a unique homomorphism

$$q\colon N^- \to L$$

such that for each n, $q, p_n = h_n$. We want to show that q is an isomorphism.

1.24. Lemma. q is surjective.

Proof. Let $\langle x\rangle$ be a sequence in L. If $\langle z\rangle$ is eventually constant with value a then $\langle a\rangle^{\sim} q = \langle z\rangle$, where $\langle a\rangle^{\sim}$ is the equivalence class of the constant sequence $\langle a, a, \ldots\rangle$. Otherwise $\langle z\rangle$ is strictly increasing. Let $\langle x\rangle$ be the sequence of integers with $x_n = z_n$ for all n (i.e. $\langle x\rangle$ is $\langle z\rangle$ considered as a sequence in N). Then we claim that for each n,

$$\langle x\rangle^{\sim} h_n = z_n$$

so that $\langle x\rangle^{\sim} q = \langle z\rangle$.

Indeed, for $m > n$, $x_m \geqq n!$ and

$$x_m \equiv z_n \pmod{n!}$$

by the definition of the g_m's. So the eventual value of $\langle xk_n\rangle$ is z_n. Thus q is surjective.

1.25. Lemma. q is injective.

Proof. Suppose that $\langle x\rangle^{\sim} q = \langle y\rangle^{\sim} q = \langle z\rangle$, where $\langle x\rangle$ and $\langle y\rangle$ are convergent sequences. We will show that $\langle x\rangle$ is equivalent to $\langle y\rangle$. Since this is clear in the case that $\langle x\rangle$ is eventually constant, assume that $\langle x\rangle$ is strictly increasing. Then the eventual value of the sequences $\langle xk_n\rangle$ and $\langle yk_n\rangle$ is z_n, so that, for all but finitely many values of m,

$$x_m \equiv y_m \pmod{n!}.$$

and hence $\langle x\rangle$ is equivalent to $\langle y\rangle$.

We have proved the following

1.26. Theorem. The monoid N^- is isomorphic to L.

It will sometimes be convenient to use L instead of N^- in order to get concrete representations. For example, notice how sequences in L are added. The n-th component of $\langle z\rangle + \langle z'\rangle$ is $z_n *_n z'_n$, where $*_n$ is the monoid operation in M_n. Thus, if both

$\langle z\rangle$ and $\langle z'\rangle$ are increasing, i.e. $n!\leqq z_n,\ z'_n$, then

$$z_n *_n z'_n = n!+\mathrm{Rem}\,(z_n+z'_n,\, n!).$$

Hence, aside from the notation "$n!$ +", the monoid operation *on increasing sequences in L* is the same as that on $\Pi(Z/n!\colon n>1)$, where $Z/n!$ is the ring of integers mod $n!$. Denoting the increasing sequences in L by $\mathbf{F}$, we have seen that $\mathbf{F}$ forms a subring of the product of the rings $(Z/n!\colon n>1)$. (In fact, $\mathbf{F}$ is the inverse limit of the diagram

$$\ldots \to Z/(n+1)! \to Z/n! \to \ldots$$

where the maps are the canonical ring homomorphisms.) We turn now to the question of getting an explicit description of the sequences in L (and thus N^-). The description depends heavily on the following

1.27. "Factorial" lemma. For each $n\geqq 1$ and each number x, $0\leqq x<n!$, there is a *unique* sequence $a_1, \ldots, a_{n-1}$ of integers with

$$0\leqq a_i\leqq i,\quad 1\leqq i<n$$

such that

$$x=\sum a_i i! = a_1+a_2 2!+\ldots+a_{n-1}(n-1)!.$$

(When $n=1$, the sequences is the empty sequence, whose sum is 0.)

This fact may be proved in a straightforward manner. Now, if $\langle z\rangle$ is an increasing sequence in L, for each n, $n!\leqq z_n<2n!$. Thus, we may write

$$z_n = n!+\sum_1^{n-1} a_i i!.$$

1.28. Proposition. If $\langle z\rangle$ is an increasing sequence in L, and for some n,

$$z_n = n!+\sum_1^{n-1} a_i i!,\quad\text{and}$$

$$z_{n+1} = (n+1)!+\sum_1^{n} b_i i!,$$

then for $i<n$, $b_i=a_i$.

Proof. Since $z_{n+1}g_{n+1}=z_n$, $\mathrm{Rem}\,(z_{n+1}, n!)=\sum a_i i!$ But $\mathrm{Rem}\,(z_{n+1}, n!)= =\sum b_i i!$. By the uniqueness part of the factorial lemma, the proposition is proved.

Hence, if $\langle z\rangle$ is an increasing sequence in L, there is a unique infinite *subdiagonal sequence* $\langle a\rangle$ (i.e. $0\leqq a_n\leqq n$, all n) such that

$$z_n = n!+\sum_1^{n-1} a_i i! \tag{1.28a}$$

for all n.

Let SD denote the set of all infinite subdiagonal sequences, and let SD^* denote the set of all finite subdiagonal sequences of integers. If $\langle a\rangle$ is any sequence in SD, let $z(a)$ denote the sequence defined by (1.28a) above. Then we have already proved part of the next theorem.

1.29. First representation theorem. For each sequence $\langle a\rangle$ in SD, $z(a)$ is an increasing sequence in L, and the map $SD \to \mathbf{F}$ defined by

$$\langle a\rangle \mapsto z(a)$$

is a bijection.

Proof. We already know the map is surjective. The fact that $z(a)$ is in fact in L (and hence in $\mathbf{F}$) is immediate, since $((n+1)!+\sum_1^n a_i i!)q_{n+1}=n!+\sum_1^{n-1} a_i i!$. If $\langle a\rangle$ and $\langle a'\rangle$ are distinct sequences in SD, then $z(a)$ and $z(a')$ are distinct by the factorial lemma. The theorem is proved.

It is easy to see that if $\langle z\rangle$ is an eventually constant sequence in L, then there is a unique finite subdiagonal sequence

$$\langle a_1, \dots a_{n-1}\rangle$$

such that for $k<n$

$$z_k = k!+\sum_1^{k-1} a_i i!,$$

and for $k>n$,

$$z_k = \sum_1^{n-1} a_i i!.$$

2. Topology

The standard topology on L (and thus on N^-) is inherited from the product topology on

$$\Pi M_n,$$

where each finite monoid M_n has the discrete topology. Thus the subbasis for the topology on L consists of all sets of the form

$$p_n^{-1}(X)$$

where p_n: $L\to M_n$ is the n-th projection map and X is some subset of M_n. Clearly L is Hausdorff in this topology, and is compact by the Tychonoff theorem since L is closed in the product. (Indeed, if $\langle z\rangle \notin L$, then for some n, $z_n g_n \neq z_{n-1}$. Then

$$p_n^{-1}(z_n)\cap p_{n-1}^{-1}(z_{n-1})$$

is an open set disjoint from L containing $\langle z\rangle$.) The subbasis sets are closed (as well as open) since

$$L-p_n^{-1}(X) = p_n^{-1}(M_n-X).$$

Thus L is a "Stone space". We will show that the Boolean algebra of the clopen (i.e. closed and open) subsets of L is isomorphic to the Boolean algebra of regular subsets of N.

Let i: $N\to L$ be the (unique) monoid homomorphism satisfying

$$i\cdot p_n = k_n \tag{2.0a}$$

for all $n>1$. The image of i consists of the eventually constant sequences. For $n>m$, let $g_{n,m}\colon M_n \to M_m$ be the composite of

$$M_n \xrightarrow{g_n} M_{n-1} \xrightarrow{g_{n-1}} \dots \xrightarrow{g_{m+1}} M_m$$

We will need the following fact.

2.1. Lemma. Suppose that $X \subset M_n$ and $Y \subset M_m$, where $n > m$. Then

$$\text{i)}\quad k_n^{-1}(X) \cap k_m^{-1}(Y) = k_n^{-1}(X \cap Z), \quad \text{and}$$

$$\text{ii)}\quad P_n^{-1}(X_n) \cap p_m^{-1}(Y) = p_n^{-1}(X \cap Z),$$

where $Z = g_{n,m}{}^{-1}(Y)$.

The easy proof is omitted.

2.2. Proposition. The subsets of L of the form

$$p_n^{-1}(X), \quad X \subset M_n,$$

are in fact a basis, being closed under finite union and finite intersection and complementation as well.

Proof. We have already noted the closure under complementation and closure under intersection follows from the preceding lemma, part ii). Closure under finite union follows from these two facts.

It follows from general topological principles that the clopen subsets of L are precisely the subsets of the form $p_n^{-1}(X)$, $n > 1$, $X \subset M_n$.

If S is a subset of N, let S^- denote the closure of $i(S)$ in L.

2.3. Lemma. Let $S = k_n^{-1}(X)$, where $X \subset M_n$. Then $S^- = p_n^{-1}(X)$. Thus the closure of a regular set is a clopen set in L.

Proof. Let $B = p_n^{-1}(X)$. Then S^- is contained in B since B is closed and $i(S) \subset \subset B$, by (2.0a). Let b be any point in B not in $i(S)$. We will show that b is a limit point of $i(S)$. Let $C = p_m^{-1}(Y)$ be any basis set containing b. We may assume that $n > m$. Applying 2.1,

$$B \cap C = p_n^{-1}(Z), \quad \text{and}$$

$$S \cap k_n^{-1}(Y) = k_n^{-1}(Z),$$

where $Z = g_{n,m}^{-1}(Y)$. Let R be the set $S \cap k_n^{-1}(Y)$. Then R is nonempty: otherwise, Z is empty, since k_n is surjective, and this would imply that $B \cap C$ would be empty, contradicting the fact that b is contained in this intersection. Now if $a \in R$, $i(a)$ is in $B \cap C$, by (2.0a) again, showing that b is a limit point of S. The proof of 2.3 is complete.

2.4. Lemma. Let R and S any subsets of N. Then $R \subset S$ iff $R^- \subset S^-$.

Proof. Only one direction is nontrivial. Suppose that R^- is contained in S^-. Let $x \in R$. Then the singleton set $\{x\}$ is a regular subset of N so the closure of $i(\{x\})$, say A, is a clopen subset of L, by 2.3 above. Since A is open and since $R^- \subset S^-$, A must contain a point of S. But since L is Hausdorff, A is a singleton set. Thus $x \in S$, proving the lemma.

2.5. Theorem. The map $R \mapsto R^-$ is a Boolean isomorphism from the Boolean algebra of regular subsets of N to the Boolean algebra of clopen subsets of L.

Proof. By Lemma 2.3, each regular subset of N is mapped to a clopen basis set and every clopen set in L is the closure of the image of a regular set. By Lemma 2.4, the map is an order (and hence Boolean) isomorphism.

The next fact follows from the standard proof of the Stone representation theorem.

2.6. Corollary. N^- is in bijective correspondence with the collection of ultrafilters on the Boolean algebra of regular subsets of N.

3. Some algebra

Recall that we have shown that the strictly increasing sequences **F** in L form a ring, isomorphic to the inverse limit of the diagram

$$\ldots \to Z/n! \to Z/(n-1)! \to \ldots$$

Although it is probably well known, we will indicate why **F** is also isomorphic as a ring to the product ring

$$\Pi(Z_p\text{: } p \text{ prime}),$$

where Z_p is the ring of p-adic integers.

One definition of Z_p (see [Kurosh, p. 154]) is the following:

3.1. Definition. Z_p consists of all sequences $\langle k \rangle$ of nonnegative integers satisfying

$$0 \leqq k_n < p^n, \tag{3.1a}$$

and

$$k_n \equiv k_{n+1} (\text{mod } p^n) \tag{3.1b}$$

for each $n > 1$. The sequences are added and multiplied pointwise, where the n-th component is reduced modulo p^n. Thus Z_p is the inverse limit of the diagram

$$\ldots \to Z/p^n \to Z/p^{n-1} \to \ldots \tag{3.2}$$

Kurosh gives an easy proof that

3.3. Z_p has no zero divisors.

We will show that each sequence $\langle z \rangle$ in **F** determines a sequence $\langle z \rangle \alpha_p$ in Z_p.

3.4. Definition of α_p: $\mathbf{F} \to Z_p$.

Given $\langle z \rangle$, for each n let $m = p^n!$. Then for all $t > m$,

$$z_t \equiv z_m\ (\text{mod } m),$$

so that

$$z_t \equiv z_m\ (\text{mod } p^n).$$

Then for each n, we define k_n as $\text{Rem}\,(z_m, p^n)$, and we let $\langle z \rangle \alpha_p = \langle k \rangle$.

Taking the target tupling of the maps α_p, p prime, we get a map

$$\alpha\colon \mathbf{F} \to \Pi(Z_p\colon\ p \text{ prime}).$$

Since the ring operations on both **F** and Z_p are defined componentwise and since reduction mod m preserves addition and multiplication, we have

3.5. Lemma. Each map α_p is a ring homomorphism and thus so is α.

3.6. Lemma. α is an isomorphism.

Proof. First we show that α is injective. Let $\langle z\rangle$ and $\langle z'\rangle$ be distinct sequences in **F**. Then for some n, $z_n \neq z_n'$. Factoring $n!$ as a product of powers of primes, there is some prime p and some m such that

$$z_n \not\equiv z_n' \,(\text{mod } p^m).$$

But for all $t > n!$,

$$z_t \equiv z_n \,(\text{mod } p^m)$$

and

$$z_t' \equiv z_n' \,(\text{mod } p^m).$$

Thus, $\langle z\rangle\alpha_p \neq \langle z'\rangle\alpha_p$, so that α is injective.

Now we prove α is in fact surjective. Suppose that we are given an element of $\Pi(Z_p\colon\ p$ prime), say

$$\langle y^p\rangle = \langle y^p(1), y^p(2), \ldots\rangle$$

for each prime p. We will construct a sequence $\langle z\rangle$ in **F** such that $\langle z\rangle\alpha_p = \langle y^p\rangle$ for each p. In order to do this, we construct an increasing sequence $\langle x\rangle$ in *Conv* with $x_1 = y^{p1}(1)$ (where $p1, p2, \ldots$, are the primes in increasing order) and for each $n > 1$,

$$\begin{aligned} &x_n > x_{n-1}; \qquad (3.7)\\ &x_n \equiv y^{p1}(n)\,(\text{mod } (p1)^n)\\ &x_n \equiv y^{p2}(n)\,(\text{mod } (p2)^n)\\ &\ldots\\ &x_n \equiv y^{pn}(n)\,(\text{mod } (pn)^n). \end{aligned}$$

Suppose that $\langle x\rangle$ satisfies the above conditions. Then $\langle x\rangle$ is obviously increasing and is, although not so obviously, convergent. Indeed, to prove $\langle x\rangle$ is convergent we will show that for each n, all but finitely many of the terms of $\langle x\rangle$ are congruent modulo $n!$. Recall that if q and q' are relatively prime, then

$$u \equiv v \,(\text{mod } qq')$$

iff

$$u \equiv v \,(\text{mod } q) \quad \textit{and} \quad u \equiv v \,(\text{mod } q').$$

Now factor a fixed $n!$ into a product of primes, say

$$n! = p1^{k1} \ldots pr^{kr},$$

whre $0 \leqq ki$, $i = 1, \ldots, r$. Let m be any number greater than both r and max ($k1, \ldots,$

... kr). Then for each $i<m$,

$$x_m \equiv y^{pi}(m) \,(\text{mod } pi^m),$$

so that

$$x_m \equiv y^{pi}(m) \,(\text{mod } pi^{ki})$$

since $m>ki$. But since $\langle y^{pi}\rangle$ is in Z_{pi},

$$y^{pi}(m) \equiv y^{pi}(ki) \,(\text{mod } pi^{ki}).$$

It follows that for $t>m$,

$$x_t \equiv y^{pi}(ki) \,(\text{mod } pi^{ki}), \quad \text{all} \quad i<r.$$

But then, for $t>m$,

$$x_t \equiv x_m \;\; (\text{mod } n!),$$

proving that $\langle x\rangle$ is convergent. Let $\langle z\rangle$ be the sequence in L determined by $\langle x\rangle$. Then $\langle z\rangle\alpha_p=\langle y^p\rangle$, for all p.

It remains to show how to obtain the sequence $\langle x\rangle$. But assuming we have found $x_1, ..., x_{n-1}$, we obtain x_n satisfying (3.7) by applying the Chinese Remainder Theorem.

The proof of the Theorem is complete.

4. Some closures

In section 2 we showed that if R is a regular subset of N, then the closure of $i(R)$ in L has the form

$$p_n^{-1}(X)$$

for some $X\subset M_n$.

4.1. Proposition. If R is an infinite regular subset of N, then R^- and thus its frontier are uncountable.

Proof. Suppose R is $k_n^{-1}(X)$, where $X\subset M_n$. If R is infinite, there is some element b in X with

$$n! \leqq b\,(<2n!).$$

Write b as $n!+\sum^{n-1} a_i i!$. Then, for all infinite subdiagonal sequences a which extend $\langle a_1, ..., a_{n-1}\rangle$, the element $z(a)$ is in the closure of R. Since there are uncountably many such sequences a, the proposition is proved.

Now we prove that if S is the nonregular (context sensitive) set of squares, i.e.

$$S=\{n^2\colon\, n\in N\}$$

then the frontier of S is also uncountable.

We will use two facts from the preceding section.

(4.2) Z_p has no zero divisors.

(4.3) $\mathbf{F}$ is isomorphic as a ring to

$$\Pi(Z_p\colon\, p \text{ prime}).$$

Let $\langle x\rangle$ be any sequence in $\mathbf{F}$ and let $\langle y\rangle=\langle x\rangle\langle x\rangle=\langle x\rangle^2$ (so that for each n, $y_n=n!+\text{Rem}\,(x_n^2, n!)$).

4.4. Lemma. $\langle y\rangle$ is a limit point of S.

Proof. Let B be a basis set containing $\langle y\rangle$. Then $B=p_n^{-1}(X_n)$, for some n and some subset X_n of M_n. Thus $y_n\in X_n$. If a is the natural number

$$a = n!+x_n$$

then $k_n(a^2)=y_n$, so that $i(a^2)\in B$, proving that $\langle y\rangle$ is a limit point of S.

Now we will show that there are uncountably many points in L of the form $\langle x\rangle\langle x\rangle$, which will complete the proof that S^- is uncountable. Indeed, in any ring with no zero divisors, if $u^2=v^2$, then $u=v$ or $u=-v$, since

$$u^2-v^2 = (u+v)(u-v).$$

Now we choose in each ring Z_p two elements, say x_p and y_p, such that x_p is disctint from y_p and from $-y_p$.

Then, for each function f: Primes → $\{0, 1\}$, let $z(f)$ be the element of $\Pi(Z_p$: p prime) defined by:

$$\begin{aligned} z(f)_p &= x_p \quad \text{if} \quad f(p) = 0;\\ &= y_p \quad \text{if} \quad f(p) = 1. \end{aligned}$$

Then, by 4.4, regarding each element $z(f)$ as an element of $\mathbf{F}$, $z(f)^2$ is a limit point of S. But if $f\neq f'$, then $z(f)^2\neq z(f')^2$; indeed, if $x_p=f(p)\neq f'(p)$, then $z(f)_p^2=x_p^2\neq y_p^2=z(f')_p^2$. We have proved:

4.5. Proposition. S has an uncountable frontier.

Thus the cardinality of the frontier of a language cannot distinguish regular from nonregular sets. In the next section we will characterize all sets whose frontier is uncountable.

5. Explicit topology

In this section we will use the first representation theorem to get a second, more geometric description of L and the topology on L. This description makes use of a locally finite rooted tree T (see [EBT]). The vertices of T are certain finite sequences in $SD^*\cup SD^*\times\{\perp\}$.

A vertex of T in the form of a pair $(\langle s\rangle, \perp)$ is written as a finite sequence $\langle s, \perp\rangle$ ending in the symbol $\perp$. The root of T is $\langle\perp\rangle$; the root has three immediate successors: $\langle\rangle$ (the empty sequence), $\langle 1\rangle$ and $\langle 0, \perp\rangle$; if

$$v = \langle a_1, \ldots, a_n, \perp\rangle$$

is a vertex in T, then $\langle a_1, \ldots, a_n\rangle$ is in SD^*, and v has the following $2n+3$ immediate successors:

$$\langle a_1, \ldots, a_n, 1\rangle, \ldots \langle a_1, \ldots, a_n, n+1\rangle \quad\text{and}$$

$$\langle a_1, \ldots, a_n, 0, \perp\rangle, \ldots, \langle a_1, \ldots, a_n, n+1, \perp\rangle.$$

All vertices in T not ending in $\perp$ are leaves; all vertices ending in $\perp$ are not.

A (root) *path* in T is a sequence $v_0=\perp, v_1, \ldots$ of vertices, perhaps infinite, such that for each n, v_{n+1} is an immediate successor of v_n. We let P denote the collection of all root paths which are either infinite or are finite and end in a leaf.

We want to prove that N^- is in bijective correspondence with P. In order to see this, we define two maps

$$\text{val}: \text{Ver} \to N$$

$$\text{path}: N \to P$$

where Ver is the set of vertices of T, as follows:

$$\text{val}\,\langle a_1, \ldots, a_{n-1}\rangle = \sum_1^{n-1} a_i i!;$$

$$\text{val}\,\langle a_1, \ldots, a_{n-1}, \perp\rangle = n!+\sum_1^{n-1} a_i i!.$$

Note that val $\langle\rangle=0$, where $\langle\rangle$ is the empty sequence.

Path is defined as follows: for each x in N, path (x) is the root path in P to the leaf vertex v, where val $v=x$. Note that if $x>0$, then $a_{n-1}>0$, and n is least number such that $x<n!$. (Of course, path$(0)=\langle\rangle$.) Lastly, if $v=\langle a_1, \ldots, a_{n-1}, \perp\rangle$, let $l(v)=n!$.

We note one important fact.

Fact. If there is a path from v to v' in T, then

$$\text{val}\,(v) \equiv \text{val}\,(v') \pmod{l(v)}.$$

We now define a function from L to P.

Definition of p: $L \to P$.

If $\langle z\rangle$ is an eventually constant sequence with value x in N, then $\langle z\rangle p=$path(x); if $\langle z\rangle$ is a strictly increasing sequence in L, then $\langle z\rangle p$ is the infinite root path

$$\perp, \quad v_1, \ldots, v_n, \ldots$$

where

$$v_n = \langle a_1, \ldots, a_{n-1}, \perp\rangle$$

$$\text{if} \quad z_n = n!+\Sigma a_i i!, \quad \text{for} \quad n>1.$$

Now we equip the set P of maximal root paths in the tree T with a topology as follows. For each vertex v of T, let $B(v)$ denote the set of all paths in P which contain v.

5.1. Definition. The topology on P is determined by taking the collection of sets $B(v)$, $v\in$Ver, as a basis.

Note that if v and v' are incomparable vertices, then $B(v)\cap B(v')$ is empty, and if $v<v'$, the intersection is $B(v')$.

Recall the bijection p: $L\to P$ above. The topology on L can be easily "seen" in P.

5.2. Theorem. p is a homeomorphism.

Proof. Let $B=B(v)$ be a basis set in P. If v is a leaf, $B(v)$ is a singleton and $p^{-1}(B)$ consists of the eventually constant sequence with eventual value val$\,(v)$.

Otherwise,

$$v = \langle a_1, \ldots, a_{n-1}, \perp\rangle \text{ so that}$$

$$p^{-1}(B) = \{\langle z\rangle \in L\colon\ z_n = n! + \Sigma a_i i!\} = p_n^{-1}(\text{val } v).$$

In either case, $p^{-1}(B)$ is a basis set in L. Thus, p is continuous. The argument that p^{-1} is also continuous is equally easy and is omitted.

The *frontier* of a subset A of a topological space is $A^- - A$. Using the above geometric picture of the topology on P, we may describe the frontiers of subsets A of N as follows. Each element x of A determines a finite path path (x) in P (ending in a leaf v with val $(v) = x$.) The collection of all the vertices in path (x) for $x \in A$, determines a subtree of T, say $T(A)$. The important fact about the tree representation is this: if we identify the elements of N with their images under path, we obtain

The second representation theorem. *The infinite paths in $T(A)$ are precisely the elements in the frontier of A.*

For example, if we want to find a set A whose frontier is only countably infinite, we might want $T(A)$ to "look like" the tree in figure 5.2a. To do this, one may define A as the set of all numbers of the form val $\langle 1, 2, \ldots, n, 0, \ldots, 0, 1\rangle$ (for $n, m > 0$, where there are m 0's).

Since we want to characterize those subsets of N whose frontier in L (or P or N^-) is uncountable, we will prove a theorem concerning those locally finite trees that have an uncountable number of infinite paths.

5.3. Definition. B_2 is the complete binary tree — i.e. each vertex in B_2 has exactly two immediate successors.

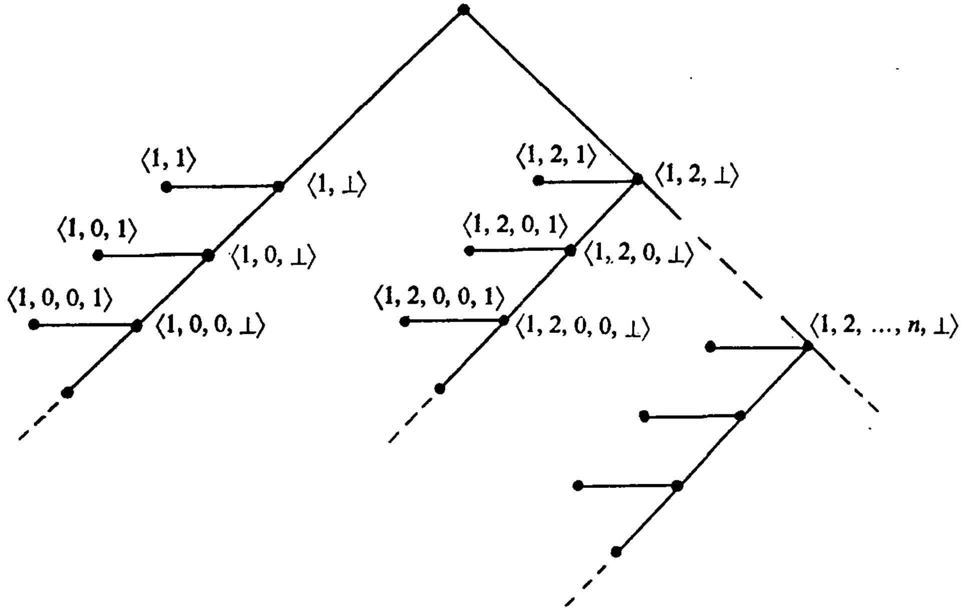


Fig. 5.2a

5.4. Definition. Let $T_i=(V_i, E_i)$, $i=1, 2$ be rooted trees, with V_i the set of vertices and E_i, the set of (ordered) edges. An *order embedding* $T_1 \to T_2$ is function $f\colon V_1 \to V_2$ such that for each pair $\langle v, v' \rangle$ of vertices in T_1, there is a path in T_1 from v to v' iff there is a path from vf to $v'f$ in T_2.

5.5. Theorem. Let T be a rooted locally finite tree. Then T has an uncountable number of infinite paths iff there is an order embedding of $B_2 \to T$.

Proof. Since clearly B_2 has uncountably many paths, it is easy to see that if there is an order embedding of B_2 in T, T has uncountably many paths as well. Now to prove the converse, we use the following fact

5.6. Lemma. Let T be a locally finite tree with uncountably many paths. Then there are two incomparable vertices v_1 and v_2 in T (i.e. it is not the case that $v_1<v_2$ or $v_2<v_1$) such that for $i=1, 2$, $T(v_i)$ has uncountably many paths, where $T(v_i)$ is the subtree of T consisting of v_i and all of its successors.

Proof of the Lemma. Since T is locally finite, for each n there are only finitely many vertices in T of depth n. For some n there must be two distinct vertices at depth n, say v_1 and v_2, such that the subtrees $T(v_1)$ and $T(v_2)$ of all descendents of v_1 and v_2 respectively both have uncountably many paths. Otherwise, T has only countably many paths, a contradiction.

Using this lemma, we can define an order embedding of B_2 in T, by induction on the depths of the vertices in B_2. The root of B_2 maps to the root of T. The two successors of the root of B_2 map to the first pair of incomparable vertices v_1 and v_2 in T such that $T(v_i)$, $i=1, 2$ has uncountably many paths. Having defined the embedding f on all vertices of B_2 of depth n such that for a vertex v in B_2, the tree $T(vf)$ has uncountably many paths, we use the lemma again to extend the definition of f one level further. The proof of the theorem is complete.

We may easily translate this result into an arithmetic form. For integers u and v, define

$$u \subseteqq v \quad \text{if} \quad u \leqq v \quad \text{and} \quad u \equiv v \pmod{l(u)};$$

i.e. if

$$u = n! + \sum a_i i!$$

and

$$v = m! + \sum b_j j!,$$

then $u \subseteqq v$ iff $n \leqq m$ and $a_i=b_i$ for $i<n$. Say that a subset W of N is "B_2-like" if for all x in W there are y, z in W such that $x \subseteqq y$ and $x \subseteqq z$ and y and z are $\subseteqq$-incomparable.

The translation of 5.5 is:

5.7. Proposition. A subset X of N has an uncountable frontier iff X contains a nonempty B_2-like subset.

We obtain the following number theoretic fact as a result.

5.8. Corollary. The set of squares contains a B_2-like subset.

6. Final Remarks

It appears that the cardinality of the frontier of a one letter language is not a useful tool for making distinctions among languages. One might then ask whether the consideration of the sequence of frontiers of a language $X \subset N$ will be more useful, where the sequence $X = X_0, X_1, \ldots$ is defined by

$$X_{n+1} = X_n^- - X_n;$$

i.e. the $n+1$-st set is the frontier of the n-th. However, it is easy to show that for every subset X of N, X_2, and hence X_n for $n > 2$, is empty.

In Scott's talk, several of the results given here were stated: the theorem in Section 3 that **F** is isomorphic to the product of the p-adic integers; the fact that N^- formed a compact, zero-dimensional Hausdorff space and our Corollary 2.6; most importantly he stated a version of the first representation theorem for the sequences in **F**.

The paper [B] has some results of a category-theoretic nature related to the theorem in Section 3. We have not made any use of the fact that N^- forms a free profinite monoid. The reader interested in other properties of N^- (and other free profinite monoids) may consult [B2] and [R].

Some of the results in Section 1 and 2 can be generalized to the case of the structure of convergent sequences of words in an arbitrary alphabet. The interesting problem of finding a concrete representation of the equivalence classes of these sequences is, as far as I know, still open. However, in the case that M is a finitely generated free *commutative* monoid, the monoid M^- is a finite powei of N^-, as Z. Esik observed.

Addendum

In a recent conversation, Scott suggested modifying the definition of the frontier of a subset S of N as follows. Instead of defining the frontier of S as the closure of S^- minus S, $S^- - S$, let:

$$\text{fron}\,(S) = S^- - \text{int}\,(S^-),$$

where "int" denotes the interior operator. This definition has the property that exactly the regular sets have an empty frontier, since $\text{fron}(S) = \emptyset$ iff the closure of S is clopen.

Does the cardinality of this "new frontier" give more information about the structure of the set? Not much. For example, when S is the set of squares, we have already shown that the closure of S is uncountable. The interior of the closure is empty, by the following observation.

Proposition.. Let S be an infinite subset of N. If C is the closure of S, $cl(S)$, $O = \text{int}\,(C) \setminus F$ is empty iff S contains no infinite regular subset. (Recall that F is the set of infinite elements of $cl(N)$, the closure of N.)

Proof. First suppose that S contains no infinite regular subset. If $x \in O$, there is a regular subset R of S such that $x \in cl(R)$, by the definition of the topology. Since R is finite, $cl(R) = R$, and x itself is a finite number. Now suppose that $\text{int}(C) \setminus F$ is empty but that R is an infinite regular subset of S. Then $cl(R)$ is a clopen subset of $\text{int}(C)$ and contains uncountably many elements of F.

Applying this proposition to the set Sq of squares, we see that $\text{int}\,(cl(Sq))$ contains only the elements of Sq itself, since, by the pumping lemma, Sq contains no infinite regular subset. Thus, the cardinality of $\text{fron}(Sq)$ is uncountable.

We now show how to construct, for any subset A of integers, a set $S=S(A)$ such that fron(S) is also uncountable, and such that A is Turing equivalent to S. (Thus, if A is nonrecursive, so is S.) In this construction, we make use of the representation of the elements of $cl(N)$ as paths in the tree T defined in section 5, as well as proposition 5.7. We define the vertices in a subtree $A(T)$ of T by induction: First assume that the vertex $\langle 0, 0, 0, \perp\rangle$ belongs to $A(T)$;
Now assume that for $4<n$, the set of vertices V_{n-1} of length $<n$ which belong to $A(T)$ have been defined. For each vertex $v \in V_{n-1}$ which is not a leaf, write v as

$$v = \langle a_1, \dots, a_{n-1}, \perp\rangle.$$

Then define:
$v1$ $(=\langle a_1, \dots, a_{n-1}, 1\rangle) \in V_n$ iff $n \in A$; furthermore, define, for $j=0$ and $j=1$
$vj\perp$ $(=\langle a_1, \dots, a_{n-1}, j, \perp\rangle) \in V_n$ iff $n \in A$; if n is not in A, then for $j=2$ and 3, $vj\perp \in V_n$.

Otherwise, vj and $vj\perp$ do not belong to V_n.
This completes the definition of the set of vertices of the tree $A(T)$.

Note that the tree B_2 may be order embedded in $T(A)$, so that by proposition 5.7 the set S determined by the leaves of $T(A)$ has an uncountable closure. Moreover, $T(A)$ does not contain all infinite paths containing a particular vertex, and hence S does not contain any infinite regular subset. Lastly, it is clear that A and S are Turing equivalent.

Acknowledgements

It is a pleasure to thank Dana Scott for introducing me to his convergent sequences and sending me some references, including the slides of this talk. Jess Wright, Ralph Tindell and Doug Troeger made several useful suggestions.

Abstract

In a talk titled "Infinite words" given in the spring of '83, Dana Scott introduced the notion of a convergent sequence of "words" (i.e. elements of a finitely generated free monoid). Scott stated a number of properties of the collection of (equivalence classes of) convergent sequences, some of which showed that this structure forms a free "profinite" monoid. The monoid of convergent sequences has a natural Stone space topology in which subsets of the free monoid have closures. Scott asked whether an investigation of the "frontiers" of languages would lead to a useful classification of languages. In this paper, an explicit description of the frontiers of all subsets of N, the one-generated free monoid, is obtained. It is shown that the cardinality of the frontier of a subset of N cannot distinguish regular from nonregular sets. A necessary and sufficient condition that a subset of N have an uncountable frontier is given.

DEPARTMENT OF PURE AND APPLIED MATHEMATICS
STEVENS INSTITUTE OF TECHNOLOGY
HOBOKEN, NJ 07030

CONSULTANT TO THE MATHEMATICAL SCIENCES DEPARTMENT
IBM WATSON RESEARCH CENTER
YORKTOWN HEIGHTS. NY 10598

References

[B] BANASCHEWSKI, B., On profinite universal algebras. *General topology and its relations to modern analysis and algebra II* (Preceedings 3rd Prague Topology Symposium 1971). Academia Prague 1972, 51—62.

[B2] BANASCHEWSKI, B., The Birkhoff Theorem for Varieties of Finite Algebras, manuscript 1980.

[EBT] ELGOT, C., BLOOM, S. L., TINDELL, R., The algebraic structure of rooted trees, *J. of Computer and System Science*, vol. 16, No. 3 (1978) 362—399.

[G] GRATZER, G., *Universal Algebra* (Van Nostrand, Princeton, Toronto, London; 1968).

[K] KUROSH, *The theory of groups*, translated by K. Hirsch, (Chesea, New York; 1960)

[R] REITERMAN, J., The Birkhoff theorem for finite algebras; *Algebra Universalis* 14 (1982) 1—10.

[S] SCOTT, D., Infinite words, slides of a lecture given during the spring of 1983.

(Received Febr. 22, 1984)

A multi-visit characterization of absolutely noncircular attribute grammars

By E. GOMBÁS and M. BARTHA

1. Introduction

Simple multi-visit attribute grammars were introduced in [1]. An attribute grammar is simple multi-visit if each nonterminal has a fixed visit-number associated with it such that, during attribute evaluation, the attributes of a node which have visit-number j are computed in the j-th visit to the node. Putting in one class the attributes having the same visit-number we get an ordered partition of the attributes of the nonterminal. Thus, a visit to a node is a sequence of actions consisting of (a) computing all the inherited attributes contained in the class corresponding to the visit, (b) making some visits to the sons of the node and (c) computing all the synthesized attributes of the class. This evaluation strategy can be implemented in a natural way translating visits to recursive procedures, where inherited and synthesized attributes correspond to input and output parameters, respectively. It was proved in [1] that the problem whether an attribute grammar is simple multi-visit is NP (time) — complete.

In [2] Kastens introduced a subclass of simple multi-visit attributed grammars and called them ordered. The problem whether an attribute grammar is ordered can already be decided in polynomial time, but the choise of this subclass seems rather heuristic. A somewhat larger subclass, which is still decidable in polynomial time was investigated in [3]. However, because of the NP-completeness mentioned above there is no reason to work out further improvements.

In this paper we investigate a class of attribute grammars that can be evaluated by a multi-visit type strategy associating a fixed set of visits with each nonterminal, but the order of these visits need not be predetermined. We call these grammars generalized simple multi-visit. It turns out that this class of attribute grammars coincides with the class of absolutely noncircular attribute grammars, for which a more complicated tree-walking evaluator was given in [4].

It is known that the problem whether an attribute grammar is absolutely noncircular is decidable in polynomial time. However we shall show that the problem whether an attribute grammar is generalized simple m-visit for a fixed m (even for $m=2$) is still NP-complete. This means that, instead of trying to minimize the number of visits, it is more useful to execute them in parallel if possible. In section 5 we describe two such parallel evaluation strategies.

It was shown in [1] that an attribute grammar is simple multi-visit iff it is l-ordered, i.e. for each nonterminal a linear order of its attributes exists such that the attributes of a node can always be evaluated in that order. Let us call an attribute grammar top-down controlled l-ordered if there exists a finite state deterministic top-down tree automaton such that the state in which the automaton reaches a node of a derivation tree determines the evaluation order of the attributes of the node. As it is to be expected, the class of absolutely noncircular attribute grammars coincides with the class of top-down controlled l-ordered attribute grammars, too.

2. Preliminaries

Since we are dealing with almost the same notions as those defined in [1] (e.g. computation sequences, visit sequences, etc.), we adopt the main terminology introduced in that paper.

An attribute grammar G consists of:

(i) A reduced context free grammar $G_0=(T, N, P, Z)$, where T and N denote the set of terminal and nonterminal symbols, respectively, P is the set of productions (syntactic rules) and $Z\in N$ is the start symbol. We shall denote nonterminal symbols always by capital letters, while terminal strings by small ones.

(ii) A set A of attributes, which is the union of two disjoint finite sets A_s and A_i. The elements of A_s and A_i are called synthesized and inherited attributes, respectively (shortly s- and i-attributes). A function v assigns each nonterminal F a set $I(F)$ of inherited and a set $S(F)$ of synthesized attributes, i.e. $v(F)=I(F)\cup S(F)$. An attribute $a\in v(F)$ will be referenced as $a(F)$. The start symbol Z has only one s-attribute, and it does not occur on the right-hand side of any production.

(iii) A set $V(a)$ of possible values for each attribute a.

(iv) A set r_p of semantic rules associated with each production p. If p is of the form $p\colon F_0\to w_0F_1\dots F_{n_p}w_{n_p}$, then a semantic rule of r_p is an equation: $a_0(F_{i_0})= =f(a_1(F_{i_1}), \dots, a_m(F_{i_m}))$, where $0\leqq i_j\leqq n_p$ and $f\colon V(a_1)\times\dots\times V(a_n)\to V(a_0)$ is a (recursive) function. This equation is interpreted by saying that attribute $a_0(F_{i_0})$ depends on attributes $a_1(F_{i_1}), \dots, a_m(F_{i_m})$ in p. We assume that G is in Bochmann normal from, i.e. the rules of r_p assign all and only the attributes in $S(F_0)$ and $I(F_j)$ $(j\geqq 1)$ using as argument only attributes in $I(F_0)$ and $S(F_j)$.

The production graph of $p\colon F_0\to w_0F_1\dots F_{n_p}w_{n_p}$ (denoted by $pg(p)$) has as nodes the disjoint union of v (F_i), $0\leqq i\leqq n_p$, and there is an edge from $a_1(F_{i_1})$ to $a_2(F_{i_2})$ iff $a_2(F_{i_2})$ depends on $a_1(F_{i_1})$ in p. For a derivation tree t of G_0 we get the derivation tree graph of t (denoted by $dtg(t)$) by pasting together the pg's of all the productions t consist of. G is noncircular if none of the derivation trees of G_0 has an oriented cycle in its dtg. In the sequel, if no confusion arises we identify a nonterminal node of a derivation tree with its label.

3. The generalized simple multi-visit property

To describe an attribute evaluation strategy we define computation sequences similar to those introduced in [1]. A computation sequence is a sequence of so called basic actions, where each basic action is either the evaluation of some i-attributes of a node, called entering the node, or the evaluation of some s-attributes of a node, called exiting the node. Thus, a basic action can be represented by a basic action symbol (ba-symbol) $i(n, A)$ or $s(n, A)$, where n denotes a nonterminal node in a derivation tree and A is a subset of $I(n)$ or $S(n)$, respectively. Our computation sequences, however, allow redundant computations, too.

Let $G=(G_0=(T, N, P, Z), A, \nu, \{V(a)|a\in A\}, \{r_p|p\in P\})$ be an attribute grammar, fixed in the rest of this section, and t a derivation tree of G_0. We always assume that a derivation tree is complete, i.e. its root is Z and its leaves are in T.

Definition 3.1. A computation sequence for t is a string h of ba-symbols, which satisfies the following four conditions.

(1) Start-end condition: the first and the last ba-symbols are $i(Z, \emptyset)$ and $s(Z, \emptyset)$.

(2) Sequentiality condition: for any two contiguous ba-symbols $x_1(n_1, A_1)$ $x_2(n_2, A_2)$ in h one of the following conditions holds.

(i) n_2 is a son of n_1 and $x_1=x_2=i$;

(ii) n_2 is the father of n_1 and $x_1=x_2=s$;

(iii) n_2 is a brother of n_1 and $x_1=s$ and $x_2=i$;

(iv) $n_2=n_1$ and $x_1\neq x_2$.

(3) Feasibility condition: For any production p consider an arbitrary occurrence of p in t, and let n_1 and n_2 be any such nonterminal nodes of this occurrence that $a_1(n_1)$ depends on $a_2(n_2)$ in p for some attributes a_1 and a_2. Then the first ba-symbol in h which contains $a_2(n_2)$ cannot precede the first such ba-symbol that contains $a_1(n_1)$.

(4) Completeness condition: For each node n of t, if $i(n, B_1), s(n, A_1), \dots$, $\dots, i(n, B_k), s(n, A_k)$ is the sequence of the ba-symbols of n occurring in h, then $\bigcup(\{A_i\cup B_i\}|i\in[k-1])$ is a partition of $\nu(n)$; furthermore $A_k\cup B_k=\emptyset$. The set $\Pi(n)=\bigcup(\{A_i\cup B_i\}|i\in[k])$ will be called the visit set of n and each $A_i\cup B_i$ $(i\in[k])$ a visit element. Since $\bigcup(\{A_i\cup B_i\}|i\in[k-1])$ is a partition, each visit element, except one is a nonvoid subset of $\nu(n)$. $v\in\Pi(n)$ will be referenced as $v(n)$ or $v(F)$, if F is the nonterminal label of n.

The (simple multi-visit) completeness condition in [1] required $\{A_i\cup B_i|i\in[k]\}$ to be an ordered partition of $\nu(n)$. The main point of our modification is that we allow making the same visit to a node several times.

Let n_0 be a node of t having sons $n_1, \dots, n_m$, $v=A\cup B\in\Pi(n_0)$ and h a computation sequence for t. A visit trace of $v(n_0)$ in h, denoted by $tr(v(n_0))$ is a substring of h beginning with $i(n_0, B)$ and ending with $s(n_0, A)$ such that there is no further ba-symbol of n_0 between these two ones. By the definition of a computation sequence $tr(v(n_0))=tr(v_1(n_{i_1}))\dots tr(v_l(n_{i_l}))$, where $v_1, \dots, v_l$ are certain visit elements of $n_{i_1}, \dots, n_{i_l}$, respectively. The sequence $v_1(n_{i_1})\dots v_l(n_{i_l})$ will be called a visit sequence of $v(n_0)$. (Note that l might be 0, and a visit element might have several visit traces and visit sequences so far.)

Let Π: $N\to\mathcal{P}(\mathcal{P}(A))$ be a function such that for every $F\in N$ $\Pi(F)=\pi\cup\{\emptyset\}$, where π is a partition of $\nu(F)$. Furthermore, let ϱ be a set of functions $\{\varrho_p|p\in P\}$ such that if p: $F_0\to w_0F_1\dots F_{n_p}w_{n_p}$, then ϱ_p assigns each $v\in\Pi(F_0)$ a sequence $v_1(F_{i_1})\dots v_l(F_{i_l})$ $(l\geqq 0)$, where $v_m\in\Pi(F_{i_m})$ $(m\in[l], i_m\in[n_p])$ and all the v_m-s are different. Π will be called a collection of visit sets for G and ϱ a visit description function for Π.

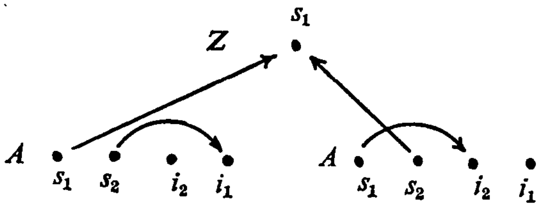
Definition 3.2. Given a collection Π of visit sets for G, a visit description function ϱ for Π and a derivation tree t, a computation sequence h for t respects ϱ if the following condition holds. For each nonterminal node n of t, if n is the left-hand side occurrence of a production p: $F_0 \to w_0 F_1 \dots F_{n_p} w_{n_p}$, then $\Pi(n) = \Pi(F_0)$ and each $v \in \Pi(n)$ has a unique visit sequence in h which is equal to $\varrho_p(v)$.

Note that there exists at most one computation sequence for t respecting ϱ. ϱ is called a generalized simple multi-visit description function for Π if each derivation tree of G has a computation sequence for it respecting ϱ.

Definition 3.3. G is a generalized simple multi-visit (gsmv) attribute grammar if there exist a collection Π of visit sets and a gsmv visit description function ϱ for Π. G is generalized simple m-visit ($m \in \mathbb{N}$) if $\|\Pi(F)\| \leqq m+1$ for each $F \in N$.

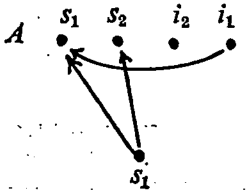
From this definition it is clear that the only essential difference between smv and gsmv evaluation strategies is that in the latter one each visit to a node must be made only if this has not been done before.

Example 3.4. Let Z, A, B and C be the nonterminals of the underlying grammar with the following attributes. $S(Z) = S(B) = \{s_1\}$, $I(Z) = I(B) = \emptyset$, $S(A) = \{s_1, s_2\}$, $I(A) = \{i_1, i_2\}$. The productions and the corresponding production graphs are listed below.



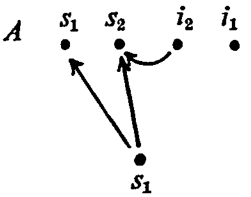
$1{:}Z \to AA$

a)



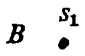
$2{:}A \to aB$

b)



$3{:}A \to bB$

c)



$4{:}B \to \lambda$

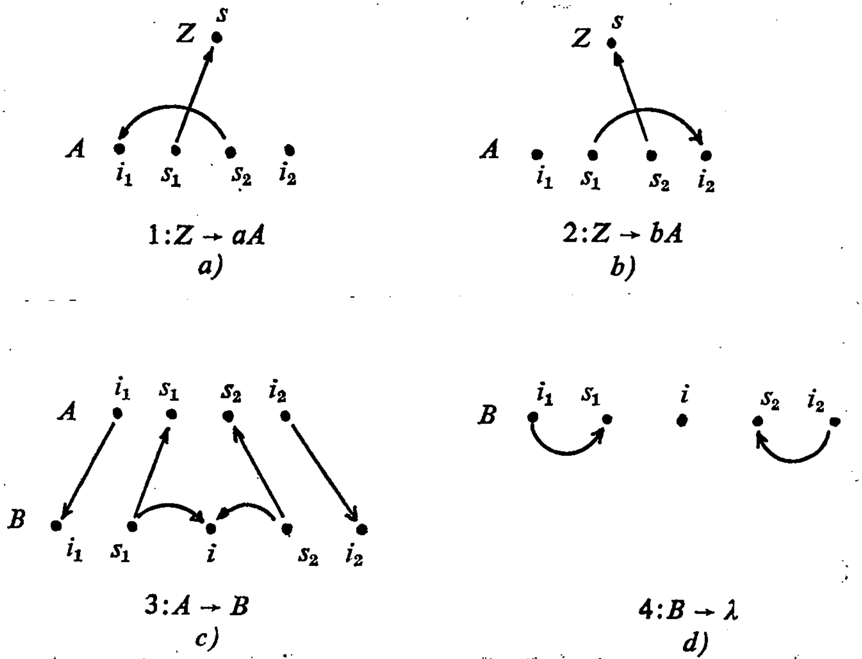
d)

If $\Pi(Z)=\Pi(B)=\{\{s_1\},\emptyset\}$ and $\Pi(A)=\{\{i_1,s_1\},\{i_2,s_2\},\emptyset\}$, then let $(\varrho_1(\{s_1\}=\{i_2,s_2\}(A_1)\{i_1,s_1\}(A_1)\{i_1,s_1\}(A_2)\{i_2,s_2\}(A_2)$, $\varrho_1(\emptyset)=\emptyset(A_1)\emptyset(A_2)$, $\varrho_2(\{i_1,s_1\})=\varrho_2(\{i_2,s_2\})=\{s_1\}(B)$, $\varrho_2(\emptyset)=\emptyset(B)$, $\varrho_3(\{i_1,s_1\})=\varrho_3(\{i_2,s_2\})=\{s_1\}(B)$, $\varrho_3(\emptyset)=\emptyset(B)$, $\varrho_4(\{s_1\})=\varrho_4(\emptyset)=\lambda$.

It is clear that ϱ is a gsmv description function for Π. However, G is not smv, because the visit elements of A cannot be evaluated in a fixed order. That is why we had to put the single visit element of B into both $\varrho_{2(3)}(\{i_1,s_1\})$ and $\varrho_{2(3)}(\{i_2,s_2\})$.

The following example shows that the void visit element is necessary.

Example 3.5. Let Z, A and B be the nonterminals of the underlying grammar with the following attributes $S(Z)=\{s\}$, $S(A)=S(B)=\{s_1,s_2\}$, $I(Z)=\emptyset$, $I(A)=\{i_1,i_2\}$, $I(B)=\{i,i_1,i_2\}$. The productions and the corresponding production graphs are listed below.



$1{:}Z\to aA$
a)

$2{:}Z\to bA$
b)

$3{:}A\to B$
c)

$4{:}B\to\lambda$
d)

G is clearly gsmv, but in the production $A\to B$ the useless attribute $i(B)$ can be evaluated only in the void visit of A.

To define the absolutely noncircular (anc) property we use the concept of induced production graph (ipg) introduced in [5]. These graphs can be obtained by adding some further edges to the production graphs. More exactly, considering a graph as a relation we get the ipg graphs by taking the least fixpoint of the following system of equations.

$ipg(p)=pg(p)\cup\{\langle a(F_i),b(F_i)\rangle|p\colon F_0\to w_0F_1\dots F_{n_p}w_{n_p}$, $i\in[n_p]$ and there is a $q\in P$ with left-hand side F_i such that $\langle a(F_i),b(F_i)\rangle\in ipg\,(q)^+\}$.

G is anc if for each $p \in P$ $ipg(p)$ is acyclic.

Let Π be a collection of visit sets for G. Π defines for every $F \in N$ a set $B(F)$ of nonvoid basic action symbols as follows. If $\Pi(F)=\{v_1, \ldots, v_n\}$, then $B(F)= =\{s(F, A_i) | i \in [n],\ v_i=A_i \cup B_i, B_i \subseteqq I(F), A_i \neq \emptyset\} \cup \{i(F, B_j) | j \in [n], v_j=A_j \cup B_j,\ A_j \subseteqq \subseteqq S(F),\ B_j \neq \emptyset\}$.

The production graph over the nonvoid ba-symbols of $p: F_0 \to w_0 F_1 \ldots F_{n_p} w_{n_p}$ (denoted by $pgb_\pi(p)$) is the following graph. The set of its nodes is the disjoint union of $B(F_i)$, $0 \leqq i \leqq n_p$, and there is an edge from $x_1(F_{i_1}, A_1)$ to $x_2(F_{i_2}, A_2)$ iff $a_2(F_{i_2})$ depends on $a_1(F_{i_1})$ in p for some $a_1 \in A_1$, $a_2 \in A_2$. From these graphs we construct the induced production graphs $\{ipgb_\pi(p) | p \in P\}$ as above, i.e. by taking the least fixpoint of the following system of equations.

$ipgb_\pi(p)=pgb_\pi(p) \cup \{\langle i(F_i, B), s(F_i, A)\rangle | 0 \leqq i \leqq n_p,\ A \neq \emptyset,\ B \neq \emptyset$ and $A \cup B \in \in \Pi(F_i)\} \cup \{\langle i(F_i, B_1, s(F_i\ , A_2)\rangle, \langle s(F_i, A_1), i(F_i, B_2)\rangle |$ none of A_i and B_i $(i=1, 2)$ is $\emptyset$, $\{A_i \cup B_i | i=1, 2\} \subseteqq \Pi(F_i)$ and there is a $g \in P$ with left-hand side F_i such that $\langle i(F_i, B_1), s(F_i, A_2)\rangle \in ipgb_\pi(q)^+\}$.

Remark 3.6. If for each $F \in N$ $\Pi(F)=\big\{\{a\} | a \in v(F)\big\} \cup \{\emptyset\}$, then $ipgb_\pi(p) \cong ipg(p)$ for every $p \in P$.

The following alogrithm can be used to compute the $ipgb_\pi$ relations.

Algorithm 3.7.

Step 1. For each $p \in P$ set $ipgb_0(p)=pgb_\pi(p) \cup \{\langle i(F, B), s(F, A)\rangle | F$ occurs in $p, A \neq \emptyset$, $B \neq \emptyset$ and ,$A \cup B \in \Pi(F)\}$

Step 2. If $j \geqq 0$, then for each $p \in P$ let $ipgb_{j+1}(p)=ipgb_j(p) \cup \{\langle i(F, B_1), s(F, A_2)\rangle, \langle s(F, A_1),\ i(F, B_2)\rangle |$ none of A_i, B_i $(i=1, 2)$ is $\emptyset$, $\{A_i \cup B_i | i=1, 2\} \subseteqq \subseteqq \Pi(F)$, F occurs on the right-hand side of p and on the left-hand side of such a $q \in P$ for which $\langle i(F, B_1),\ s(F, A_2)\rangle \in ipgb_j(q)^+\}$

Repeat step 2 until such a j is found for which $ipgb_j(p)=ipgb_{j+1}(p)$ for all $p \in P$. Then $ipgb_\pi(p)=ipgb_j(p)$.

It is easy to see that the time complexity of this algorithm is polynomial in the parameters of G.

Lemma 3.8. Given a collection Π of visit sets for G, there exists a gsmv description function ϱ for Π iff $ipgb_\pi(p)$ is acyclic for every $p \in P$.

Proof. (a) (ϱ exists$\Rightarrow ipgb_\pi$ is acyclic)

It is enough to prove that if $\langle x_1(F, A_1), x_2(F, A_2)\rangle \in ipgb_\pi(p) \setminus pgb_\pi(p)$, then for every derivation tree t and F-labelled node n_0 of t the following statement holds. If h is the computation sequence for t respecting ϱ, then the first occurrence of $x_1(n_0, A_1)$ precedes that of $x_2(n_0, A_1)$ in h. We shall prove this statement by an induction following algorithm 3.7. First we make the following observation.

If $A_1 \subseteqq v_0^1(n_0)$ and $A_2 \subseteqq v_0^2(n_0)$ for some $v_0^1, v_0^2 \in \Pi(n_0)$, then consider the sequence $n_0, \ldots, n_m$ $(m \geqq 0)$ of nodes and the sequences $v_0^1(n_0), \ldots, v_m^1(n_m)$ and $v_0^2(n_0), \ldots, v_m^2(n_m)$ of visit elements with the following properties.

1) For each $0 \leqq i < m$, n_{i+1} is the father of n_i and $v_{i+1}^j(n_{i+1})$ $(j=1, 2)$ are those visit elements for which the ba-symbols of $v_i^j(n_i)$ can be found in the visit trace corresponding to $v_{i+1}^j(n_{i+1})$ as first occurrences in h.

2) a) $v_m^1(n_m) = v_m^2(n_m) = v_m$ but $v_i^1(n_i) \neq v_i^2(n_i)$ if $i < m$, or

b) $n_m = Z$ and $v_m^1(Z) \neq v_m^2(Z)$.

It is important to note that the correct choice of $v_{i+1}^j(n_{i+1})$ in 1) can also be done by stepping upwards in t, following the nodes $n_{i+1}, \ldots, n_m, \ldots, Z$.

Now it is clear that the first occurrence of $x_1(n_0, A_1)$ precedes that of $x_2(n_0, A_2)$ in h iff one of the following three conditions holds.

(i) $m=0$;

(ii) $m \geqq 1$, $v_m^1 = v_m^2$ and v_{m-1}^1 precedes v_{m-1}^2 in $\varrho(v_m)$;

(iii) $n_m = Z$, $v_m^1 = \{a\}$, $v_m^2 = \emptyset$.

This means that the order of the first occurrence of $x_1(n_0, A_1)$ and $x_2(n_0, A_2)$ in h does not depend on the subtree of t below n_0.

As a basis of our induction let $\langle x_1(F, A_1), x_2(F, A_2)\rangle \in ipgb_0(p)$. The only possible case $x_1 = i$, $x_2 = s$ and $A_1 \cup A_2 \in \Pi(F)$ is trivial.

Now let $\langle x_1(F, A_1), x_2(F, A_2)\rangle \in ipgb_{j+1}(p) \setminus ipgb_j(p)$ $(j \geqq 0)$, and suppose that the statement holds for every appropriate $\langle x_1(F', A_1'), x_2(F', A_2')\rangle \in ipgb_j(p')$. Again, we can restrict ourselves to the case $x_1 = i$, $x_2 = s$ and $A_1 \cup A_2 \notin \Pi(F)$. Then F is a right-hand side nonterminal of p. If n_0 is an occurrence of a right-hand side nonterminal of some $r \in P$, then by construction $\langle i(F, A_1), s(F, A_2)\rangle \in ipgb_{j+1}(r)$, too. Let $q \in P$ such that the left-hand side of q is F and $\langle i(F, A_1), s(F, A_2)\rangle \in ipgb_j(q)^+$. Then there exists a sequence of nonvoid ba-symbols $x_0(F_0, B_0) \ldots x_l(F_l, B_l)$ such that $x_0(F_0, B_0) = i(F, A_1)$, $x_l(F_l, B_l) = s(F, A_2)$ and $\langle x_{i-1}(F_{i-1}, B_{i-1}), x_i(F_i, B_i)\rangle \in$ $\in ipgb_j(q)$ for each $i \in [l]$. Change the subtree below n_0 to an arbitrary subtree with top production q. If t' is the resulting subtree and h' is the computation sequence for t' respecting ϱ, then we have again that the order of the first occurrence of $i(F, A_1)$ and $s(F, A_2)$ is the same in h' as in h. On the other hand, the inductive hypothesis and the feasibility condition imply that for each $i \in [l]$ the first occurrence of $x_{i-1}(F_{i-1}, B_{i-1})$ precedes that of $x_i(F_i, B_i)$ in h', so we are through.

(b) ($ipgb_\pi$ is acyclic $\Rightarrow \varrho$ exists)

For $p\colon F_0 \to w_0 F_1 \ldots F_{n_p} w_{n_p} \in P$, $v_0 = A \cup B \in \Pi(F_0)$ $(A \subseteqq S(F_0), B \subseteqq I(F_0))$ we construct $\varrho(v_0)$ as follows. Let $s(F_{i_1}, A_1), \ldots, s(F_{i_k}, A_k)$, $i(F_{j_1}, B_1), \ldots, i(F_{j_m}, B_m)$ be all the nonvoid ba-symbols that can be reached on a reversed path from $s(F_0, A)$ in $ipgb_\pi(p)$ (provided it is a nonvoid ba-symbol). Then define $\varrho(v_0(F_0)) = v_1(F_{n_1}) \ldots$ $\ldots v_l(F_{n_l})$ so that

(i) $v_1(F_{n_1}), \ldots, v_l(F_{n_l})$ are all the visit elements of the right-hand side nonterminals such that $F_{n_t} = F_{i_r}$ and $v_i(F_{n_i}) \cap A_r(F_{i_r}) \neq \emptyset$ for some $r \in [k]$, or $F_{n_t} = F_{j_s}$ and $v_i(F_{n_i}) \cap B_s(F_{j_s}) \neq \emptyset$ for some $s \in [m]$;

(ii) if $1 \leqq i < j \leqq l$, $x_1(F_{n_i}, A_1)$ and $x_2(F_{n_j}, A_2)$ are nonvoid ba-symbols of $v_i(F_{n_i})$ and $v_j(F_{n_j})$, respectively, then $\langle x_2(F_{n_j}, A_2), x_0(F_{n_i}, A_1)\rangle \notin ipgb_\pi(p)^+$.

Those nonvoid visit elements of the right-hand side nonterminals that are not listed in (i) for any $v \in \Pi(F_0)$ are put into $\varrho(\emptyset(F_0))$ in an arbitrary order satisfying (ii). Finally, the tail of $\varrho(\emptyset(F_0))$ is $\emptyset(F_1) \ldots \emptyset(F_{n_p})$. It is easy to check that such a ϱ is indeed a gsmv description function for Π.

The proof of the following lemma is left to the reader.

Lemma 3.9. If Π is a collection of visit sets for G such that $ipgb_\pi(p)$ is acyclic for every $p\in P$, then G is anc.

Our main theorem is now an immediate consequence of lemmas 3.8, 3.9 and remark 3.6.

Theorem 3.10. G is gsmv iff it is anc.

4. The top-down controlled *l*-ordered property

A deterministic finite state derivation tree automaton (dfsdt) for a context free grammar $G_0=(N, T, P, Z)$ is a triple $\mathbf{Q}=(Q, \bar{q}, \{u_p|p\in P\})$, where Q is a finite set (the set of states), $\bar{q}\in Q$ is the initial state, and if p: $F_0\to w_0F_1...F_{n_p}w_{n_p}\in P$, then u_p is a set of so called transition rules of the form $q_0(F_0)\to q_1(F_1)...q_{n_p}(F_{n_p})$ $(q_i\in Q)$, where each $q\in Q$ occurs at most once on the left-hand side of these rules. $\mathbf{Q}$ functions in the following well-known way on an input derivation tree t of G_0. It starts by assigning state $\bar{q}$ to the root of t, then, if a nonterminal node n_0 of t has already been assigned a state q_0, it assigns states (if possible) to the nonterminal sons of n_0 as follows. If n_0 is an occurrence of the left-hand side of p: $F_0\to w_0F_1...F_{n_p}w_{n_p}$, $n_1, ..., n_{n_p}$ are the corresponding occurences of $F_1, ..., F_{n_p}$, respectively and $q_0(F_0)\to q_1(F_1)...q_{n_p}(F_{n_p})\in \in u_p$, then for each $i\in[n_p]$ $\mathbf{Q}$ assigns q_i to n_i. t is accepted by $\mathbf{Q}$ iff it is able to assign a state to each nonterminal node of it in the above way.

Given an attribute grammar G and a dfsdt $\mathbf{Q}$ for G_0, a $\mathbf{Q}$-defined l-ordering of G is a partial function O which assigns for some pairs $\langle q, F\rangle$ $(q\in Q, F\in N)$ a linear order of $v(F)$.

Definition 4.1. An attribute grammar G is top-down controlled l-ordered (tcl-ordered) iff there is a dfsdt $\mathbf{Q}$ for G_0 and a $\mathbf{Q}$-defined l-ordering O of G such that for every derivation tree t of G_0 there exists a linear order O_t of the nodes of dtg (t) with the following properties.

(i) if a depends on b in a production occurring in t, then for the corresponding occurrences of a and b in dtg (t) we have $a<b$ in O_t (feasibility);

(ii) $\mathbf{Q}$ accepts all the derivation trees of G_0 (completeness);

(iii) if n is an F-labelled node in t and $\mathbf{Q}$ assigns state q to n, then $O(q, F)$ is defined, and $a<b$ in $O(q, F)$ iff $a<b$ in O_t (O is respected).

Theorem 4.2. G is gsmv iff it is tcl-ordered.

Proof. We show the equivalence of anc and tcl-ordered properties.

(a) (tcl-ordered$\Rightarrow$anc)

It is enough to prove that for every derivation tree t, if n is an F-labelled node in t and $\mathbf{Q}$ assigns state q to n, then the following statement holds. If $\langle a(F), b(F)\rangle\in \in ipg\,(p)$ for some $p\in P$ with $a\in I(F)$ and $b\in S(F)$, then $a<b$ in $O(q, F)$. We omit the proof of this statement since it is analogous to that of lemma 3.8/(a).

(b) (anc$\Rightarrow$tcl-ordered)

Let $Q=\bigcup(O(F)\ F\in N)$, where $O(F)$ is the set of l-orderings of $v(F)$. We construct $\{r_p|p\in P\}$ and $H\subseteq Q\times N$ by the following procedure.

Step 1. Put $(\bar{q}, Z)$ into H, where $\bar{q}$ is the unique l-ordering of $v(Z)$.

Step 2. If $(q_0, F_0)\in H$ (q_0 is an l-ordering of $v(F_0)$), p: $F_0\to w_1F_1\dots F_{n_p}w_{n_p}$, then construct the graph $\overline{ipg}(p)=ipg(p)\cup\{(a(F_0), b(F_0))|a<b$ in $q_0\}$. For each $i\in[n_p]$ choose an arbitrary l-ordering q_i of $v(F_i)$ that extends the partial order $\overline{ipg}(p)^+|v(F_i)$. Then add $\{(q_i, F_i)|i\in[n_p]\}$ to H and $q_0(F_0)\to q_1(F_1)\dots q_{n_p}(F_{n_p})$ to r_p.

Repeat step 2 until no more new elements can be added to H.

It is easy to see that if G is anc, then the graph $\overline{ipg}(p)$ constructed in step 2 is always acyclic, so the automaton $\mathbf{Q}=(Q, \bar{q}, \{r_p|p\in P\})$ and the $\mathbf{Q}$-defined l-ordering $O=\{((q, F), q)|(q, F)\in H\}$ satisfy the requirements of definition 4.1.

5. Implementations of the gsmv strategy

1) Implementation using recursive procedures.

Let $G=(G_0=(T, N, P, Z), A, v, \{r_p|p\in P\})$ a gsmv attribute grammar, Π a collection of visit sets of G and ϱ a gsmv visit description function for Π. We assume that before starting attribute evaluation we are given a structure tree, the nodes of which are represented by suitable data structures (e.g. records) with components for the attributes, references to the sons, a rule indicator that indicates the production applied to the node and a boolean flag for each visit element of the node which is true iff the visit has already been executed. The initial value of these flags is false. Then for each p: $F_0\to w_0F_1\dots F_{n_p}\,w_{n_p}\in P$ and $v_0\in\Pi(F_0)$ we have a

```
procedure VISIT-$p$-$v_0(n_0)$; node $(n_0)$;
 comment $\varrho_p(v_0)=v_1(F_{m_1})\dots v_l(F_{m_l})$;
 begin
    compute all the $i$-attributes of $v_0$ at $n_0$
    for $k=1$ to $l$
 comment let $n_1, \dots, n_{n_p}$ be the sons of $n_0$ having
    labels $F_1, \dots, F_{n_p}$, respectively;
      begin
        if $\neg$ flag-$v_k\cdot n_{m_k}$ then
          begin
            flag-$v_k\cdot n_{m_k}=true$;
            case rule indicator$\cdot n_{m_k}$ of
 comment let $q_1, \dots, q_j$ be all the productions with left-hand side $F_{m_k}$;
            $q_1$: VISIT-$q_1$-$v_k(n_{m_k})$;
              ⋮
            $q_j$: VISIT-$q_j$-$v_k(n_{m_k})$;
            esac
          end;
      end;
    compute all the $s$-attributes of $v_0$ at $n_0$
 end
```

2) Implementation using a system of tasks with parameters.

Here we assume that each node n of the structure tree has the following binary semaphores beyond the components (exept the flags) defined so far

(i) binary semaphores $i(n, v)$ and $s(n, v)$ for each $v \in \Pi(n)$ such that $i(n, v)$ $s\big((n, v)\big)$ is *true* iff the evaluation of all the i-attributes (s-attributes, respectively) has already terminated. We do not need $i(n, v)$ $\big(s(n, v)\big)$ if there are no i-attributes (s-attributes, respectively) in $v(n)$.

(ii) a binary semaphore $x(n, v)$, which is *true* iff the evaluation of v has already begun.

The initial value of these semaphores is clearly *false*. We treat semaphores $i(n, v)$ and $s(n, v)$ by the following primitives:

```
wait (w): L: if $w=0$ then goto L
send (w): $w: =1$
```

Semaphores $x(n, v)$ will be treated by the indivisible Boolean procedure $TS(x)$ (cf. [6]) of the form:

```
Boolean procedure TS(x)
  Boolean x;
    begin
      TS:=x;
      x:=true;
    end
```

Now for each p: $F_0 \to w_0 F_1 \dots F_{n_p} w_{n_p}$ and $v \in \Pi(F_0)$ we have a task:

```
$T_{p,v}(n_0)$; node $n_0$;
  wait $\big(i(m_0, v_1^0)\big)$; ...; wait $\big(i(m_0, v_0^k)\big)$;
comment: $m_0$ is the father of $n_0$, it is a left-hand side occurrence of $q \in P$ and
    $\{i(m_0, v_0^i) | i \in [k]\}$ are all the semaphores such that some $a \in v$ depends on
    some $b \in v_0^i$ in $q$;
  wait $\big(s(m_{i_1}, v_{i_1}^1)\big)$; ...; wait $\big(s(m_{i_l}, v_{i_l}^l)\big)$;
comment: $m_1, \dots, m_{n_q}$ are all the sons of $m_0 (n_0 = m_j$ for some $j \in [n_q])$, $i_s \in [n_q]$
    for each $s \in [l]$ and $\{s(m_{i_s}, v_{i_s}) | s \in [l]\}$ are all the semaphores such that some
    $a \in v$ depends on some $b \in v_{i_s}^s$ in $q$;
  compute all the $i$-attributes of $v$;
  send $\big(i(n, v)\big)$;
  for $j=1$ to $i$
    begin
comment: $\varrho(v) = v_1(F_{r_1}) \dots v_i(F_{r_i})$, $n_1, \dots, n_{n_p}$ are the sons of $n_0$ such that $n_s$
    is a left-hand side occurrence of $q_s \in P$;

      if $\neg TS\big(x(n_{r_j}, v_j)\big)$ then activate $T_{q_{r_j}, v_j}(n_{r_j})$;
    end;
  wait $\big(s(n_{r_1}, v_1)\big)$; ...; wait $\big(s(n_{r_i}, v_i)\big)$;
  compute all the $s$-attributes of $v$;
  send $s(n_0, v)$
```

We omitted the two case statements that should be applied to find the rules q and $\{q_{r_j} | j \in [i]\}$.

3) Implementation using a task system so that overlaps are allowed among the visit sets.

For any nonterminal F choose a production p_F such that F occurs on the right-hand side of p_F. For each s-attribute $a \in v(F)$ let $v_a = \{a, b_1, ..., b_m\}$, where $\{b_i | i \in [m]\}$ are all the i-attributes of F such that a depends on b_i in $ipg(p_F)$. Clearly, v_a does not depend on the choice of p_F. Let $\Pi(F) = \{v_a | a \in S(F)\}$. Now we assume that each attribute $a \in v(n)$ has own semaphores $x(n, a)$ and $y(n, a)$ at node n of the structure tree. $y(n, a)$ is true iff the evaluation of a has terminated at node n. If $a \in A_i$, then $x(n, a)$ is true iff the evaluation of a has already begun at n, and if $a \in A_s$, then $x(n, a)$ is true iff the evaluation of v_a has already begun. It p: $F_0 \rightarrow w_0 F_1 ... F_{n_p} w_{n_p} \in P$ and $v_a \in \Pi(F_0)$, then we have the following task:

```
$T_{p, v_a}(n_0)$; node $(n_0)$;
    for $j=1$ to $m$
  comment: $v_a = \{a, b_1, ..., b_m\}$;
    if $\neg TS(x(n_0, b_j))$ then
      begin wait $(y(m_0, b^j))$; ...; wait $(y(m_0, b_k^j))$;
        wait $(y(m_{i_1}^{(j)}; a_1^j))$; ...; wait $(y(m_{i_l}^{(j)}, a_l^j))$;
  comment: $m_0$ is the father of $n_0$, it is a left-hand side occurrence of $q \in P$,
    $m_1, ..., m_{n_q}$ are the sons of $m_0$ and $\{b_i^j | i \in [k]\} \cup \{a_s^j | s \in [l]\}$ are all the attri-
    butes $b_j$ depends on in $q$;
        compute $b_j$;
        send $(y(n_0, b_j))$;
      end;
  comment: $a_1, ..., a_i$ aie all the $s$-attributes a depends on in $p, n_1, ..., n_{n_p}$ are
    the sons of $n_0$ such that $n_s$ is the left-hand side of $q_s \in P$, $a_j \in v(F_{r_j})$ for each
    $j \in [i]$;
    if $\neg TS(x(n_{r_j}, a_j))$ then activate $T_{q_{r_j}, a_j}$;
    wait $(y(n_{r_1}, a_1))$; ...; wait $(y(n_{r_i}, a_i))$;
    compute a;
    send $(y(n_0, a))$
```

6. NP-completeness

We have seen that the problem whether an attribute grammar is gsmv can be decided in polynomial time. However, we shall prove that it is NP-complete to decide whether an attribute grammar is generalized simple m-visit for a fixed $m \geqq 2$.

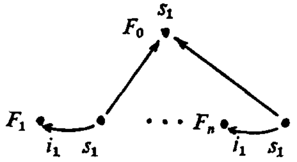
Theorem 6.1. The problem whether an attribute grammar is generalized simple 2 visit is NP-complete.

Proof: This problem is in NP, since we can guess an appropriate collection of visit sets Π by a nondeterministic algorithm in polynomial time and check whether $ipgb_\pi$ is acyclic using algorithm 3.7.

To prove NP-completeness we construct for every Boolean formula F_0 in conjunctive normal form an attribute grammar $G(F_0)$ such that F_0 is satisfiable iff $G(F_0)$ is gs-2-visit.

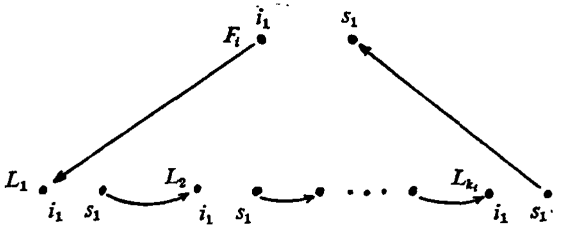
Let F_0 be a Boolean formula which is a product of sums of literals. A literal is either P or *not* P for some Boolean variable P. If $F_0 = F_i * ... * F_n$ and $F_i = L_1^i + ... + L_{k_i}^i$, $i \in [n]$, then in $G(F_0)$ we have a nonterminal P for each variable P, nonterminals F_1, ...,

..., F_n for each factor, nonterminals $L_1^i, ..., L_{k_i}^i$, $i\in[n]$ for each literal and nontermina, F_0, which is the start symbol. If P is a variable nonterminal, then $I(P)=\{i_1, i_2\}$l $S(P)=\{s, s_1, s_2\}$, F_0 has s_1 and all the other nonterminals have i_1 and s_1. The productions and the corresponding production graphs are the following.



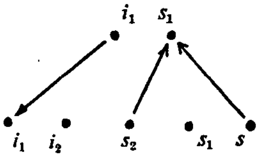
$F_0 \to F_1 * ... * F_n$

a)



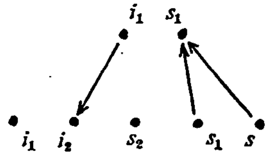
$F_i \to L_1 + ... + L_{k_i}$

b)



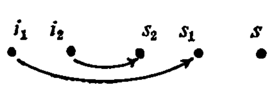
$L_j^i \to P$

c)



$L_j^i \to$ <u>not</u> P

d)

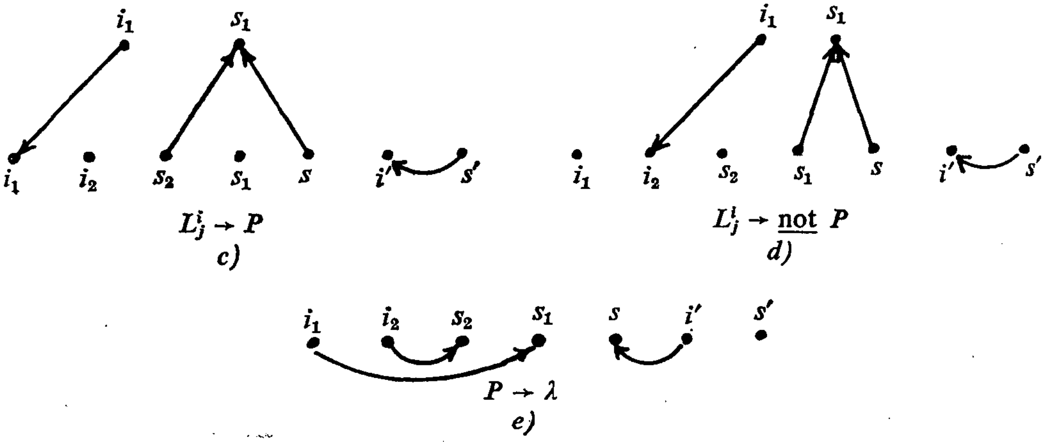


$P \to \lambda$

e)

Let $G'(F_0)$ be the attribute grammar which differs from $G(F_0)$ only in that $v(P)= =\{i_1, i_2, s_1, s_2\}$ for each Boolean variable P of F_0. The *pg*'s of $G'(F_0)$ are the restrictions of the *pg*'s of $G(F_0)$ to this set of attributes. It was shown in [1] (Theorem 4.1) that $G'(F_0)$ is simple multi visit iff F_0 is satisfiable. On the other hand $G(F_0)$ is *gs*-2 visit iff $G'(F_0)$ is simple multi visit. Indeed, if $G'(F_0)$ is smv, then we can fix $\Pi(P)= =\{\{i_1, s_1, s\}, \{i_2, s_2\}\}$ or $\Pi(P)=\{\{i_1, s_1\}, \{i_2, s_2, s\}\}$ for each variable P. If $G'(F_0)$ is not smv, then there are at least two occurrences of a variable P in the derivation tree of $G'(F_0)$ such that the visits $\{i_1, s_1\}$ and $\{i_2, s_2\}$ of P must be evaluated in different order at these occurrences. Hence s cannot be put into any of these two visit sets, it must be evaluated in a third visit.

To show NP-completeness for $m>2$ we only have to define $m-2$ preliminary visits to the variable nonterminals so that e.g. for $m=3$ the last three *pg*-graphs of the previons figure should be



$L_j^i \to P$
c)

$L_j^i \to$ not P
d)

$P \to \lambda$
e)

Abstract

The concept of smv attribute grammar is generalized by using redundant computation sequences in the specification of visit sets and visit sequences. It is proved that these gsmv attribute grammars are the same as the absolutely noncircular ones.

An attribute grammar is top-down controlled *l*-ordered if there is a deterministic finite state top-down tree automaton such that for every node of every derivation tree, the order in which the attributes of the node must be evaluated is determined by the state in which the automaton passes through the node. It is proved that an attribute grammar is generalized smv iff it is top-down controlled *l*-ordered.

Finally it ls shown that the problem, whether an attribute grammar is gs *m*-visit for a fixed $m \geqq 2$ is NP-complete.

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

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

References

[1] ENGELFRIET, J. and FILE, G., Simple multi-visit attribute grammars, Journal of Computer and System Sciences, v. 24, 1982, pp. 283—314.

[2] KASTENS, U., Ordered attribute grammars, Acta Informatica, v. 13, 1980, pp. 229—256.

[3] GYIMÓTHY, T., SIMON, E. and MAKAY, A., An implementation of the HLP, Acta Cybernetica, v. 6, 1983.

[4] KENNEDY, K. and WARREN, S. K., Automatic generation of efficient evaluators for attribute grammars, Conf. Record of the Third ACM Symp. on Principles of Programming Languages, 1976, pp. 32—49.

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

[6] SHAW, A. C., The logical design of operating systems, Prentice-Hall, Inc., 1974.

(Received Nov. 17, 1983)

Indexed LL(k) Grammars

R. Parchmann, J. Duske, J. Specht

1. Introduction

In the literature a number of extensions of context-free languages have been proposed. The motivation for this is to describe certain constructs of programming languages which are not context-free. An important class of such an extension are the indexed languages introduced by Aho [1].

In the area of context-free languages, the LL(k) languages are of special interest. In [10] and [11] proposals have been made to generalize this notion to the indexed case.

Indexed languages coincide with the IO-macro languages introduced in [2]. In [6], Mehlhorn defined the notion of a strong LL(k) macro grammar and investigated this class of grammars. Furthermore he defined the notion of a general LL(k) macro grammar and stated the following problems for these classes of grammars:

(a) Is the class of languages defined by the strong LL(k) condition equal to the class of languages defined by the general LL(k) condition?

(b) Is the general LL(k) condition decidable for a given k?

In [10], Sebesta and Jones gave a positive answer to question (b) for indexed grammars without e-productions in the case $k=1$.

In this paper we will answer completely these two questions for indexed LL(k) grammars.

In Section 2, basic notions and definitions will be given and compared with those introduced in [10] and [11].

In Section 3 it will be proved that the strong indexed LL(k) property is decidable for a given k.

In Section 4 we will show that the strong indexed LL(k) languages are deterministic indexed languages which were introduced in [7]. In Section 5 deterministic context-free languages will be characterized as right linear strong indexed LL(1) languages.

In Section 7 the main result of this paper, namely the decidability of the general indexed LL(k) property will be proved. This will be shown by using a general transformation on indexed grammars given in Section 6. This transformation converts an arbitrary indexed grammar into an equivalent grammar which is a strong indexed LL(k) grammar iff the original one is a general indexed LL(k) grammar. This answers the question (a) posed above. The decidability of the strong indexed LL(k) property then implies the decidability of the general indexed LL(k) property, which answers question (b).

2. Definitions of indexed LL(k) grammars

In this section we will consider subclasses of indexed grammars. Aho [1] introduced indexed grammars and languages. We will state these notions in the following form:

Definition 2.1. An *indexed grammar* is a 5-tuple $G=(N, T, I, P, S)$, where

(1) N, T, I are finite pairwise disjoint sets; the sets of *variables, terminals,* and *indices,* respectively.

(2) P is a finite set of pairs (Af, α), $A\in N$, $f\in I\cup\{e\}$, $\alpha\in(NI^*\cup T)^*$, the set of *productions.* (Af, α) is denoted by $Af\to\alpha$.

(3) $S\in N$, the *start variable.*

Let $\alpha=u_1B_1\beta_1u_2B_2\beta_2\ldots B_k\beta_ku_{k+1}$ with $u_i\in T^*$ for $i\in[1\colon k+1]$, and $B_j\in N$, $\beta_j\in I^*$ for $j\in[1\colon k]$ with $k\geqq 0$ be an element of $(NI^*\cup T)^*$ and let $\gamma\in I^*$. Then we set

$$\alpha:\gamma = u_1B_1\beta_1\gamma u_2B_2\beta_2\gamma \ldots B_k\beta_k\gamma u_{k+1}.$$

For $u, v\in(NI^*\cup T)^*$ we set $u\Rightarrow v$ iff $u=\varphi_1Af\gamma\varphi_2$, $v=\varphi_1(\alpha:\gamma)\varphi_2$ with $\varphi_1, \varphi_2\in$ $\in(NI^*\cup T)^*$ and $Af\to\alpha\in P$. $\overset{n}{\Rightarrow}$ is the n-fold product and $\overset{*}{\Rightarrow}$ is the reflexive, transitive closure of $\Rightarrow$.

The language $L(G)$ generated by an indexed grammar $G=(N, T, I, P, S)$ is the set $L(G)=\{w\mid w\in T^*$ and $S\overset{*}{\Rightarrow}w\}$. A language L is called an *indexed language* iff $L=L(G)$ for an indexed grammar G.

The subclasses of indexed grammars considered in this paper are the indexed LL(k) — and strong indexed LL(k) grammars, whose definitions are generalizations of the corresponding context-free notions. Furthermore we will compare these definitions with the corresponding definitions introduced in [10] and [11]. First we will introduce some basic notions.

Let Σ be an alphabet and let $k\geqq 1$ be an integer. ${}^{(k)}\Sigma^*$ denotes the set of all words w over Σ with $|w|\leqq k$, where $|w|$ denotes the length of w. The function ${}^{(k)}\colon \Sigma^*\to{}^{(k)}\Sigma^*$ is the identity on ${}^{(k)}\Sigma^*$ and assigns to each $w\in\Sigma^*$ with $|w|>k$ the prefix of w of length k.

Now let $G=(N, T, I, P, S)$ be an indexed grammar. Let π: $Af\to\beta$ be a production, let $k\geqq 1$, and let $\gamma\in I^*$. Then we set

$$\mathrm{First}_k(\pi) = \{{}^{(k)}u\mid S\overset{*}{\Rightarrow} wA\alpha \overset{\pi}{\Rightarrow} w\theta \overset{*}{\Rightarrow} wu\},$$

$$\mathrm{First}_k(\pi, \gamma) = \{{}^{(k)}u\mid A\gamma \overset{\pi}{\Rightarrow} \theta \overset{*}{\Rightarrow} u\},$$

where $w, u\in T^*$, and where $\overset{*}{\Rightarrow}$ and $\overset{\pi}{\Rightarrow}$ are leftmost derivations. Furthermore we set for $\theta\in(NI^*\cup T)^*$

$$\mathrm{First}_k(\theta) = \{{}^{(k)}u\mid \theta \overset{*}{\Rightarrow} u\} \text{ with } u\in T^*.$$

From now on, all derivations are assumed to be leftmost derivations.

Now we define the notion of an indexed LL(k) grammar (ILL(k) grammar). This notion corresponds to the context-free LL(k) grammars.

Definition 2.2. Let $G=(N, T, I, P, S)$ be an indexed grammar and let $k \geqq 1$ be an integer. G is called an *ILL*(k) *grammar* if the following holds:
Let

$$S \overset{*}{\Rightarrow} wA\gamma\alpha \overset{\pi}{\Rightarrow} w\theta_1 \overset{*}{\Rightarrow} wx \quad \text{and}$$

$$S \overset{*}{\Rightarrow} wA\gamma\alpha \overset{\pi'}{\Rightarrow} w\theta_2 \overset{*}{\Rightarrow} wy$$

be two leftmost derivations with $A \in N, \gamma \in I^*$, $\alpha \in (NI^* \cup T)^*$, and $w, x, y \in T^*$. Then ${}^{(k)}x = {}^{(k)}y$ implies $\pi = \pi'$.

(*Note:* $I=\emptyset$ yields the context-free LL(k) grammars.)

Remark. Let G be an ILL(k) grammar. Then for each word $w \in L(G)$ there is exactly one leftmost derivation according to G.

Example 2.1. Consider the indexed grammar $G=(N, T, I, P, S)$ with $N = \{S, A, B, C\}$, $T=\{a, b, c\}$ and $I=\{f, g\}$. The productions in P are π_1: $S \to aAf$, π_2: $S \to bAg$, π_3: $A \to B$, π_4: $A \to C$, π_5: $Bf \to a$, π_6: $Cf \to b$, π_7: $Cg \to a$, and π_8: $Bg \to c$. Only the following derivations are possible:

$$S \Rightarrow aAf \Rightarrow aBf \Rightarrow aa,$$

$$S \Rightarrow aAf \Rightarrow aCf \Rightarrow ab,$$

$$S \Rightarrow bAg \Rightarrow bBg \Rightarrow bc,$$

$$S \Rightarrow bAg \Rightarrow bCg \Rightarrow ba.$$

Obviously G is an ILL(l) grammar.

As for the context-free case it is possible to define the notion of a strong ILL(k) grammar.

Definition 2.3. Let $G=(N, T, I, P, S)$ be an indexed grammar and let $k \geqq 1$ be an integer. G is called a *strong ILL*(k) *grammar* if $\text{First}_k(\pi) \cap \text{First}_k(\pi') = \emptyset$ holds for all productions $\pi \neq \pi'$ which possess the same lefthand side or are of the form π: $A \to \alpha$, π': $Af \to \alpha'$ with $f \in I$.

(*Note:* $I=\emptyset$ yields the context-free strong LL(k) grammars.)

Remark. It is easy to see that strong ILL(k) grammars are ILL(k) grammars. The ILL(1) grammar G of Example 2.1 is not a strong ILL(1) grammar because $\text{First}_1(\pi_3)=\{a, c\}$ and $\text{First}_1(\pi_4)=\{a, b\}$. This shows that an ILL(k) grammar is not necessarily a strong ILL(k) grammar even for $k=1$. This differs from the context free case.

We can state:

Theorem 2.1. 1) A strong ILL(k) grammar is an ILL(k) grammar. 2) An ILL(1) grammar is not necessarily a strong ILL(1) grammar.

We will call a language a *(strong) ILL(k) language* if there exists a (strong) ILL(k) grammar generating this language.

We will now compare our definition of ILL(k) — and strong ILL(k) grammars with the definitions of indexed LL2(k) — and indexed LL1(k) grammars introduced in [10].

Obviously, an ILL(k) grammar is an indexed LL2(k) grammar. On the other hand consider the indexed grammar given by the productions π_1: $S \to Af$, π_2: $A \to a$, and π_3: $Af \to a$. The two possible leftmost derivations

$$S \overset{\pi_1}{\Longrightarrow} Af \overset{\pi_2}{\Longrightarrow} a,$$

$$S \overset{\pi_1}{\Longrightarrow} Af \overset{\pi_3}{\Longrightarrow} a$$

of the same word a according to this grammar show that it is not an ILL(1) grammar, but the LL2(1) condition is still satisfied.

The notion of a strong ILL(k) grammar and that of an LL1(k) grammar are incomparable.

The indexed grammar G_1 given by the productions π_1: $S \to aAf$, π_2: $S \to bAg$, π_3: $Af \to ab$, and π_4: $Ag \to ac$ is obviously a strong ILL(1) grammar.

G_1 is not an indexed LL1(1) grammar, since BASE(G_1) (see [10]) is not a context-free strong LL(1) grammar. This stems from the fact that in leftmost derivations according to G_1, applicability of π_3 excludes applicability of π_4 and vice versa. On the other hand consider the indexed grammar G_2 given by the productions π_1: $S \to Af$, π_2: $S \to Ag$, and π_3: $A \to a$ in P. G_2 is not a strong ILL(1) grammar, since $\mathrm{First}_1(\pi_1) \cap \mathrm{First}_1(\pi_2) = \{a\}$. G_2 obviously is an indexed LL1(1) grammar.

Furthermore, G_2 is not an indexed LL2(1) grammar, which shows that Theorem 4 in [10] is false.

In [11], three types of indexed LL(k) grammars, the α, β, γ-ILL(k) grammars were introduced. It is easy to see that the definitions of α-ILL(k)- and ILL(k) grammars and those of β-ILL(k)- and strong ILL(k) grammars coincide. These notions are defined but are not investigated further in [11], where only γ-ILL(k) grammars are investigated. These grammars do not even include all context-free LL(k) grammars.

3. Properties of strong ILL(k) grammars and languages

In this section we will first show that, given an indexed grammar G and a production $\pi \in P$, the language $\mathrm{First}_k(\pi)$ can be given effectively. This result implies that it is decidable whether a given indexed grammar is a strong ILL(k) grammar for a given k. Furthermore, we will call an indexed grammar "reduced", if each production occurs in at least one derivation of a terminal word. With the aid of $\mathrm{First}_k(\pi)$, we can construct, given an indexed grammar G, an equivalent reduced indexed grammar.

Theorem 3.1. Let an indexed grammar $G=(N, T, I, P, S)$, a production $\pi \in P$, and an integer $k \geqq 1$ be given. Then an indexed grammar G'' with $L(G'')= =\mathrm{First}_k(\pi)$ can be constructed effectively.

Proof. Construct the indexed grammar $G'=(N, T', I, P', S)$ with $T'= =T \cup \{\#\}$, and $P'=P \cup \{\pi'\}$ where π': $Af \to \#\alpha$ if π: $Af \to \alpha$. It is easy to construct a finite transducer M with $M(L(G'))=\mathrm{First}_k(\pi)$. Here we use the notion "finite transducer" as given in [1].

From Theorem 3.1 and Lemma 3.2 in [1] it follows that we can construct effectively an indexed grammar G'' with $L(G'')=M(L(G'))=\text{First}_k(\pi)$.

Now we can state

Corollary 3.1. Let $G=(N, T, I, P, S)$ be an indexed grammar, π a production of G, $k\geqq 1$ an integer, $\gamma\in I^*$, and $\theta\in(NI^*\cup T)^*$. Given a $v\in{}^{(k)}T^*$ it is decidable whether

(1) $v\in\text{First}_k(\pi)$, (2) $v\in\text{First}_k(\theta)$, (3) $v\in\text{First}_k(\pi, \gamma)$ holds.

Proof. (1) Construct an indexed grammar G'' with $L(G'')=\text{First}_k(\pi)$. In [5] it is shown that the membership problem for indexed languages is decidable.

(2) With the aid of G construct the indexed grammar $G'=(N\cup\{S'\}, T, I, P', S')$ where S' is a new start variable and $P'=P\cup\{\pi'\}$ where π': $S'\to\theta$ is a new production. Then we have $\text{First}_k(\theta)=\text{First}_k(\pi')$.

(3) Let Af be the lefthand side of the production π. If π cannot be applied to $A\gamma$, then we have $\text{First}_k(\pi, \gamma)=\emptyset$. If $A\gamma \overset{\pi}{\Rightarrow} \hat{\theta}$, then $\text{First}_k(\pi, \gamma)=\text{First}_k(\hat{\theta})$ holds.

Corollary 3.2. Let $G=(N, T, I, P, S)$ be an indexed grammar and $k\geqq 1$ be an integer. It is decidable whether G is a strong ILL(k) grammar.

Since the language $\text{First}_k(\pi)$ can be given effectively for an indexed grammar G, it is possible to single out all productions of G which never appear in derivations of terminal words. We will call an indexed grammar without such productions "reduced".

Definition 3.1. An indexed grammar $G=(N, T, I, P, S)$ with $L(G)\neq\emptyset$ is called *reduced* if for each $\pi\in P$ there exists a derivation of a terminal word in which π is applied.

Theorem 3.2. Let $G=(N, T, I, P, S)$ be an indexed grammar with $L(G)\neq\emptyset$. Then it is possible to construct a reduced indexed grammar $G'=(N, T, I, P', S)$ which is equivalent to G.

Proof. Determine for each production π the language $\text{First}_1(\pi)$. If $\text{First}_1(\pi)=\emptyset$ then remove the production. The grammar G' obtained in this way is reduced and obviously $L(G)=L(G')$ holds.

4. Strong ILL(k) languages are deterministic indexed languages

In [7] an indexed pushdown automaton (IPDA) has been defined, and it has been shown that these automata accept exactly the indexed languages.

Furthermore, a deterministic IPDA (d-IPDA) has been introduced in [7]. The class of languages accepted by these automata is called the class of deterministic indexed languages (DIL's). This class has properties similar to those of the class of deterministic context-free languages [7, 8]. In this section we will show that the strong ILL(k) languages form a subclass of the DIL's.

Theorem 4.1. If $G=(N, T, I, P, S)$ is a strong ILL(k) grammar then $L(G)$ is a deterministic indexed language.

Proof. We will construct a d-IPDA K which accepts the language $L(G)\$^k$ where \$ is an endmarker not in T. Since the deterministic indexed languages are closed under right quotient with regular sets [8], the language $L(G)$ is a DIL.

The states of K are the contents of a buffer of length k. This buffer contains a lookahead of k input symbols. Furthermore, the automaton simulates leftmost derivations according to G.

Set $K=(Z, X, \Gamma_1, \Gamma_2, \delta, z_0, A_0, g_0, F)$ with $Z={}^{(k)}X^* \cup \{z_f\}$, $X=T\cup\{\$\}$, $\Gamma_1=N\cup T\cup\{A_0\}$, where A_0 is a new element, $\Gamma_2=I$, $z_0=g_0=e$, $F=\{z_f\}$, and δ will be defined as follows:

(1) For all $u\in X^*$ with $|u|\leqq k-2$ and $a\in X$ set

$$(ua, (A_0, e))\in\delta(u, a, (A_0, e)).$$

For all $u\in X^{(k-1)}$, $a\in X$ set

$$(ua, (S, e)(A_0, e))\in\delta(u, a, (A_0, e)).$$

(2) Let π: $Af\to B_1\gamma_1\ldots B_r\gamma_r$ be a production of G, $r\geqq 0$, $B_i\in N\cup T$, and $\gamma_i\in I^*$ for $i\in[1: r]$. Then set $(v, (B_1, \gamma_1)\ldots(B_r, \gamma_r))\in\delta(v, e, (A, f))$ if $|v|=k$ and $\bar{v}\in\mathrm{First}_k(\pi)$, where $\bar{v}$ is the maximal prefix of v with $\bar{v}\in T^*$.

(3) For all $b\in X$, $a\in T$, and $u\in X^{k-1}$ set $(ub, e)\in\delta(au, b, (a, e))$.

(4) $(z_f, e)\in\delta(\$^k, e, (A_0, e))$.

Obviously, we have $L(K)=L(G)\$^k$. Since G is a strong ILL(k) grammar, we have $|\delta(z, x, (B, g))|\leqq 1$ for all $z\in Z, x\in X\cup\{e\}$, $B\in\Gamma_1$, and $g\in I\cup\{e\}$. (For example: $\delta(v, e, (A, e))$ with $A\in N$ can only be defined in (2). If $|\delta(v, e, (A, e))|>1$ we have a contradiction to the strong ILL(k) condition.)

It is easy to see that in each configuration (z, w, θ) of K at most one move is possible. (For example: $\delta(v, e, (A, e))\neq\emptyset$ and $\delta(v, e, (A, f))\neq\emptyset$ in (2) for $A\in N$ and $f\in I$ leads to a contradiction to the strong ILL(k) condition.)

Therefore K is a deterministic IPDA.

5. Strong ILL(k) languages and deterministic context-free languages

Theorem 4.1. shows that the class of strong ILL(k) languages is contained in the class of DIL's. The DIL's include all deterministic context-free languages, which we will now characterize as a special class of strong ILL(1) languages.

Theorem 5.1. For each deterministic context-free language L there exists a strong ILL(1) grammar G with $L=L(G)$.

Proof. Choose a deterministic pda $K=(Z, T, \Gamma, \delta, z_0, A_0, F)$ with $L=L(K)$. We may assume that in a final state, K may make no e-move (see [4], p. 239). Now construct the following indexed grammar $G=(N, T, I, P, S)$ with $N=Z\cup\{S\}$ and $I=\Gamma$. The productions of G will be defined as follows:

1) $S\to z_0A_0$ is in P

2) If $\delta(z, a, A)=(z', B_1\ldots B_r)$, then the production $zA\to az'B_1\ldots B_r$ is in P.

3) For each $z\in F$ the production $z\to e$ is in P.

Obviously, G is a strong ILL(1) grammar generating L.

The productions of the indexed grammar G in the foregoing proof are of a special "right linear" form. Let us define:

Definition 5.1. An indexed grammar $G=(N, T, I, P, S)$ is called a *right linear indexed grammar,* if each production in P has one of the forms $Af \to aB\gamma$ or $Af \to a$ with $A, B \in N$, $f \in I \cup \{e\}$, $a \in T \cup \{e\}$, and $\gamma \in I^*$.

Recall that an indexed grammar $G=(N, T, I, P, S)$ is called an *RIR grammar* (right linear indexed right linear) if all productions in P are of one of the forms $Af \to aB$, $Af \to a$, or $A \to aBf$, where $A, B \in N$, $a \in T \cup \{e\}$, and $f \in I \cup \{e\}$.

RIR grammars generate exactly the context-free languages. (see [1]).

Obviously, each RIR grammar is a right linear indexed grammar. On the other hand, it is easy to show that for each right linear indexed grammar there is an equivalent RIR grammar.

Therefore we can state:

Theorem 5.2. Right linear indexed grammars generate exactly the context-free languages.

Now we can state:

Corollary 5.1. Each deterministic context-free language is generated by a right linear strong ILL(1) grammar.

To prove the converse of this statement, we first need the following lemma.

Lemma 5.1. For each right linear indexed grammar $G=(N, T, I, P, S)$ there exists an equivalent right linear indexed grammar G' with the following properties:

1) There is exactly one start production.
2) All the other productions are of the form $Af \to \alpha$ with $f \neq e$.
3) If G is a strong ILL(k) grammar, then G' is a strong ILL(k) grammar.

Proof. Set $G'=(N', T, I', P', S')$ with $N'=N \cup \{S'\}$, $I'=I \cup \{\#\}$ and $P'=\{S' \to S\#\} \cup P''$, where P'' is defined as follows:

a) If $Af \to \alpha \in P$, $f \neq e$, then $Af \to \alpha \in P''$.
b) If $A \to \alpha \in P$, then $Ag \to \alpha$: $g \in P''$ for all $g \in I'$.

Obviously, G' is a right linear indexed grammar which satisfies 1) and 2), and is equivalent to G. Furthermore, it is easy to see that if G is a strong ILL(k) grammar, then G' is a strong ILL(k) grammar too.

Now we can prove:

Theorem 5.3. Each right linear indexed grammar $G=(N, T, I, P, S)$ which is a strong ILL(1) grammar, generates a deterministic context-free language.

Proof. If $L(G)=\emptyset$ then $L(G)$ is a deterministic context-free language. If $L(G)\neq\emptyset$, construct $G'=(N', T, I', P', S')$ according to the proof of Lemma 5.1. Furthermore we can assume w.l.o.g. that G' is reduced. We will define a pda K which accepts the language $L(G')\$$, where \$ is a new symbol. K buffers a lookahead of length one in its states and simulates leftmost derivations according to G'. The strong ILL(1) property of G' then implies that K is a deterministic pda.

Set $K=(Z, X, \Gamma, \delta, z_0, \#, F)$ with $Z=N'\times(T \cup \{e, \$\}) \cup \{z_f\}$, $X=T\cup\{\$\}$, $\Gamma=I'$, $z_0=(S', e)$, $F=\{z_f\}$, and define δ as follows:

(1) For all $c \in X$ let $((S, c), \#) \in \delta((S', e), c, \#)$.

(2) Let π: $Af \to aB\gamma$ be a production in P with $A \neq S'$.
 (a) If $a \neq e$ then $((B, c), \gamma) \in \delta((A, a), c, f)$ for all $c \in X$.
 (b) If $a = e$ then $((B, b), \gamma) \in \delta((A, b), e, f)$ for all $b \in {}^{(1)}\mathrm{First}_1(\pi)\$$.

(3) Let π: $Af \to a$ be a production in P.
 (a) If $a \neq e$ then $(z_f, e) \in \delta((A, a), \$, f)$.
 (b) If $a = e$ then $(z_f, e) \in \delta((A, \$), e, f)$.

First we make the following observations concerning K:

Claim 1. If $\delta((A, a), c, f) \neq \emptyset$ for $a, c \in X$ then there is a production π: $Af \to a\alpha$ and $\{a\} = \mathrm{First}_1(\pi) = {}^{(1)}\mathrm{First}_1(\pi)\$$.

Claim 2. If $\delta((A, a), e, f) \neq \emptyset$ for $a \in X$ then there is a production π: $Af \to \alpha$ with $\alpha = e$ or the first symbol of α is in N and furthermore $a \in {}^{(1)}\mathrm{First}_1(\pi)\$$.

Claim 1 and Claim 2 correspond to the subcases (a) and (b) respectively in the above definition of δ.

Claim 3. For all $z \in Z$, $c \in X \cup \{e\}$, and $f \in \Gamma$ we have $|\delta(z, c, f)| \leqq 1$.

If $z = (S', e)$ then $|\delta(z, c, f)| \leqq 1$ obviously holds. Now assume $z = (A, a)$ with $a \in X$ and $|\delta(z, c, f)| > 1$. Then there are productions π and π' with $\pi \neq \pi'$ and $a \in {}^{(1)}(\mathrm{First}_1(\pi)\$) \cap {}^{(1)}(\mathrm{First}_1(\pi')\$)$. This is a contradiction to the strong ILL(1) property of G.

Now consider $z = (A, a)$ with $a \in X$ and $f \in \Gamma$. If $\delta(z, e, f) \neq \emptyset$ and $\delta(z, c, f) \neq \emptyset$ for a $c \in X$ then Claim 1 and Claim 2 state the existence of two productions π and π' with $\pi \neq \pi'$ and furthermore $a \in {}^{(1)}(\mathrm{First}_1(\pi)\$) \cap {}^{(1)}(\mathrm{First}_1(\pi')\$)$. But this is a contradiction to the strong ILL(1) property of G.

If $z = (S', e)$ then $\delta(z, e, f) = \emptyset$ holds. Together with Claim 3 this shows that K is a deterministic pda.

To prove $L(K) = L(G')\$$ we need

Claim 4. If $S\# \overset{n}{\Rightarrow} wA\gamma\# \overset{*}{\Rightarrow} wv$, $w, v \in T^*$, $A \in N$, according to G' then $(z_0, wv\$, \#) \vdash^{n+1} ((A, c), v', \gamma\#)$ according to K, and $cv' = v\$$ with $c \neq e$.

The claim will be proved by induction on n.

If $n = 0$ then $w = \gamma = e$ and $A = s$. According to K we have

$$((S', e), v\$, \#) \vdash ((S, c), v', \#) \quad \text{with} \quad cv' = v\$ \quad \text{and } c \neq e.$$

Assume the claim holds for all $k \leqq n$.

Let $S\# \overset{n}{\Rightarrow} wA\gamma\# \overset{\pi}{\Rightarrow} waB\gamma'\# \overset{*}{\Rightarrow} wav$ be given where $a \in T \cup \{e\}$. From the induction hypothesis $(z_0, wav\$, \#) \vdash^{n+1} ((A, c'), v'', \gamma\#)$ with $c'v'' = av\$$ and $c' \neq e$ follows. If now $a \in T$ then $c' = a$ and $v'' = v\$$ holds. According to K the move $((A, a), v\$, \gamma\#) \vdash ((B, c), v', \gamma'\#)$ with $cv' = v\$$ and $c \neq e$ is possible. If $a = e$ then $c'v'' = v\$$ and $c' \in {}^{(1)}(\mathrm{First}_1(\pi)\$)$ holds. According to K the move $((A, c'), v'', \gamma\#) \vdash ((B, c'), v'', \gamma'\#)$ is possible. This completes the induction.

Now, by induction on n we will show

Claim 5. If $(z_0, wv\$, \#) \vdash^{n+1} ((A, c), v', \gamma\#)$ with $w, v \in T^*$, $cv' = v\$$ and $c \neq e$ holds according to K then $S\# \overset{n}{\Rightarrow} wA\gamma\#$ holds according to G.

If $n=0$ then we have $((S', e), wv\$, \#) \vdash ((S, c), v', \gamma\#)$ with $cv'=v\$$ according to K. This implies $w=\gamma=e$ and $S\#\overset{0}{\Rightarrow}wS\gamma\#$ holds according to G. Assume the claim holds for all $k \leqq n$ and let $(z_0, wv\$, \#) \overset{n+1}{\vdash} ((A', c'), v'', \gamma'\#) \vdash$ $\vdash ((A, c), v', \gamma\#)$ with $cv'=v\$$ and $c \neq e$ be given.

If $v''=v'$ then $c=c'$ and $c'v''=v\$$ holds. From the induction hypothesis $S\#\overset{n}{\Rightarrow}wA'\gamma'\#$ follows. Since $((A', c), v', \gamma'\#) \vdash ((A, c), v', \gamma\#)$ holds, there is a production π with $wA'\gamma'\#\overset{\pi}{\Rightarrow}wA\gamma\#$.

If $v'' \neq v'$ then $v''=cv'$ and $w=w'c'$. From the induction hypothesis $S\#\overset{n}{\Rightarrow}$ $\overset{n}{\Rightarrow}w'A'\gamma'\#$ follows. Since $((A', c'), v'', \gamma'\#) \vdash ((A, c), v', \gamma\#)$ holds there is a production π with $w'A'\gamma'\#\overset{\pi}{\Rightarrow}w'c'A\gamma\# = wA\gamma\#$. This completes the induction.

Now let $w \in L(K)$, i.e. $(z_0, w, \#) \overset{*}{\vdash} ((A, c), d, \gamma\#) \vdash (z_f, e, \gamma')$. Here we have either $d=\$$ or $d=e$ and $c=\$$. If $d=\$$ then $w=w'c\$$. Obviously, $w'c \in T^*$ holds. Therefore the derivation $S\#\overset{*}{\Rightarrow}w'A\gamma\#$ exists according to Claim 5. Since $((A, c), \$, \gamma\#) \vdash (z_f, e, \gamma')$ holds, there is a production π: $Af \to c$ with $w'A\gamma\#\overset{\pi}{\Rightarrow}$ $\overset{\pi}{\Rightarrow}w'c$. Therefore $w'c \in L(G)$ and $w=w'c\$ \in L(G')\$$. If $d=e$ and $c=\$$ then $w=w'\$$ with $w' \in T^*$, and the derivation $S\#\overset{*}{\Rightarrow}w'A\gamma\#$ exists according to Claim 5. Since $((A, \$), e, \gamma\#) \vdash (z_f, e, \gamma')$ holds there is a production $\pi: Af \to e$ with $w'A\gamma\#\overset{\pi}{\Rightarrow}w'$. Therefore $w' \in L(G')$ and $w=w'\$ \in L(G')\$$, and hence $L(K) \subseteqq L(G)\$$.

Conversely, let $w \in L(G')$, i.e. $S\#\overset{*}{\Rightarrow}w'A\gamma\# \Rightarrow w=w'c'$, $c' \in T \cup \{e\}$. Then $(z_0, w'c'\$, \#) \overset{*}{\vdash} ((A, c), d, \gamma\#)$ with $cd=c'\$$ and $c \neq e$ holds according to Claim 4. If $c'=e$ then $c=\$$ and $d=e$ hold. Hence the move $((A, \$), e, \gamma\#) \vdash (z_f, e, \gamma')$ exists. If $c' \in T$ then $c=c'$, $d=\$$, and the move $((A, c), \$, \gamma\#) \vdash (z_f, e, \gamma')$ exists. Therefore $w\$ \in L(K)$ and hence $L(G)\$ \subseteqq L(K)$.

Together with the inclusion $L(K) \subseteqq L(G)\$$ this shows $L(K)=L(G)\$$.

Since K is a deterministic pda, $L(G)\$$ is a deterministic context-free language which implies that $L(G')$ is a deterministic context-free language as well (see [3], Theorem 11.2.2).

Combining Corollary 5.1 and Theorem 5.3 we can state

Theorem 5.4. The class of deterministic context-free languages is exactly the class of right linear indexed strong ILL(1) languages.

The indexed grammar for the language $\{a^nb^nc^n | n \geqq 1\}$ given in [1] is a strong ILL(1) grammar. This proves

Theorem 5.5. The class of strong ILL(1) languages properly contains the class of the deterministic context-free languages.

6. A general transformation of indexed grammars

In Section 3 it was shown that the strong indexed LL(k) property is decidable. This will be used in Section 7 to prove the main result of this paper, namely the decidability of the (general) indexed LL(k) property.

To this end we first investigate in this section properties of two functions which are defined with respect to a given indexed grammar. Let $G=(N, T, I, P, S)$ be an indexed grammar and let λ: $I^* \to L$ and μ: $(NI^* \cup T)^* \to M$ be two functions in two finite nonempty sets L and M. λ and μ can be interpreted as assigning information

$\lambda(\gamma)$ and $\mu(\theta)$ from finite domains L and M to words $\gamma \in I^*$ and $\theta \in (NI^* \cup T)^*$, respectively. If λ and μ satisfy certain compatibility conditions C_λ and C_μ then it is possible to construct a grammar $G_{\lambda\mu}$ equivalent to G which simulates the derivations of G and attaches in left sentential forms $wA\gamma\theta$ of G the information $\lambda(\gamma)$ and $\mu(\theta)$ to the variable A. This means that the values of the functions λ and μ can be "computed" during leftmost derivations.

Let us now state the compatibility conditions C_λ and C_μ.

(C_λ) If $\lambda(\gamma_1) = \lambda(\gamma_2)$ then $\lambda(f\gamma_1) = \lambda(f\gamma_2)$ for all $\gamma_1, \gamma_2 \in I^*$, and $f \in I$.

(C_μ) If $\lambda(\gamma_1) = \lambda(\gamma_2)$ and $\mu(\theta_1) = \mu(\theta_2)$ then $\mu(\theta : \gamma_1\theta_1) = \mu(\theta : \gamma_2\theta_2)$ for all $\theta_1, \theta_2 \in (NI^* \cup T)^*$, $\theta \in NI^* \cup T$, and $\gamma_1, \gamma_2 \in I^*$.

Note: If C_λ is satisfied then it is easy to see that $\lambda(\gamma_1)=\lambda(\gamma_2)$ implies $\lambda(\alpha\gamma_1)=\lambda(\alpha\gamma_2)$ for all $\alpha \in I^*$. Furthermore it easy to prove that if C_μ holds then $\lambda(\gamma_1)=\lambda(\gamma_2)$ and $\mu(\theta_1)=\mu(\theta_2)$ imply $\mu(\theta:\gamma_1\theta_1)=\mu(\theta:\gamma_2\theta_2)$ for all $\theta \in (NI^* \cup T)^*$.

We will now give examples for functions λ and μ satisfying the conditions C_λ and C_μ.

Example 6.1. Let $G=(N, T, I, P, S)$ be an indexed grammar. Set $L=\mathscr{P}(N)$, the power set of N, and set $\lambda(\gamma)=\{A \mid A\gamma{}^*\Rightarrow e\}$. We will show that λ satisfies condition C_λ. For this purpose let $\gamma_1, \gamma_2 \in I^*$ with $\lambda(\gamma_1)=\lambda(\gamma_2)$, $f \in I$ and $A \in \lambda(f\gamma_1)$ be given. Since $A \in \lambda(f\gamma_1)$, there exists a derivation $Af\gamma_1{}^*\Rightarrow e$. If $Af{}^*\Rightarrow e$, then $A \in \lambda(f\gamma_2)$ obviously holds. Otherwise there exists a derivation $Af{}^*\Rightarrow B_1 \dots B_n$ with $B_i \in N$, $i \in [1: n]$ and $B_i\gamma_1{}^*\Rightarrow e$ for $i \in [1: n]$ holds. Therefore $B_i \in \lambda(\gamma_1)=\lambda(\gamma_2)$ and hence $B_i\gamma_2{}^*\Rightarrow e$ for $i \in [1: n]$ holds too. Consequently we have $Af\gamma_2{}^*\Rightarrow B_1\gamma_2 \dots B_n\gamma_2{}^*\Rightarrow e$ and $A \in \lambda(f\gamma_2)$ follows. This shows $\lambda(f\gamma_1) \subseteq \lambda(f\gamma_2)$. The converse inclusion follows by symmetry.

Example 6.2. Let $G_c=(N, T; P, S)$ be a context-free grammar which can be interpreted as an indexed grammar $G=(N, T, I, P, S)$ with $I=\emptyset$. Let L be a finite nonempty set and let λ: $I^* \to L$ be defined by $\lambda(e)=q$ for a $q \in L$. Obviously λ satisfies condition C_λ. For a $k \geqq 1$ set $M={}^{(k)}T^*$ and define $\mu(\theta)=\{u \mid \theta{}^*\Rightarrow v,\ v \in T^*,\ u={}^{(k)}v\}=\mathrm{First}_k(\theta)$ for all $\theta \in (N \cup T)^*$. We will show that μ satisfies condition C_μ. Let $\theta_1, \theta_2 \in (N \cup T)^*$ with $\mu(\theta_1)=\mu(\theta_2)$ be given. Let $\theta \in (N \cup T)^*$ and let $v \in \mu(\theta\theta_1)$. Hence there exists a derivation $\theta\theta_1{}^*\Rightarrow w$ with $v={}^{(k)}w$. This derivation can be written as $\theta\theta_1{}^*\Rightarrow u_1\theta_1{}^*\Rightarrow u_1u_2=w$. If $|u_1| \geqq k$ then $v={}^{(k)}u_1 \in \mu(\theta\theta_2)$ holds. Now assume $|u_1|<k$. We can state the existence of a derivation $\theta_2{}^*\Rightarrow \hat{u}_2$ with ${}^{(k)}\hat{u}_2={}^{(k)}u_2$ since $\mu(\theta_1)=\mu(\theta_2)$. Hence $\theta\theta_2{}^*\Rightarrow u_1\theta_2{}^*\Rightarrow u_1\hat{u}_2$ and ${}^{(k)}u_1\hat{u}_2=v \in \mu(\theta\theta_2)$ holds. Therefore we have $\mu(\theta\theta_1) \subseteq \mu(\theta\theta_2)$. The converse inclusion simply holds by symmetry.

This completes the examples and we return to the general discussion.

Let $\bar{L}=L \cup \{\bar{q}\}$ where $\bar{q}$ is a new element. With the aid of λ the function $\bar{\lambda}$: $I^* \times \bar{L} \to \bar{L}$ is defined for all $\gamma \in I^*$ and $q \in \bar{L}$ by

$$\begin{aligned}\bar{\lambda}(\gamma, q) &= \lambda(\gamma\gamma') \quad \text{if} \quad q = \lambda(\gamma') \quad \text{for a} \quad \gamma' \in I^* \\ &= \bar{q} \quad \text{otherwise.}\end{aligned}$$

If λ satisfies the condition C_λ then $\bar{\lambda}$ is well defined.

Now set $\bar{M}=M\cup\{\bar{m}\}$ where $\bar{m}$ is a new element and define the function $\bar{\mu}$: $(NI^*\cup T)^*\times\bar{M}\times\bar{L}\to\bar{M}$ for all $\theta\in(NI^*\cup T)^*$, $m\in\bar{M}$, and $q\in\bar{L}$ by

$$\begin{aligned}\bar{\mu}(\theta, m, q) &= \mu(\theta:\gamma\theta_1) \quad \text{if} \quad q=\lambda(\gamma) \quad \text{for a} \quad \gamma\in I^* \\ &\qquad\qquad\qquad\quad \text{and} \quad m=\mu(\theta_1) \quad \text{for a} \quad \theta_1\in(NI^*\cup T)^* \\ &= \bar{m} \quad \text{otherwise.}\end{aligned}$$

If μ satisfies the condition C_μ then $\bar{\mu}$ is well defined. Now we can state

Lemma 6.1.

(1) If λ is effectively computable and satisfies the condition C_λ then $\lambda(I^*)$ and $\bar{\lambda}$ are effectively computable, too.

(2) If λ and μ are effectively computable and satisfy the conditions C_λ and C_μ then $\mu((NI^*\cup T)^*)$ and $\bar{\mu}$ are effectively computable, too.

Proof. (1) We will first show that the value of λ for an index word with length greater than or equal to $|L|$ can be obtained by applying λ to a word of length less than $|L|$.

Let λ be effectively computable, i.e. there is an algorithm A_λ which determines for each given $\gamma\in I^*$ the value $\lambda(\gamma)$. With the aid of A_λ determine the set $\lambda({}^{(|L|-1)}I^*)$. (Recall that ${}^{(|L|-1)}I^*$ denotes the set of all words over I with length less than or equal to $|L|-1$.)

Let $\gamma=f_r\dots f_1\in I^*$ with $r\geqq|L|$ be given and let $q_k=\lambda(f_k\dots f_1)$ for $k\in[0:r]$. There exist $i,j\in[0:r]$ with $i<j$ and $q_i=q_j$ because $|L|=r$. Since λ satisfies the condition C_λ we have $\lambda(\gamma)=\lambda(f_r\dots f_{j+1}f_i\dots f_1)$. Hence $\lambda(I^*)=\lambda({}^{(|L|-1)}I^*)$ and this implies that $\lambda(I^*)$ is effectively computable.

To show that $\bar{\lambda}$ is effectively computable let $\gamma\in I^*$ and $q\in\bar{L}$ be given. If $q\notin\lambda(I^*)$ then $\bar{\lambda}(\gamma, q)=\bar{q}$ holds. Otherwise, if $q\in\lambda(I^*)$, determine a $\gamma'\in{}^{(|L|-1)}I^*$ with $\lambda(\gamma')=q$ and compute the value $\lambda(\gamma\gamma')$ with the aid of A_λ. This completes the proof of (1).

(2) We will first show that for each $\theta\in(NI^*\cup T)^*$ there exists a $\theta'\in(NI^*\cup T)^*$ with $\mu(\theta)=\mu(\theta')$, and with the length of the index words in θ' restricted by $|L|$. In the next step we show that it suffices to compute the values of μ for words over $(NI^*\cup T)^*$ with length less than $|M|$.

Let λ be effectively computable and let λ and μ satisfy the conditions C_λ and C_μ. First we will show by induction on the length that for each $\theta\in(NI^*\cup T)^*$ there exists a $\theta'\in\left(N({}^{(|L|-1)}I^*)\cup T\right)^*$ with $\mu(\theta)=\mu(\theta')$. In case $\theta=e$ the assertion is trivial. Let $\theta=\theta_1\theta_2$ with $\theta_1\in NI^*\cup T$ and $\theta_2\in(NI^*\cup T)^*$. The induction hypothesis guarantees the existence of a $\theta_2'\in\left(N({}^{(|L|-1)}I^*)\cup T\right)^*$ with $\mu(\theta_2)=\mu(\theta_2')$.

If $\theta_1=a\in T$ then $\mu(\theta)=\mu(a\theta_2)=\mu(a\theta_2')=\mu(\theta')$ with $\theta'\in\left(N({}^{(|L|-1)}I^*)\cup T\right)^*$ since μ satisfies condition C_μ.

Now let $\theta_1=A\gamma\in NI^*$. Part (1) of this lemma guarantees the existence of a $\gamma'\in{}^{(|L|-1)}I^*$ with $\lambda(\gamma)=\lambda(\gamma')$. Since μ satisfies the condition C_μ, we have $\mu(\theta)=$ $=\mu(A:\gamma\theta_2)=\mu(A:\gamma'\theta_2')=\mu(\theta')$ with $\theta'\in\left(N({}^{(|L|-1)}I^*)\cup T\right)^*$.

Now let μ be effectively computable, i.e. there is an algorithm A_μ which determines for each $\theta\in(NI^*\cup T)^*$ the value $\mu(\theta)$. With the aid of A_μ determine the set $\mu\left({}^{(|M|-1)}(N({}^{(|L|-1)}I^*)\cup T)^*\right)$. We will show that this set equals $\mu((NI^*\cup T)^*)$.

Let a $\theta\in(NI^*\cup T)^*$ be given. We have proved above the existence of a $\theta'\in(N({}^{(|L|-1)}I^*)\cup T)^*$ with $\mu(\theta)=\mu(\theta')$. Let $\theta'=\theta_r\ldots\theta_1$ with $\theta_k\in NI^*\cup T$ for $k\in[1: r]$ and $r\geqq|M|$. If we set $m_k=\mu(\theta_k\ldots\theta_1)$ for $k\in[0: r]$ then there exist $i,j\in[0: r]$ with $i<j$ and $m_i=m_j$. Since μ satisfies the condition C_μ, we have $\mu(\theta')=\mu(\theta_r\ldots\theta_{j+1}\theta_i\ldots\theta_1)$.

This implies $\mu((NI^*\cup T)^*)=\mu({}^{(|M|-1)}N({}^{(|L|-1)}I^*)\cup T)^*)$ and therefore $\mu((NI^*\cup T)^*)$ is effectively computable.

It remains to be shown that $\bar{\mu}$ is effectively computable. Let $\theta\in(NI^*\cup T)^*$, $m\in\overline{M}$, and $q\in\overline{L}$ be given. If $m\in\mu((NI^*\cup T)^*)$ and $q\in\lambda(I^*)$ then determine a θ_1 with $\mu(\theta_1)=m$ and a γ with $\lambda(\gamma)=q$ and compute the value $\mu(\theta:\gamma\theta_1)=\bar{\mu}(\theta, m, q)$ with the aid of A_μ. Otherwise, if $m=\overline{m}$ or $q=\bar{q}$, we have $\bar{\mu}(\theta, m, q)=\overline{m}$. This completes the proof of the lemma.

Starting with an indexed grammar $G=(N, T, I, P, S)$ and functions λ and μ satisfying C_λ and C_μ, we will construct an indexed grammar $G_{\lambda\mu}=(N', T, I', P', S')$. This grammar is structurally equivalent to G, i.e. generates the same set of terminal strings and the same set of derivation trees (ignoring the labels of the intermediate nodes).

Define $N'=N\times\overline{M}\times\overline{L}$, $I'=I\times\overline{L}$ and $S'=(S, m_0, q_0)$ with $m_0=\mu(e)$ and $q_0=\lambda(e)$.

For the definition of P' we need two functions φ and ψ. The function φ: $I^*\times\overline{L}\to(I\times\overline{L})^*=I'^*$ attaches a second component to each index f_i in an index word $f_1\ldots f_n$. For a given $q\in\overline{L}$ the second component of f_i will be value $\bar{\lambda}(f_{i+1}\ldots\ldots f_n, q)$, i.e. φ is defined by

$$\varphi(e, q)=e, \quad \text{and}$$

$$\varphi(f\gamma, q)=(f, \bar{\lambda}(\gamma, q))\varphi(\gamma, q) \quad \text{for all} \quad \gamma\in I^*, \quad f\in I, \quad \text{and} \quad q\in\overline{L}.$$

The function ψ: $(NI^*\cup T)^*\times\overline{M}\times\overline{L}\to((N\times\overline{M}\times\overline{L})(I\times\overline{L})^*\cup T)^*=(N'I'^*\cup T)^*$ attaches two components to the variables A_i in a word $A_1\gamma_1A_2\gamma_2\ldots A_n\gamma_n$ of $(NI^*\cup\cup T)^*$ with $A_i\in N\cup T, \gamma_i\in I^*$ for $i\in[1: n]$. For a given $m\in\overline{M}$ and $q\in\overline{L}$ the values of these components will be $\bar{\mu}(A_{i+1}\gamma_{i+1}\ldots A_n\gamma_n, m, q)$ and $\bar{\lambda}(\gamma_i, q)$ respectively. Furthermore the γ_j will be replaced by $\varphi(\gamma_j, q)$ for $j\in[1: n]$, i.e. ψ is defined by

$$\psi(e, m, q) \quad = e,$$

$$\psi(a\theta, m, q) \; = a\psi(\theta, m, q) \quad \text{and}$$

$$\psi(A\gamma\theta, m, q)=(A, \bar{\mu}(\theta, m, q), \bar{\lambda}(\gamma, q))\varphi(\gamma, q)\psi(\theta, m, q);$$

for all $A\in N$, $\gamma\in I^*$, $\theta\in(NI^*\cup T)^*$, $m\in\overline{M}$, $q\in\overline{L}$, and $a\in T$.

Now we are able to define the productions of $G_{\lambda\mu}$. Let π: $Af\to\beta$ be a production of P. Then for all $m\in\mu((NI^*\cup T)^*)$ and $q\in\lambda(I^*)$ the production

$$\pi^{m,q}\colon \psi(Af, m, q)\to\psi(\beta, m, q) \quad \text{is in} \quad P'.$$

Note that $\psi(Af, m, q)=(A, m, \bar{\lambda}(f, q))\varphi(f, q)\in N'\times(I'\cup\{e\})$, since $\varphi(f, q)=e$ if $f=e$, or $\varphi(f, q)=(f, q)$ if $f\neq e$.

$G_{\lambda\mu}$ is called the *$\lambda\mu$-grammar of G*. If the functions λ and μ are effectively computable, then the function ψ is also effectively computable and the grammar $G_{\lambda\mu}$ can be constructed effectively.

The homomorphism δ: $(N'\cup I'\cup T)^*\rightarrow(N\cup I\cup T)^*$ defined by $\delta(A, m, q)=A$, $\delta(f, q)=f$, and $\delta(a)=a$ for all $A\in N, m\in\overline{M}$, $q\in\overline{L}, f\in I, a\in T$, deletes the components attached to variables $A\in N$ and indices $f\in I$. Obviously, if $\theta_1' \overset{\pi^{m,q}}{\Rightarrow} \theta_2'$ according to $G_{\lambda\mu}$, where $\theta_1', \theta_2'\in(N'I'^*\cup T)^*$, then $\delta(\theta_1') \overset{\pi}{\Rightarrow} \delta(\theta_2')$ holds according to G.

Furthermore for all $\theta\in(NI^*\cup T)^*$, $m\in\overline{M}$, $q\in\overline{L}$, we have $\delta(\psi(\theta, m, q))=\theta$.

If $S'=(S, m_0, q_0) \overset{\pi_1^{m_1, q_1}}{\Rightarrow} \theta_1' \dots \overset{\pi_n^{m_n, q_n}}{\Rightarrow} \theta_n'$ is a leftmost derivation according to $G_{\lambda\mu}$, then

$$S \overset{\pi_1}{\Rightarrow} \delta(\theta_1') \Rightarrow \dots \overset{\pi_n}{\Rightarrow} \delta(\theta_n')$$

is called the *corresponding leftmost derivation according to G.*

In the following two lemmas we will make precise the structural equivalence of the grammars G and $G_{\lambda\mu}$. To prove these lemmas we need a number of facts concerning the functions $\bar{\lambda}$, $\bar{\mu}$, φ, and ψ.

Claim 1. For all $\theta, \theta_1, \theta_2\in(NI^*\cup T)^*$, and $\gamma, \gamma_1\in I^*$ we have

(i) $$\bar{\lambda}(e, \lambda(\gamma)) = \lambda(\gamma),$$

(ii) $$\bar{\lambda}(\gamma, q_0) = \lambda(\gamma) \quad \text{where} \quad q_0 = \lambda(e),$$

(iii) $$\bar{\lambda}(\gamma_1, \bar{\lambda}(\gamma, q)) = \bar{\lambda}(\gamma_1\gamma, q) \quad \text{for all} \quad q\in\overline{L},$$

(iv) $$\bar{\mu}(e, \mu(\theta), \lambda(\gamma)) = \mu(\theta),$$

(v) $$\bar{\mu}(\theta, m_0, q_0) = \mu(\theta) \quad \text{where} \quad m_0 = \mu(e) \quad \text{and} \quad q_0 = \lambda(e),$$

(vi) $$\bar{\mu}(\theta_1\theta, m, q) = \bar{\mu}(\theta_1, \bar{\mu}(\theta, m, q), q) \quad \text{for all} \quad m\in\overline{M} \quad \text{and} \quad q\in\overline{L},$$

(vii) $$\bar{\mu}(\theta{:}\gamma, m, q) = \bar{\mu}(\theta, m; \bar{\lambda}(\gamma, q)) \quad \text{for all} \quad m\in\overline{M} \quad \text{and} \quad q\in\overline{L}.$$

These identities are easily obtained from the definitions of the functions $\bar{\lambda}$ and $\bar{\mu}$.

The following three claims state properties of the functions φ and ψ. All claims are proved by induction on the length of words over I and $NI^*\cup T$ respectively.

Claim 2. For all $\gamma, \gamma_1\in I^*$ and $q\in\overline{L}$ we have

$$\varphi(\gamma\gamma_1, q) = \varphi(\gamma, \bar{\lambda}(\gamma_1, q))\varphi(\gamma_1, q).$$

Proof. The assertion holds for $\gamma=e$. If $\gamma=f\gamma'$ with $f\in I$ then

$$\begin{aligned}\varphi(f\gamma'\gamma_1, q) &= (f, \bar{\lambda}(\gamma'\gamma_1, q))\varphi(\gamma'\gamma_1, q)\\ &= (f, \bar{\lambda}(\gamma', \bar{\lambda}(\gamma_1, q)))\varphi(\gamma', \bar{\lambda}(\gamma_1, q))\varphi(\gamma_1, q)\end{aligned}$$

(see Claim 1 (iii) and induction hypothesis)

$$= \varphi(f\gamma', \bar{\lambda}(\gamma_1, q))\varphi(\gamma_1, q)$$

(see definition of φ).

Claim 3. For all $\theta, \theta_1\in(NI^*\cup T)^*$, $m\in\overline{M}$, $q\in\overline{L}$ we have

$$\psi(\theta_1\theta, m, q) = \psi(\theta_1, \bar{\mu}(\theta, m, q), q)\psi(\theta, m, q).$$

Proof. The assertion holds for $\theta_1 = e$. Assume that $\theta_1 = a\theta_1'$ with $a \in T$ and that the assertion holds for $\theta_1'\theta$. Then, by the definition of ψ we can conclude

$$\begin{aligned}\psi(a\theta_1'\theta, m, q) &= a\psi(\theta_1'\theta, m, q)\\ &= a\psi(\theta_1', \bar{\mu}(\theta, m, q), q)\psi(\theta, m, q)\\ &= \psi(a\theta_1', \bar{\mu}(\theta, m, q), q)\psi(\theta, m, q)\end{aligned}$$

Now assume $\theta_1 = A\gamma\theta_1'$ with $A\gamma \in NI^*$ and $\theta_1' \in (NI^* \cup T)^*$, and assume the assertion holds for $\theta_1'\theta$. Then, by the definition of ψ and Claim 1 (vi) we can conclude

$$\begin{aligned}\psi(A\gamma\theta_1'\theta, m, q) &= (A, \bar{\mu}(\theta_1'\theta, m, q), \bar{\lambda}(\gamma, q))\varphi(\gamma, q)\psi(\theta_1'\theta, m, q)\\ &= (A, \bar{\mu}(\theta_1'\theta, m, q), \bar{\lambda}(\gamma, q))\varphi(\gamma, q)\psi(\theta_1', \bar{\mu}(\theta, m, q), q)\psi(\theta, m, q)\\ &= (A, \bar{\mu}(\theta_1', \bar{\mu}(\theta, m, q), q), \bar{\lambda}(\gamma, q))\varphi(\gamma, q)\psi(\theta_1', \bar{\mu}(\theta, m, q), q)\psi(\theta, m, q)\\ &= \psi(A\gamma\theta_1', \bar{\mu}(\theta, m, q), q)\psi(\theta, m, q).\end{aligned}$$

Claim 4. For all $\theta \in (NI^* \cup T)^*$, $\gamma \in I^*$, $m \in \overline{M}$, and $q \in \overline{L}$ we have

$$\psi(\theta\colon \gamma, m, q) = \psi(\theta, m, \bar{\lambda}(\gamma, q))\colon \varphi(\gamma, q).$$

Proof. The assertion holds for $\theta = e$. Assume $\theta = a\theta_1$ with $a \in T$ and the assertion holds for θ_1. From the definition of ψ it follows:

$$\begin{aligned}\psi((a\theta_1)\colon\gamma, m, q) = \psi(a(\theta_1\colon \gamma), m, q) &= a\psi(\theta_1\colon \gamma, m, q)\\ &= a(\psi(\theta_1, m, \bar{\lambda}(\gamma, q))\colon \varphi(\gamma, q)) =\\ &= (a\psi(\theta_1, m, \bar{\lambda}(\gamma, q)))\colon \varphi(\gamma, q) =\\ &= \psi(a\theta_1, m, \bar{\lambda}(\gamma, q))\colon \varphi(\gamma, q)\end{aligned}$$

Now assume $\theta = A\gamma_1\theta_1$ with $A\gamma_1 \in NI^*$, $\theta_1 \in (NI^* \cup T)^*$ and assume that the assertion holds for θ_1. Then, by the definition of ψ, Claim 1 (iii), Claim 1 (vii), and Claim 2 we have

$$\begin{aligned}&\psi((A\gamma_1\theta_1)\colon\gamma, m, q) = \psi(A\gamma_1\gamma(\theta_1\colon \gamma), m, q)\\ &= (A, \bar{\mu}(\theta_1\colon \gamma, m, q), \bar{\lambda}(\gamma_1\gamma, q))\varphi(\gamma_1\gamma, q)\psi(\theta_1\colon \gamma, m, q)\\ &= (A, \bar{\mu}(\theta_1, m, \bar{\lambda}(\gamma, q)), \bar{\lambda}(\gamma_1\gamma, q))\varphi(\gamma_1, \bar{\lambda}(\gamma, q))\varphi(\gamma, q)\,\psi(\theta_1, m, \bar{\lambda}(\gamma, q))\colon \varphi(\gamma, q)\\ &= ((A, \bar{\mu}(\theta_1, m, \bar{\lambda}(\gamma, q)), \bar{\lambda}(\gamma_1, \bar{\lambda}(\gamma, q)))\varphi(\gamma_1, \bar{\lambda}(\gamma, q))\psi(\theta_1, m, \bar{\lambda}(\gamma, q)))\colon \varphi(\gamma, q)\\ &= \psi(A\gamma_1\theta_1, m, \bar{\lambda}(\gamma, q))\colon \varphi(\gamma, q).\end{aligned}$$

For the remainder of this section we will use the following general assumptions:

(*) Let $G = (N, T, I, P, S)$ be an indexed grammar and let $\lambda\colon I^* \to L$ and $\mu\colon (NI^* \cup T)^* \to M$ be two functions in two finite sets L and M satisfying the conditions C_λ and C_μ. Let $\bar{L} = L \cup \{\bar{q}\}$ and $\bar{M} = M \cup \{\bar{m}\}$ where $\bar{q}$ and $\bar{m}$ are new elements and let $G_{\lambda\mu} = (N', T, I', P', S')$ be the $\lambda\mu$-grammar of G.

The next lemma establishes a correspondence between a leftmost derivation step in G and an analogous leftmost derivation step in $G_{\lambda\mu}$.

Lemma 6.2. Under the assumptions (*) for all $w \in T^*$, $\gamma \in I^*$, $\theta \in (NI^* \cup T)^*$ the following holds:

If π: $Af \to \beta \in P$ and $wAf\gamma\theta \overset{\pi}{\Rightarrow} w\beta:\gamma\theta$ according to G, then for all $m_1 \in \mu((NI^* \cup T)^*)$ and $q_1 \in \lambda(I^*)$ we have $\psi(wAf\gamma\theta, m_1, q_1) \overset{\pi^{m,q}}{\Rightarrow} \psi(w\beta:\gamma\theta, m_1, q_1)$, where $m = \bar{\mu}(\theta, m_1, q_1)$ and $q = \bar{\lambda}(\gamma, q_1)$.

Proof. Since $m_1 \in \mu((NI^* \cup T)^*)$ and $q_1 \in \lambda(I^*)$, the same holds for m and q, hence $\pi^{m,q} \in P'$. Consider $\pi^{m,q}$: $(A, m, \bar{\lambda}(f, q))\varphi(f, q) \to \psi(\beta, m, q) \in P'$. By the definitions of φ and ψ and Claim 1 (iii) we have

$$\psi(wAf\gamma\theta, m_1, q_1) = w(A, m, \bar{\lambda}(f\gamma, q_1))\varphi(f\gamma, q_1)\psi(\theta, m_1, q_1) =$$
$$= w(A, m, \bar{\lambda}(f, q))\varphi(f, q)\varphi(\gamma, q_1)\psi(\theta, m_1, q_1)$$

This shows that $\pi^{m,q}$ is applicable to $\psi(wAf\gamma\theta, m_1, q_1)$, i.e. the following leftmost derivation step is possible:

$$w(A, m, \bar{\lambda}(f, q))\varphi(f, q)\varphi(\gamma, q_1)\psi(\theta, m_1, q_1)$$
$$\overset{\pi^{m,q}}{\Rightarrow} w\psi(\beta, m, q)\colon \varphi(\gamma, q_1)\psi(\theta, m_1, q_1).$$

Using the Claims 3 and 4 we have

$$w\psi(\beta, m, q):\varphi(\gamma, q_1)\psi(\theta, m_1, q_1) = w\psi(\beta:\gamma, m, q_1)\psi(\theta, m_1, q_1)$$
$$= \psi(w\theta_1\theta, m_1, q_1).$$

This completes the proof.

The following lemma is in some sense the converse of Lemma 6.2 and describes the simulation of a leftmost derivation step according to $G_{\lambda\mu}$ in the grammar G.

Lemma 6.3. Under the asumptions (*) the following holds for all $w \in T^*$, $\gamma \in I^*$, and $\theta \in (NI^* \cup T)^*$: If $\pi^{m,q}$: $\psi(Af, m, q) \to \psi(\beta, m, q) \in P'$ and $\psi(wAf\gamma\theta, m_1, q_1) \overset{\pi^{m,q}}{\Rightarrow} \bar{\theta}$, where $m_1 \in \mu((NI^* \cup T)^*)$ and $q_1 \in \lambda(I^*)$, then $m = \bar{\mu}(\theta, m_1, q_1)$, $q = \bar{\lambda}(\gamma, q_1)$, and $\bar{\theta} = \psi(w\beta:\gamma\theta, m_1, q_1)$.

Proof. Since $\psi(wAf\gamma\theta, m_1, q_1) =$

$$w(A, \bar{\mu}(\theta, m_1, q_1), \bar{\lambda}(f\gamma, q_1))\varphi(f\gamma, q_1)\psi(\theta, m_1, q_1) =$$
$$w(A, \bar{\mu}(\theta, m_1, q_1), \bar{\lambda}(f, \bar{\lambda}(\gamma, q_1)))\varphi(f, \bar{\lambda}(\gamma, q_1))\varphi(\gamma, q_1)\psi(\theta, m_1, q_1)$$

(see Claim 1 (iii)) we have $m = \bar{\mu}(\theta, m_1, q_1)$ and $q = \bar{\lambda}(\gamma, q_1)$.

Furthermore, with the aid of Claims 3 and 4 we have:

$$\bar{\theta} = w\psi(\beta, m, q)\colon \varphi(\gamma, q_1)\psi(\theta, m_1, q_1) = w\psi(\beta:\gamma, m, q_1)\psi(\theta, m_1, q_1)$$
$$= \psi(w\theta_1\theta, m_1, q_1)$$

This completes the proof.

The repeated application of Lemma 6.2 yields

Corollary 6.1. Under the assumptions (*) we have: If $S \overset{*}{\Rightarrow} wA\gamma\theta$ acccording to G with $w \in T^*$, $A \in N$, $\gamma \in I^*$, and $\theta \in (NI^* \cup T)^*$ then $S' = (S, m_0, q_0) \overset{*}{\Rightarrow} \psi(wA\gamma\theta, m_0, q_0)$ according to $G_{\lambda\mu}$.

Remark. Since

$$\psi(wA\gamma\theta, m_0, q_0) = w(A, \bar{\mu}(\theta, m_0, q_0), \bar{\lambda}(\gamma, q_0))\varphi(\gamma, q_0)\psi(\theta, m_0, q_0) =$$
$$= w(A, \mu(\theta), \lambda(\gamma))\varphi(\gamma, q_0)\psi(\theta, m_0, q_0)$$

holds, we conclude: $G_{\lambda\mu}$ simulates the.leftmost derivations of left sentential forms $wA\gamma\theta$ of G and attaches the information $\lambda(\gamma)$ and $\mu(\theta)$ to the variable A.

Repeated application of Lemma 6.3 yields

Corollary 6.2. Let $S'=(S, m_0, q_0)\overset{*}{\Rightarrow}\theta'$ be a leftmost derivation according to $G_{\lambda\mu}$ and let $S\overset{*}{\Rightarrow}\theta$ be the corresponding leftmost derivation according to G. Then $\theta'=\psi(\theta, m_0, q_0)$ holds.

Furthermore, with Lemma 6.2 we obtain

Corollary 6.3. Under the assumptions (*) we have: If $\theta\in(NI^*\cup T)^*$, $m\in\mu((NI^*\cup \cup T)^*)$, $q\in\lambda(I^*)$, and $v\in T^*$ then $\psi(\theta, m, q)\overset{*}{\Rightarrow}v$ according to $G_{\lambda\mu}$ iff $\theta\overset{*}{\Rightarrow}v$ according to G. In particular $L(G)=L(G_{\lambda\mu})$ holds.

Remark. The underlying principle of this construction is applied for example in [1, 9]. In [1] the function λ of Example 6.1 is used for constructing a normal form of an indexed grammar. In [9] the function μ of Example 6.2 is used in investigations of context-free LL(k) grammars.

A construction similar to the construction of $G_{\lambda\mu}$ can be applied to indexed pushdown automata and pushdown automata. The principles of this construction are used in [7, 8] in the indexed case and, for example, to prove closure properties of deterministic languages in the context-free case (cf. [3], Section 11.2).

7. Decidability of indexed LL(k)

In this section we will prove our main theorem concerning the connection between ILL(k) and strong ILL(k) grammars. For this purpose we will introduce two special functions λ and μ in the following manner.

Let $G=(N, T, I, P, S)$ be an indexed grammar with $P=\{\pi_1, \pi_2, \ldots, \pi_p\}$. Set $L=(\mathscr{P}({}^{(k)}T^*))^p$, the set of all p-vectors, whose components are subsets of ${}^{(k)}T^*$, and $M=\mathscr{P}({}^{(k)}T^*)$ with $k\geqq 1$.

Define $\lambda\colon I^*\to L$ by $\lambda(\gamma)=(q_1, \ldots, q_p)$ where $q_i=\mathrm{First}_k(\pi_i, \gamma)$ holds for $i\in[1\colon p]$. Furthermore define $\mu\colon (NI^*\cup T)^*\to M$ by $\mu(\theta)=\mathrm{First}_k(\theta)$. From Corollary 3.1 we know that λ and μ are effectively computable.

First we want to show that λ satisfies condition C_λ. For this purpose let $\gamma_1, \gamma_2\in I^*$ with $\lambda(\gamma_1)=\lambda(\gamma_2)$ be given, i.e. $\mathrm{First}_k(\pi_i, \gamma_1)=\mathrm{First}_k(\pi_i, \gamma_2)$ holds for all $i\in[1\colon p]$. Under this assumption we will show by induction on n that for each $\gamma\in I^*$ and each production π_j the following holds:

If

$$A\gamma\gamma_1 \overset{\pi_j}{\Rightarrow} \theta_1 \overset{n}{\Rightarrow} u \quad \text{with} \quad u\in T^* \quad \text{then}$$

$$A\gamma\gamma_2 \overset{\pi_j}{\Rightarrow} \theta_2 \overset{*}{\Rightarrow} v \quad \text{with} \quad v\in T^* \quad \text{and} \quad {}^{(k)}u = {}^{(k)}v.$$

(Recall, $\overset{n}{\Rightarrow}$ denotes a leftmost derivation in n steps.)

Obviously this assertion holds for $n=0$. Now let $A\gamma\gamma_1 \overset{\pi_j}{\Rightarrow} \theta_1 \overset{n+1}{\Rightarrow} u$ with $u \in T^*$ be given.

If $\gamma=e$, the assertion follows from $\lambda(\gamma_1)=\lambda(\gamma_2)$.

If $\gamma \neq e$, no index of γ_1 can be consumed in the first step of the derivation. Therefore we have

$$\theta_1 = B_1\beta_1 \ldots B_r\beta_r \quad \text{with} \quad B_i \in N \cup T, r > 0, \quad \text{and}$$

$$\beta_i = \gamma_i'\gamma_1 \quad \text{with} \quad \gamma_i' \in I^* \quad \text{if} \quad B_i \in N$$

$$= e \quad \text{otherwise.}$$

In addition $B_i\beta_i \overset{n_i}{\Rightarrow} u_i$ with $n_i \leqq n+1$ holds for $i \in [1: r]$ where $u=u_1 \ldots u_r$. Since $\gamma \neq e$ it is possible to apply π_j to $A\gamma\gamma_2$ which yields

$$A\gamma\gamma_2 \overset{\pi_j}{\Rightarrow} B_1\beta_1' \ldots B_r\beta_r' = \theta_2 \quad \text{with}$$

$$\beta_i' = \gamma_i'\gamma_2 \quad \text{with} \quad \gamma_i' \in I^* \quad \text{if} \quad B_i \in N$$

$$= e \text{ otherwise.}$$

The induction hypothesis quarantees the existence of the derivations

$$B_i\beta_i' \overset{*}{\Rightarrow} v_i \quad \text{with} \quad v_i \in T^* \quad \text{and} \quad {}^{(k)}v_i = {}^{(k)}u_i \quad \text{for} \quad i \in [1: r].$$

Consequently we have

$$A\gamma\gamma_2 \overset{\pi_j}{\Rightarrow} \theta_2 \overset{*}{\Rightarrow} v_1 \ldots v_r = v \quad \text{with} \quad {}^{(k)}v = {}^{(k)}u.$$

This completes the induction, and hence

$$\mathrm{First}_k(\pi_j, \gamma\gamma_1) \subseteqq \mathrm{First}_k(\pi_j, \gamma\gamma_2).$$

The converse inclusion holds by symmetry. The special case $\gamma \in I$ shows that λ satisfies condition C_λ.

We now prove that μ satisfies condition C_μ.

Let $\gamma_1, \gamma_2 \in I^*$ with $\lambda(\gamma_1)=\lambda(\gamma_2)$ and $\theta_1, \theta_2 \in (NI^* \cup T)^*$ with $\mu(\theta_1)=\mu(\theta_2)$ be given, i.e. for all productions π_i we have $\mathrm{First}_k(\pi_i, \gamma_1)=\mathrm{First}_k(\pi_i, \gamma_2)$, and furthermore $\mathrm{First}_k(\theta_1)=\mathrm{First}_k(\theta_2)$ holds.

Now for each $\theta \in NI^* \cup T$ the equality $\mathrm{First}_k(\theta{:}\gamma_1\theta_1)=\mathrm{First}_k(\theta{:}\gamma_2\theta_2)$ has to be shown.

If $\theta \in T$ this assertion holds obviously.

If $\theta=A\gamma \in NI^*$ then observe that $\mathrm{First}_k(A\gamma\gamma_1)=\bigcup\limits_{i \in J} \mathrm{First}_k(\pi_i, \gamma\gamma_1)$ holds where J is the set of all numbers of productions with lefthand side Af, $f \in I \cup \{e\}$. Since $\lambda(\gamma_1)=\lambda(\gamma_2)$ and λ satisfies condition C_λ, we have $\lambda(\gamma\gamma_1)=\lambda(\gamma\gamma_2)$. Hence we have

$$\mathrm{First}_k(A\gamma\gamma_1) = \bigcup_{i \in J} \mathrm{First}_k(\pi_i, \gamma\gamma_2) = \mathrm{First}_k(A\gamma\gamma_2).$$

This equality enables us to conclude

$$\begin{aligned}\mathrm{First}_k(A\gamma\gamma_1\theta_1) &= {}^{(k)}(\mathrm{First}_k(A\gamma\gamma_1)\,\mathrm{First}_k(\theta_1))\\ &= {}^{(k)}(\mathrm{First}_k(A\gamma\gamma_2)\,\mathrm{First}_k(\theta_2))\\ &= \mathrm{First}_k(A\gamma\gamma_2\theta_2).\end{aligned}$$

Consequently μ satisfies the condition C_μ.

According to Lemma 6.1 we observe that $\lambda(I^*)$, $\bar{\lambda}$, $\mu((NI^*\cup T)^*)$, and $\bar{\mu}$ are effectively computable.

The $\lambda\mu$-grammar of G defined by these special functions λ and μ will be called the S_k*-grammar* of G and denoted by $S_k(G)$.

Now we can state our main theorem.

Theorem 7.1. Let $G=(N, T, I, P, S)$ be an indexed grammar and let $k\geqq 1$. $S_k(G)$, the S_k-grammar of G, is a strong ILL(k) grammar iff G is an ILL(k) grammar.

Proof. (a) Let G be an ILL(k) grammar. If $S_k(G)$ does not satisfy the strong ILL(k) condition, then the following two cases are possible.

(1) There are two productions $\pi_i^{m,q}$: $\psi(Af, m, q)\to\psi(\beta_1, m, q)$ and $\pi_j^{m,q}$: $\psi(Af, m, q)\to\psi(\beta_2, m, q)$ in P' with the same lefthand side, where $i\neq j$, $f\in I\cup\{e\}$, and $\mathrm{First}_k(\pi_i^{m,q})\cap\mathrm{First}_k(\pi_j^{m,q})\neq\emptyset$, i.e. there are two leftmost derivations

$$S' \overset{*}{\Rightarrow} w\psi(Af, m, q)\gamma'\theta' \overset{\pi_i^{m,q}}{\Rightarrow} w\theta_1'\theta' \overset{*}{\Rightarrow} wu\theta' \overset{*}{\Rightarrow} wuv \tag{7.1}$$

and

$$S' \overset{*}{\Rightarrow} \bar{w}\psi(Af, m, q)\bar{\gamma}'\bar{\theta}' \overset{\pi_j^{m,q}}{\Rightarrow} \bar{w}\bar{\theta}_1'\bar{\theta}' \overset{*}{\Rightarrow} \bar{w}\bar{u}\bar{\theta}' \Rightarrow \bar{w}\bar{u}\bar{v} \tag{7.2}$$

according to $S_k(G)$ with $w, \bar{w}, u, \bar{u}, v, \bar{v}\in T^*$, $\gamma', \bar{\gamma}'\in I'^*$, and

$$\theta', \theta_1', \bar{\theta}', \bar{\theta}_1'\in(N'I'^*\cup T)^*, \quad \text{where}$$

$$\theta_1' \overset{*}{\Rightarrow} u, \quad \theta' \overset{*}{\Rightarrow} v,$$

$$\bar{\theta}_1' \overset{*}{\Rightarrow} \bar{u}, \quad \bar{\theta}' \overset{*}{\Rightarrow} \bar{v},$$

and ${}^{(k)}uv={}^{(k)}\bar{u}\bar{v}$ holds.

Consider the derivation (7.1). We have the corresponding leftmost derivation

$$S \overset{*}{\Rightarrow} wAf\gamma\theta \overset{\pi_i}{\Rightarrow} w\theta_1\theta \overset{*}{\Rightarrow} wu\theta \overset{*}{\Rightarrow} wuv,$$

where $\gamma=\delta(\gamma')$, $\theta_1=\delta(\theta_1')$, and $\theta=\delta(\theta')$.

Then from Corollary 6.2 we conclude $w\psi(Af, m, q)\gamma'\theta'=\psi(wAf\gamma\theta, m_0, q_0)$. Since $\psi(wAf\gamma\theta, m_0, q_0)\overset{\pi_i^{m,q}}{\Rightarrow}w\theta_1'\theta'$ we have from Lemma 6.3 $m=\bar{\mu}(\theta, m_0, q_0)= =\mu(\theta)$, and $q=\bar{\lambda}(\gamma, q_0)=\lambda(\gamma)$.

Analogously, considering the derivation (7.2), we have a corresponding leftmost derivation

$$S \overset{*}{\Rightarrow} \bar{w}Af\bar{\gamma}\bar{\theta} \overset{\pi_j}{\Rightarrow} \bar{w}\bar{\theta}_1\bar{\theta} \overset{*}{\Rightarrow} \bar{w}\bar{u}\bar{\theta} \overset{*}{\Rightarrow} \bar{w}\bar{u}\bar{v},$$

where $\bar{\gamma}=\delta(\bar{\gamma}')$, $\bar{\theta}_1=\delta(\bar{\theta}_1')$, and $\bar{\theta}=\delta(\bar{\theta}')$. Furthermore we have $m=\mu(\bar{\theta})$ and $q=\lambda(\bar{\gamma})$.

Since $\lambda(\gamma)=\lambda(\bar{\gamma})$ we have in particular $\mathrm{First}_k(\pi_j, f\gamma)=\mathrm{First}_k(\pi_j, f\bar{\gamma})$ and since $\mu(\theta)=\mu(\bar{\theta})$ we have $\mathrm{First}_k(\theta)=\mathrm{First}_k(\bar{\theta})$.

If $|\bar{u}| \geqq k$ then ${}^{(k)}\bar{u} \in \mathrm{First}_k(\pi_j, f\bar{\gamma})$ and therefore ${}^{(k)}\bar{u} \in \mathrm{First}_k(\pi_j, f\gamma)$, i.e. $Af\gamma \,{}^{\pi_j}\!\!\Rightarrow \beta_2 : \gamma \,{}^*\!\!\Rightarrow \hat{u}$ according to G with ${}^{(k)}\hat{u} = {}^{(k)}\bar{u}$.

Then the following derivation according to G is possible:

$$S \,{}^*\!\!\Rightarrow wAf\gamma\theta \,{}^{\pi_j}\!\!\Rightarrow w\beta_2 : \gamma\theta \,{}^*\!\!\Rightarrow w\hat{u}v$$

with ${}^{(k)}\hat{u}v = {}^{(k)}\bar{u} = {}^{(k)}uv$. But this is a contradiction to the ILL(k) property of G.

If $|\bar{u}| < k$, then $\bar{u} \in \mathrm{First}_k(\pi_j, f\bar{\gamma})$ and therefore $\bar{u} \in \mathrm{First}_k(\pi_j, f\gamma)$, i.e. $Af\gamma \,{}^{\pi_j}\!\!\Rightarrow$ ${}^{\pi_j}\!\!\Rightarrow \beta_2 : \gamma \,{}^*\!\!\Rightarrow \bar{u}$ according to G. Since $\mathrm{First}_k(\theta) = \mathrm{First}_k(\bar{\theta})$ there exists $\theta \,{}^*\!\!\Rightarrow \hat{v}$ according to G with ${}^{(k)}\hat{v} = {}^{(k)}\bar{v}$. Then the following derivation according to G is possible:

$$S \,{}^*\!\!\Rightarrow wAf\gamma\theta \,{}^{\pi_j}\!\!\Rightarrow w\beta_2 : \gamma\theta \,{}^*\!\!\Rightarrow w\bar{u}\theta \,{}^*\!\!\Rightarrow w\bar{u}\hat{v}$$

with ${}^{(k)}\bar{u}\hat{v} = {}^{(k)}\bar{u}\bar{v} = {}^{(k)}uv$. This is a contradiction to the ILL(k) property of G.

(2) There are productions $\pi_i^{m,q} : \psi(Af, m, q) \to \psi(\beta_1, m, q)$, $f \in I$, and $\pi_j^{m,q'}$: $\psi(A, m, q') \to \psi(\beta_2, m, q')$ in P' with $q' = \bar{\lambda}(f, q)$ and

$$S' \,{}^*\!\!\Rightarrow w\psi(Af; m, q)\gamma'\theta' \,{}^{\pi_i^{m,q}}\!\!\Rightarrow w\theta_1'\theta' \,{}^*\!\!\Rightarrow wu\theta' \,{}^*\!\!\Rightarrow wuv \tag{7.3}$$

and

$$S' \,{}^*\!\!\Rightarrow \bar{w}\psi(A, m, q')\bar{\gamma}'\bar{\theta}' \,{}^{\pi_j^{m,q'}}\!\!\Rightarrow \bar{w}\bar{\theta}_1'\bar{\theta}' \,{}^*\!\!\Rightarrow \bar{w}\bar{u}\bar{\theta}' \,{}^*\!\!\Rightarrow \bar{w}\bar{u}\bar{v} \tag{7.4}$$

according to $S_k(G)$ with $w, \bar{w}, u, \bar{u}, v, \bar{v} \in T^*, \gamma', \bar{\gamma}' \in I'^*$,

$$\theta_1', \theta', \bar{\theta}_1', \bar{\theta}' \in (N'I'^* \cup T)^* \quad \text{where}$$

$$\theta_1' \,{}^*\!\!\Rightarrow u, \quad \theta' \,{}^*\!\!\Rightarrow v,$$

$$\bar{\theta}_1' \,{}^*\!\!\Rightarrow \bar{u}, \quad \bar{\theta}' \,{}^*\!\!\Rightarrow \bar{v}$$

and ${}^{(k)}uv = {}^{(k)}\bar{u}\bar{v}$ holds.

Consider the derivation (7.3). We have the corresponding leftmost derivation

$$S \,{}^*\!\!\Rightarrow wAf\gamma\theta \,{}^{\pi_i}\!\!\Rightarrow w\theta_1\theta \,{}^*\!\!\Rightarrow wu\theta \,{}^*\!\!\Rightarrow wuv,$$

where $\gamma = \delta(\gamma')$, $\theta_1 = \delta(\theta_1')$, and $\theta = \delta(\theta')$. As above we conclude $m = \mu(\theta)$ and $q = \lambda(\gamma)$.

Considering the derivation (7.4), we have the corresponding leftmost derivation

$$S \,{}^*\!\!\Rightarrow \bar{w}A\bar{\gamma}\bar{\theta} \,{}^{\pi_j}\!\!\Rightarrow \bar{w}\bar{\theta}_1\bar{\theta} \,{}^*\!\!\Rightarrow \bar{w}\bar{u}\bar{\theta}' \,{}^*\!\!\Rightarrow \bar{w}\bar{u}\bar{v},$$

where $\bar{\gamma} = \delta(\bar{\gamma}')$, $\bar{\theta}_1 = \delta(\bar{\theta}_1')$, and $\bar{\theta} = \delta(\bar{\theta}')$. Furthermore $m = \mu(\bar{\theta})$ and $q' = \lambda(\bar{\gamma})$.

Since $\lambda(f\gamma) = \bar{\lambda}(f, q) = q' = \lambda(\bar{\gamma})$ we have in particular $\mathrm{First}_k(\pi_j, \bar{\gamma}) = \mathrm{First}_k(\pi_j, f\gamma)$ and since $\mu(\theta) = \mu(\bar{\theta})$ we have $\mathrm{First}_k(\bar{\theta}) = \mathrm{First}_k(\theta)$.

If $|\bar{u}| \geqq k$ then ${}^{(k)}\bar{u} \in \mathrm{First}_k(\pi_j, \bar{\gamma})$ and therefore ${}^{(k)}u \in \mathrm{First}_k(\pi_j, f\gamma)$, i.e. $Af\gamma \,{}^{\pi_j}\!\!\Rightarrow$ ${}^{\pi_j}\!\!\Rightarrow \beta_2 : f\gamma \,{}^*\!\!\Rightarrow \hat{u}$ with ${}^{(k)}\hat{u} = {}^{(k)}\bar{u}$.

Hence the derivation

$$S \,{}^*\!\!\Rightarrow wAf\gamma\theta \,{}^{\pi_j}\!\!\Rightarrow w\beta_2 : f\gamma\theta \,{}^*\!\!\Rightarrow w\hat{u}\theta \,{}^*\!\!\Rightarrow w\hat{u}v$$

according to G is possible with ${}^{(k)}\hat{u}v = {}^{(k)}\hat{u} = {}^{(k)}\bar{u} = {}^{(k)}uv$. This is a contradiction to the ILL(k) property of G.

If $|\bar{u}|<k$ then $\bar{u}\in\text{First}_k(\pi_j, \bar{\gamma})$ and therefore $\bar{u}\in\text{First}_k(\pi_j, f\gamma)$, i.e. $Af\gamma \xRightarrow{\pi_j}$ $\xRightarrow{\pi_j} \beta_2{:}\, f\gamma \xRightarrow{*} \bar{u}$ holds according to G. Since $\text{First}_k(\theta)=\text{First}_k(\bar{\theta})$ we have $\theta \xRightarrow{*} \hat{v}$ with ${}^{(k)}\hat{v}={}^{(k)}v$. Hence

$$S \xRightarrow{*} wAf\gamma\theta \xRightarrow{\pi_j} w\beta_2{:}f\gamma\theta \xRightarrow{*} w\bar{u}\theta \xRightarrow{*} w\bar{u}\hat{v}$$

is possible according to G with ${}^{(k)}\bar{u}\hat{v}={}^{(k)}\bar{u}\bar{v}={}^{(k)}uv$ which contradicts the ILL(k) poperty of G.

(b) Let $S_k(G)$ be a strong ILL(k) grammar. Assume that G is not an ILL(k) grammar, i.e. there are two leftmost derivations

$$S \xRightarrow{*} wA\gamma\theta \xRightarrow{\pi_i} w\theta_1 \xRightarrow{*} wx \quad \text{and}$$

$$S \xRightarrow{*} wA\gamma\theta \xRightarrow{\pi_j} w\theta_2 \xRightarrow{*} wy$$

with $A\in N$, $\gamma\in I^*$, $\theta_1, \theta_2\in(NI^*\cup T)^*$, $w, x, y\in T^*$, ${}^{(k)}x={}^{(k)}y$ and $i\neq j$.

Let us consider the case that the lefthand side of the two productions are different, i.e. π_i: $A\to\beta_1$ and π_j: $Af\to\beta_2$ with $f\in I$. Hence $\gamma=f\gamma'$ holds. Set $m=$ $=\mu(\theta)=\bar{\mu}(\theta, m_0, q_0), q'=\lambda(\gamma')=\bar{\lambda}(\gamma', q_0)$ and $q=\lambda(\gamma)=\bar{\lambda}(\gamma, q_0)=\bar{\lambda}(f, q')$.

In P' there are the productions

$$\pi_i^{m,q}\colon\ \psi(A, m, q) \to \psi(\beta_1, m, q)$$

and

$$\pi_j^{m,q'}\colon\ \psi(Af, m, q') \to \psi(\beta_2, m, q').$$

With Corollary 6.1, Lemma 6.2 and Corollary 6.3 the existence of the leftmost derivations

$$S' \xRightarrow{*} \psi(wA\gamma\theta, m_0, q_0) \xRightarrow{\pi_i^{m,q}} \psi(w\theta_1, m_0, q_0) \xRightarrow{*} wx \quad \text{and}$$

$$S' \xRightarrow{*} \psi(wA\gamma\theta, m_0, q_0) \xRightarrow{\pi_j^{m,q'}} \psi(w\theta_2, m_0, q_0) \xRightarrow{*} wy$$

according to $S_k(G)$ follows.

Since ${}^{(k)}x={}^{(k)}y$ we have $\text{First}_k(\pi_i^{m,q})\cap\text{First}_k(\pi_j^{m,q'})\neq\emptyset$ which is a contradiction to the strong ILL(k) property of $S_k(G)$.

In a similar manner the case that π_i and π_j posess the same lefthand side yield a contradiction.

This completes the proof of the theorem.

Now we can easily derive the following decidability result.

Theorem 7.2. Given an indexed grammar G and an integer $k\geqq 1$. It is then decidable whether G is an ILL(k) grammar.

Proof. S_k-grammar of G is effectively constructable since the functions λ and μ defining this grammar are effectively computable. Furthermore it is decidable whether $S_k(G)$ is a strong ILL(k) grammar (cf. Corollary 3.2).

Clearly, given an indexed grammar G, it is not decidable whether there exists a k such that G is an indexed LL(k) grammar, for otherwise this question would be decidable in the contextfree case.

Furthermore, the construction used in the proof of Theorem 7.1 shows

Theorem 7.3. The classes of ILL(k) and strong ILL(k) languages coincide.

Abstract

The classes of indexed LL(*k*) grammars and strong indexed LL(*k*) grammars are defined. First the class of strong indexed LL(*k*) grammars is investigated. In particular it is shown that the strong indexed LL(*k*) property is decidable and that the class of strong indexed LL(*k*) languages is contained in the class of deterministic indexed languages. Furthermore it is proved that the deterministic contect-free languages coincide with the right linear strong indexed LL(1) languages and are a proper subclass of the strong indexed LL(1) languages. The remainder of the paper is devoted to proving the decidability of the (general) indexed LL(*k*) property. To prove this result, a general transformation of indexed grammars is introduced. This transformation unifies proof techniques used in the context-free and indexed areas.

PROF. DR. RAINER PARCHMANN
INSTITUT FÜR INFORMATIK, UNIVERSITÄT HANNOVER
WELFENGARTEN 1
D-3000 HANNOVER 1, WEST GERMANY

References

[1] AHO, A. V., "Indexed grammars", J. Assoc. Comput. Mach. 15 (1968), 647—671.
[2] FISCHER, M. J., "Grammars with macro-like productions", in IEEE Conference Record of the Ninth Annual Symposium on Switching and Automata Theory, (1969), 131—142.
[3] HARRISON, M. A., Introduction to Formal Language Theory, Addison Wesley, Reading, Mass., 1978.
[4] HOPCROFT, J. E. and ULLMAN, J. D., Introduction to Automata Theory, Languages and Computation, Addison Wesley, Reading, Mass., 1979.
[5] MAIBAUM, T. S. E., "Pumping lemmas for term languages", J. Comput. System Sci. 17 (1978), 319—330.
[6] MEHLHORN, K., "Parsing macro grammars top down", Inf. Contr. 40 (1979), 123—143.
[7] PARCHMANN, R., DUSKE, J. and SPECHT, J., "On deterministic indexed languages", Inform. Contr. 45 (1980), 48—67.
[8] PARCHMANN, R., DUSKE, J. and SPECHT, J., "Closure properties of deterministic indexed languages", Inform. Contr. 46 (1980), 200—218.
[9] ROSENKRANTZ, D. J. and STEARNS, R. E., "Properties of deterministic top-down grammars", Inform. Contr. 17 (1970), 226—256.
[10] SEBESTA, R. W. and JONES, N. D., "Parsers for indexed grammars", Internat. J. Comput. and Inform. Sci. 7 (1978), 345—359.
[11] WEISS, K., "Deterministische indizierte Grammatiken", Doctoral Dissertation, University of Karlsruhe, 1976.

(Received July 11, 1983)

On ν_1-products of commutative automata

By F. GÉCSEG

The aim of this paper is to show the existence of commutative automata such that each of them forms a homomorphically complete system with respect to the ν_1-product in the class of all commutative automata.

By an *automaton* we mean a system $\mathbf{A}=(X, A; \delta)$, where X is a nonvoid finite set of *input signals, A* is a nonvoid finite set of *states*, and δ: $A\times X\to A$ is the *transition function*. The automaton $\mathbf{A}$ is *commutative* if $\delta(a, xy)=\delta(a, yx)$ holds for arbitrary $a\in A$ and $x, y\in X$. (The transition $\delta(a, p)$ $(a\in A, p\in X^*)$ is defined by $\delta(a, e)=$ $=a$ and $\delta(a, qx)=\delta(\delta(a, q), x)$ $(a\in A, q\in X^*, x\in X)$, where X^* is the set of all finite words over X and e denotes the empty word.)

Since automata can be considered unary algebras (cf. for instance [5]) the concept of a subautomaton of an automaton, and those of isomorphism and homomorphism of automata can be defined in a natural way.

Let $\mathbf{A}_i=(X_i, A_i, \delta_i)$ $(i=1, ..., k)$ be a system of automata, X a nonvoid finite set and φ: $A_1\times...\times A_k\times X\to X_1\times...\times X_k$ a function. Take the automaton $\mathbf{A}=$ $=(X, A, \delta)$ given by $A=A_1\times...\times A_k$ and $\delta((a_1, ..., a_k), x)=(\delta_1(a_1, x_1), ...$ $..., \delta_k(a_k, x_k))$ $((a_1, ..., a_k)\in A, x\in X)$, where $(x_1, ..., x_k)=\varphi(a_1, ..., a_k, x)=$ $=(\varphi_1(a_1, ..., a_k, x), ..., \varphi_k(a_1, ..., a_k, x))$. Then $\mathbf{A}$ is called the *product* of $\mathbf{A}_1, ..., \mathbf{A}_k$ *with respect to* X and φ, and we denote it by

$$\prod_{i=1}^{k} \mathbf{A}_i[X, \varphi].$$

Consider the above product $\mathbf{A}$, and take a non-negative integer i. We say that $\mathbf{A}$ is an α_i*-product* if for every t $(1\leqq t\leqq k)$, φ_t is independent of its j^{th} component $(1\leqq j\leqq k)$ whenever $t\geqq j+i$. Moreover, if for all t $(=1, ..., k)$, $(a_1, ..., a_k)\in A$ and $x\in X$, $\varphi_t(a_1, ..., a_k, x)$ may depend on x only then $\mathbf{A}$ is a *quasi-direct product*.

Again take the product $\mathbf{A}$ above. Moreover, let ν: $N_k\to\mathfrak{P}(N_k)$ be a mapping, where N_k is the set of the first k positive integers and $\mathfrak{P}$ is the powerset-operator. If i is a non-negative integer such that for every $t\in N_k$, $|\nu(t)|\leqq i$ and φ_t is independent of its j^{th} component $(1\leqq j\leqq k)$ whenever $j\notin\nu(t)$ then $\mathbf{A}$ is called a ν_i*-product* (see [1]). If $\mathbf{A}_1=...=\mathbf{A}_k=\mathbf{B}$ then $\mathbf{A}$ is a ν_i*-power* of $\mathbf{B}$. Moreover, in φ_t we shall indicate only those variables,on which it may depend. Finally, for the ν_i-product $\mathbf{A}$ we shall use the notation

$$\mathbf{A} = \prod_{t=1}^{k} \mathbf{A}_t[X, \varphi, \nu].$$

Let $\mathscr{K}$ be a class of automata. Then
$\mathbf{H}(\mathscr{K})$: homomorphic images of automata from $\mathscr{K}$.
$\mathbf{S}(\mathscr{K})$: subautomata of automata from $\mathscr{K}$.
$\mathbf{Q}(\mathscr{K})$: quasi-direct products of automata from $\mathscr{K}$.
$\mathbf{P}_{\nu_i}(\mathscr{K})$: ν_i-products of automata from $\mathscr{K}$.

For every prime number p consider the automata $\mathbf{A}_p=(X_p, A_p, \delta_p)$, where $X_p=\{x_0, x_1, ..., x_{p-1}\}$, $A_p=\{0, 1, ..., p-1\}$ and $\delta_p(i, x_j)=i\oplus_p j$, where $0\leqq i,j<p$ and $\oplus_p$ denotes the modulo p addition. Obviously, each $\mathbf{A}_p$ is a commutative automaton.

We are now ready to state and prove the following

Theorem. For an arbitrary commutative automaton A and for every prime number p the inclusion $\mathbf{A}\in\mathbf{HSP}_{\nu_1}(\{\mathbf{A}_p\})$ holds.

Proof. For every prime number p and every positive integer n take the automaton $\mathbf{B}_{(p,n)}=(X, B_{(p,n)}, \delta_{(p,n)})$ with $X=\{x, y\}$, $B_{(p,n)}=\{0, 1, ..., p^n-1\}$, $\delta_{(p,n)}(i, x)=i\oplus_{p^n}1$ and $\delta_{(p,n)}(i, y)=i$ $(i=0, 1, ..., p^n-1)$. Moreover, for every natural number n let $\mathbf{E}_n=(\{x, y\}, \{0, 1, ..., n\}, \delta_n)$ be the $n+1$ state *elevator*, that is the automaton with $\delta_n(i, y)=i$ $(i=0, ..., n)$ and

$$\delta_n(i, x) = \begin{cases} i+1 & \text{if} \quad 0 \leqq i < n, \\ n & \text{if} \quad i = n. \end{cases}$$

Denote by $\mathscr{K}$ the class of all $\mathbf{B}_{(p,n)}$ and $\mathbf{E}_n$. In [3] it is shown that $\mathbf{HSQ}(\mathscr{K})$ is the class of all commutative automata. Since every quasi-direct product of ν_i-products of automata is isomorphic to a ν_i-product of the same automata in order to prove our theorem it is enough to show that for arbitrary prime numbers p, q and positive integer n the inclusions $\mathbf{B}_{(q,n)}\in\mathbf{HSP}_{\nu_1}(\{\mathbf{A}_p\})$ and $\mathbf{E}_n\in\mathbf{HSP}_{\nu_1}(\{\mathbf{A}_p\})$ hold. We start with the proof of $\mathbf{B}_{(q,n)}\in\mathbf{HSP}_{\nu_1}(\{\mathbf{A}_p\})$. Let us fix p, q and n. We distinguish the following two cases.

Case 1. $p\neq q$. Then q^n divides p^m-1 for some $m>0$. (We may also assume that $m>1$.) Therefore, it is sufficient to show the existence of a ν_1-power $\mathbf{B}=(X, B, \delta)$ of $\mathbf{A}_p$ such that $\mathbf{B}$ contains a subautomaton which is a cycle of length p^m-1 under x, and y induces the identity mapping of this subautomaton.

Let $\mathbf{B}=(X, B, \delta)=(\underbrace{\mathbf{A}_p\times...\times\mathbf{A}_p}_{p^m-1 \text{ times}})[X, \varphi, \nu]$ be the ν_1-product, where for every $i\in\{1, ..., p^m-1\}$

$$\nu(i) = \begin{cases} i-1 & \text{if} \quad i > 1, \\ p^m-1 & \text{if} \quad i = 1, \end{cases}$$

and for arbitrary $i\in\{1, ..., p^m-1\}$ and $j\in\{0, ..., p-1\}$

$$\varphi_i(j, z) = \begin{cases} x_j & \text{if} \quad z = x, \\ x_0 & \text{if} \quad z = y. \end{cases}$$

Take a $b=(b_1, b_2, ..., b_{p^m-1})$ from B. Then, by the definition of $\mathbf{B}$, we obviously have $\delta(b, x)=(b_1\oplus_p b_{p^m-1}, b_2\oplus_p b_1, ..., b_{p^m-1}\oplus_p b_{p^m-2})$.

In the rest of the paper all multiplications of integers, and all the binomial coefficients $\binom{k}{l}\left(=\frac{k!}{l!(k-l)!}\right)$ are taken modulo p. Moreover, $\oplus$ and $\ominus$ will stand for the modulo p addition and modulo p substraction, respectivelly. Finally, we denote p^m-1 by t.

One can easily show, by induction on k, that

$$\delta(b, x^k) = \left(\binom{k}{0} b_1 \oplus \binom{k}{1} b_t \oplus \binom{k}{2} b_{t-1} \oplus \dots \oplus \binom{k}{k} b_{t-k+1},\right.$$

$$\binom{k}{0} b_2 \oplus \binom{k}{1} b_1 \oplus \binom{k}{2} b_t \oplus \binom{k}{3} b_{t-1} \oplus \dots \oplus \binom{k}{k} b_{t-k+2}, \dots$$

$$\left.\dots, \binom{k}{0} b_t \oplus \binom{k}{1} b_{t-1} \oplus \dots \oplus \binom{k}{k} b_{t-k}\right)$$

if $1 \leqq k < t$, and

$$\delta(b, x^t) = \left(\left(\binom{t}{0} \oplus \binom{t}{t}\right) b_1 \oplus \binom{t}{1} b_t \oplus \binom{t}{2} b_{t-1} \oplus \dots \oplus \binom{t}{t-1} b_2,\right.$$

$$\left(\binom{t}{0} \oplus \binom{t}{t}\right) b_2 \oplus \binom{t}{1} b_1 \oplus \binom{t}{2} b_t \oplus \binom{t}{3} b_{t-1} \oplus \dots \oplus \binom{t}{t-1} b_3, \dots$$

$$\left.\dots, \left(\binom{t}{0} \oplus \binom{t}{t}\right) b_t \oplus \binom{t}{1} b_{t-1} \oplus \dots \oplus \binom{t}{t-1} b_1\right).$$

We would like to find a b such that $\delta(b, x^t) = b$ holds, i.e.,

$$\left(\binom{t}{0} \oplus \binom{t}{t}\right) b_1 \oplus \binom{t}{1} b_t \oplus \binom{t}{2} b_{t-1} \oplus \dots \oplus \binom{t}{t-1} b_2 = b_1,$$

$$\left(\binom{t}{0} \oplus \binom{t}{t}\right) b_2 \oplus \binom{t}{1} b_1 \oplus \binom{t}{2} b_t \oplus \dots \oplus \binom{t}{t-1} b_3 = b_2$$

$$(*) \qquad \vdots$$

$$\left(\binom{t}{0} + \binom{t}{t}\right) b_t \oplus \binom{t}{1} b_{t-1} \oplus \binom{t}{2} b_{t-2} \oplus \dots \oplus \binom{t}{t-1} b_1 = b_t$$

holds. Let us consider (*) a system of equations over the prime field $\{0, 1, \dots, p-1\}$ with modulo p addition and modulo p multiplication, where $b_1, b_2, \dots, b_t$ are unknowns. Add (modulo p) $(p-1)b_i$ to both sides of the i^{th} equation in (*) for every i $(=1, \dots, t)$. Then we get the linear homogeneous system of equations

$$\binom{t}{0} b_1 \oplus \binom{t}{1} b_t \oplus \binom{t}{2} b_{t-1} \oplus \dots \oplus \binom{t}{t-1} b_2 = 0,$$

$$\binom{t}{0} b_2 \oplus \binom{t}{1} b_1 \oplus \binom{t}{2} b_t \oplus \dots \oplus \binom{t}{t-1} b_3 = 0,$$

$$(**) \qquad \vdots$$

$$\binom{t}{0} b_t \oplus \binom{t}{1} b_{t-1} \oplus \binom{t}{2} b_{t-2} \oplus \dots \oplus \binom{t}{t-1} b_1 = 0.$$

Using the congruence $\binom{t+1}{l}\equiv 0 \pmod p$ $(1\leqq l\leqq t)$, one can easily show that for every l $(=0, 1, \dots, t-1)$, $\binom{t}{l}\equiv 1 \pmod p$ if l is even, and $\binom{t}{l}\equiv p-1 \pmod p$ if l is odd. (Therefore, the determinant of (**) is 0, consequently (**) has a nontrivial solution.) It can be seen immediately that $b_1=1$, $b_2=1$, $b_3=0, \dots, b_t=0$ is a solution of (**). Moreover, by the construction of **B**, the states b, $\delta(b, x), \dots, \delta(b, x^{t-1})$ are pairwise distinct, that is they form a cycle of length t $(=p^m-1)$ under x.

Case 2. $p=q$. We now show the existence of a ν_1-power $\mathbf{B}=(X, B, \delta)$ of $\mathbf{A}_p$ such that **B** contains a subautomaton which is a cycle of length p^n under x, and y induces the identity mapping on this subautomaton.

Let $\mathbf{B}=(X, B, \delta)=(\underbrace{\mathbf{A}_p\times\dots\times\mathbf{A}_p}_{p^n \text{ times}})[X, \varphi, \nu]$ be the ν_1-product given in the following way. For every $i\in\{1, \dots, p^n\}$

$$\nu(i)=\begin{cases} i-1 & \text{if } i>1,\\ \emptyset & \text{if } i=1,\end{cases}$$

and for arbitrary $i\in\{2, \dots, p^n\}$ and $j\in\{0, \dots, p-1\}$

$$\varphi_i(j, z)=\begin{cases} x_j & \text{if } z=x,\\ x_0 & \text{if } z=y,\end{cases}$$

moreover, $\varphi_1(x)=\varphi_1(y)=x_0$.

Take the state $b=(1, 0, \dots, 0)$ of **B**. One can prove easily, by induction on k, that for every k $(=1, \dots, p^n-1)$

$$\delta(b, x^k)=\left(\binom{k}{0}, \binom{k}{1}, \dots, \binom{k}{k}, 0, \dots, 0\right).$$

Therefore

$$\delta(b, x^{p^n})=\left(\binom{p^n}{0}, \binom{p^n}{1}, \dots, \binom{p^n}{p^n-1}\right).$$

As it has been noted, for every k $(=1, \dots, p^n-1)$, $\binom{p^n}{k}\equiv 0 \pmod p$. Thus $\delta(b, x^{p^n})=b$ showing that the states b, $\delta(b, x), \dots, \delta(b, x^{p^n-1})$ form a desired cycle.

To prove that for arbitrary natural number n and prime number p the inclusion $\mathbf{E}_n\in\mathbf{HSP}_{\nu_1}(\{\mathbf{A}_p\})$ holds take the automaton

$$\mathbf{B}=(X, B, \delta)=(\underbrace{\mathbf{A}_p\times\dots\times\mathbf{A}_p}_{p^n \text{ times}})[X, \varphi, \nu]$$

defined in the same way as in Case 2 with the following exceptions: $\nu(1)=p^n$ and $\varphi_1(j, x)=x_p\Theta_j$ $(j=0, \dots, p-1)$. Again take $b=(1, 0, \dots, 0)$. Then, like in Case 2, for every k $(=1, \dots, p^n-1)$

$$\delta(b, x^k)=\left(\binom{k}{0}, \binom{k}{1}, \dots, \binom{k}{k}, 0, \dots, 0\right).$$

Therefore

$$\delta(b, x^{p^n})=(0, 0, \dots, 0)$$

showing that the states $b, \delta(b, x), \ldots, \delta(b, x^{p^n})$ form a p^n+1 state elevator. Since $p^n > n$ this ends the proof of the Theorem.

Remark. It follows from Theorem 3 in [4] that there exists no finite system of automata which is homomorphically complete with respect to the α_1-product in the class of all commutative automata. Thus, by the Theorem above, in this respect the ν_1-product is more powerful than the α_1-product. (By the Theorem in [2], the ν_1-product is not stronger than any of the α_i-products if $i > 1$.)

Acknowledgements

The author wants to thank Dr. B. Imreh for detecting a gap in the first version of the proof of the Theorem.

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

References

[1] DÖMÖSI, P., IMREH, B., On ν_i-products of automata, Acta Cybernet., v. VI. 2, 1983, pp. 149—162.

[2] ÉSIK, Z., HORVÁTH, GY., The α_2-product is homomorphically general, Papers on Automata Theory, v. V. 1983, pp. 49—62.

[3] GÉCSEG, F., On subdirect representations of finite commutative unoids, Acta Sci. Math. (Szeged), v. 36, 1974, pp. 33—38.

[4] GÉCSEG, F., On products of abstract automata, Acta Sci. Math. (Szeged), v. 38, 1976, pp. 21—43.

[5] GÉCSEG, F., STEINBY, M., Tree automata, Akadémiai Kiadó, Budapest, 1984.

(Received Febr. 1, 1984)

On finite definite automata

By B. IMREH

In this paper we consider the isomorphic realizations of finite definite automata. First we present a characterization of finite subdirectly irreducible definite automata. Secondly we give necessary and sufficient conditions for a system of automata to be isomorphically complete for the class of all finite definite automata with respect to the α_i-products. It will turn out that every finite definite automaton can be embedded isomorphically into an α_0-product of reset automata with two states.

By automaton we always mean a finite automaton without output. Since an automaton can be considered a unoid the notions such as subautomaton, isomorphism, embedding, homomorphism, congruence relation can be introduced in a natural way. Further on we shall use the following notations: X^* denotes the free monoid generated by X, $|p|$ denotes the length of the word $p \in X^*$, if there is no danger of confusion then we use the convenient notation ax and ap for $\delta(a, x)$ and $\delta(a, p)$ respectively.

An automaton $\mathbf{A}=(X, A, \delta)$ is called *definite* if there exists a natural number n such that $|p| \geqq n$ implies $|Ap|=1$ for any $p \in X^*$ where $Ap=\{ap\colon a \in A\}$. Specially, if $n=1$ then $\mathbf{A}$ is called a *reset* automaton, furthermore if there exists a state $a_0 \in A$ such that $|p| \geqq n$ implies $Ap=\{a_0\}$ for any $p \in X^*$ then $\mathbf{A}$ is *nilpotent.*

In the paper [3] we gave a characterization of finite subdirectly irreducible nilpotent automata. Now we generalize this result for definite automata. Namely, it holds the following

Theorem 1. A definite automaton $\mathbf{A}=(X, A, \delta)$ $(|A|>2)$ is subdirectly irreducible if and only if $\mathbf{A}$ has two different states a_0, b_0 such that

(i) $a_0x=b_0x$ holds for any $x \in X$,

(ii) for any $a, b \in A$ if $a \neq b$ and $\{a, b\} \nsubseteq \{a_0, b_0\}$ then there exists an input sign $x \in X$ with $ax \neq bx$.

Proof. In order to prove the necessity assume that $\mathbf{A}$ is subdirectly irreducible. Consider the set B of all subsets of A with two elements and define the relation $\leqq$ on B in the following way: for any $\{a, b\}$, $\{c, d\} \in B$ $\{a, b\} \leqq \{c, d\}$ if and only if there exists a word $p \in X^*$ such that $\{a, b\}p=\{c, d\}$. Since $\mathbf{A}$ is definite the defined relation is antisymmetric and thus, it is a partial ordering on B. Now we shall show that there is a greatest element in B. Because of finiteness it is enough to show that there exists only one maximal element in B. Assume to the contrary that $\{a, b\}$ and $\{c, d\}$ $\{a, b\}\neq\{c, d\}$, $\{a, b\}$, $\{c, d\}\in B$ are maximal in B. Then $ax=bx$ and $cx=dx$ hold for any $x\in X$. Consider the following relations on A: for any $u, v\in A$

$u\varrho v$ if and only if $\{u, v\}\subseteqq\{a, b\}$ or $u=v$,

$u\sigma v$ if and only if $\{u, v\}\subseteqq\{c, d\}$ or $u=v$.

It is obvious that ϱ and σ are nontrivial congruence relations of **A** and $\varrho\cap\sigma=\Delta_A$ where Δ_A denotes the equality relation on **A**. This yields that **A** is subdirectly reducible which contradicts our assumption on **A**. Therefore, B has a greatest element. Let $\{a_0, b_0\}$ denote this one. Then it is obvious that conditions (i) and (ii) are satisfied.

To prove the sufficiency assume that (i) and (ii) are satisfied by a definite automaton $\mathbf{A}=(X, A, \delta)$. Consider again the set B with its ordering defined above. From conditions (i) and (ii) it follows that $\{a_0, b_0\}$ is the greatest element of B. Take the following relation on A: for any $u, v\in A$

$u\theta v$ if and only if $\{u\ v\}\subseteqq\{a_0, b_0\}$ or $u=v$.

By (i) we obtain that θ is congruence relation. Next we shall show that θ is the smallest nontrivial congruence of **A**. Indeed, let ϱ denote an arbitrary nontrivial congruence relation of **A**. Then there exist $a, b\in A$ such that $a\neq b$ and $a\varrho b$, furthermore, there is a word $p\in X^*$ such that $\{a, b\}p=\{a_0, b_0\}$ since $\{a_0, b_0\}$ is the greatest element in B. But then $a_0\varrho b_0$ also holds which implies $\theta\leqq\varrho$. On the other hand, it is known that the existence of the smallest nontrivial congruence implies the subdirect irreducibility which completes the proof of Theorem 1.

Remark 1. Using Theorem 1 we obtain a simple algorithm to decide whether a definite automaton is subdirectly irreducible. Namely, if the automaton is given by a transition table of which rows correspond to each input sign and columns correspond to each state then we have the following criterion.

A definite automaton is subdirectly irreducible if and only if there exist two equal columns in its table and leaving one of them, the columns of the remained table are pairwise different.

Remark 2. In [5] M. Katsura introduced the following family of strongly connected definite automata. Let $X=\{x, y\}$ and denote by $1<p_1<p_2<\dots$ the sequence of all prime numbers. Then any natural number $n>1$ can be written uniquely as $p_1^{e_1}\dots p_{r-1}^{e_{r-1}}p_r^{e_r}$ where $e_{r-1}\geqq 1$ and $e_r=0$. Let $\mathbf{A}_n=(X, A_n, \delta_n)$ where $A_n=\{a_0\}\cup\cup\{a_{ij}\colon 1\leqq i\leqq r;\ 0\leqq j\leqq e_i\}$ and

$$\delta_n(a_{ij}, x)=\begin{cases} a_{ij+1} & \text{if } 1\leqq i\leqq r-1 \text{ and } 0\leqq j\leqq e_i-1,\\ a_0 & \text{otherwise,}\end{cases}$$

$$\delta_n(a_{ij}, y)=\begin{cases} a_{i+1,0} & \text{if } j=0 \text{ and } 1\leqq i\leqq r-1,\\ a_{r0} & \text{if } j=0 \text{ and } i=r,\\ a_{10} & \text{otherwise,}\end{cases}$$

$$\delta_n(a_0, x)=a_0,\ \delta_n(a_0, y)=a_{10}.$$

From the definition of $\mathbf{A}_n$ it follows that if $n=p_1^{e_1}\dots p_r^{e_r}$ and $e_i\geqq 1$, $e_j\geqq 1$ hold for some $1\leqq i\neq j\leqq r$ then $\{a_0, a_{ie_i}; a_{je_j}\}x=a_0$ and $\{a_0, a_{ie_i}, a_{je_j}\}y=a_{10}$ and thus, by Theorem 1, we obtain that $\mathbf{A}_n$ is subdirectly reducible. On the other hand, if $n=p_1^0\dots p_{i-1}^0 p_i^{e_i} p_{i+1}^0 (e_i\geqq 1)$ is a prime-power then $\{a_0, a_{ie_i}\}x=a_0$, $\{a_0, a_{ie_i}\}y=a_{10}$ and, by a simple computation, it can be seen that $|\{u, v\}x|=2$ or $|\{u, v\}y|=2$ holds for any elements $u, v\in A$ such that $u\neq v$ and $\{u, v\}\subsetneqq\{a_0, a_{ie_i}\}$. Therefore, by Theorem 1, we get that $\mathbf{A}_n$ is subdirectly irreducible. Summarizing, we have the following statement.

For any natural number $n>1$ the automaton $\mathbf{A}_n$ is subdirectly irreducible if and only if n is prime-power.

Remark 3. A well-known special type of definite automata is the shift register. It was investigated in papers [4] and [6]. Now we consider the subdirect irreducibility of these automata. For this reason let $n\geqq 1$ be an arbitrary natural number and X a non-empty finite set with at least two elements. Then the automaton $\mathbf{A}(X, n)=(X, X^n, \delta_n)$ is called a *shift register* where $\delta_n(x_1\dots x_n, x)=x_2\dots x_n x$ for any $x_1\dots x_n\in\in X^n$ and $x\in X$. Observe that $\delta_n(ux_2\dots x_n, x)=\delta_n(vx_2\dots x_n, x)$ holds for any words $ux_2\dots x_n, vx_2\dots x_n\in X^n$ and $x\in X$. From this it follows that conditions of Theorem 1 are not satisfied. Thus we have the following assertion.

Every shift register with at least three elements is subdirectly reducible.

Next we shall study the α_i-products (see [1] or [2]) from the point of view of isomorphic completeness for the class of all definite automata. For this reason let i be a nonnegative integer, furthermore, let Σ be an arbitrary system of automata. Σ is called *isomorphically complete* for the class of all definite automata with respect to the α_i-product if any definite automaton can be embedded isomorphically into an α_i-product of automata from Σ. We are going to use the following obvious statement.

Lemma. If an automaton $\mathbf{A}$ can be embedded isomorphically into an α_0-product of automata $\mathbf{A}_t$ $(t=1, \dots, k)$ and for some $1\leqq j\leqq k$ the automaton $\mathbf{A}_j$ can be embedded into an α_0-product of automata $\mathbf{B}_m$ $(m=1, \dots, s)$ then the automaton $\mathbf{A}$ can be embedded isomorphically into an α_0-product of automata $\mathbf{A}_1, \dots, \dots, \mathbf{A}_{j-1}, \mathbf{B}_1, \dots, \mathbf{B}_s, \mathbf{A}_{j+1}, \dots, \mathbf{A}_k$.

Concerning the isomorphic realization of definite automata with respect to the α_0-product we have the following result.

Theorem 2. A system Σ of automata is isomorphically complete for the class of all definite automata with respect to the α_0-product if and only if Σ contains an automaton which has two different states a, b and two input signs x, y such that $ax=bx=b$ and $ay=by=a$ hold.

Proof. In order to prove the necessity take the definite automaton $\mathbf{U}=(\{x, y\}, \{a, b\}, \delta)$ where $\delta(a, x)=\delta(b, x)=b$ and $\delta(a, y)=\delta(b, y)=a$. Because of the isomorphic completeness of Σ there exists an α_0-product $\prod_{j=1}^{k} \mathbf{A}_j(X, \varphi)$ of automata from Σ such that $\mathbf{U}$ can be embedded isomorphically into this product. Let $(a_1, \dots, \dots, a_k)$, $(a_1', \dots, a_k')$ denote the images of the elements a, b under a suitable isomorphism. Then among the sets $\{a_t, a_t'\}$ $(t=1, \dots, k)$ there should be at least one which has more then one element. Let r be the least index for which $a_r\neq a_r'$. It is obvious that the automaton $\mathbf{A}_r\in\Sigma$ satisfies the condition.

To prove the sufficiency assume that the automaton $\mathbf{B}\in\Sigma$ has the suitable states and input signs. To verify the completeness it is enough to show that any definite automaton can be embedded isomorphically into an α_0-product of reset automata with two states since any reset automaton with two states can be embedded isomorphically into an α_0-product of $\mathbf{B}$ with a single factor, and thus, by our Lemma, we obtain the completeness of Σ.

We prove by induction on the number of states of the automaton. In the case $n\leqq 2$ our statement is trivial. Now let $n>2$ and suppose that the statement is valid for any $m<n$. Let $\mathbf{A}=(X, A, \delta_A)$ be an arbitrary definite automaton with n states. If $\mathbf{A}$ is subdirectly reducible then $\mathbf{A}$ can be embedded isomorphically into a direct product of definite automata with fewer states than n. Therefore, by our induction hypothesis and Lemma, the statement is valid. Now assume that $\mathbf{A}$ is subdirectly irreducible. Let $A=\{a_1, \ldots, a_n\}$. Then from Theorem 1 it follows that there exist $a_i, a_j\in A$ $(i\neq j)$ such that $a_i x=a_j x$ holds for any $x\in X$. Without loss of generality we may assume that $i=n-1$ and $j=n$. Define the relation ϱ on A as follows: for any $a_r, a_s\in A$

$$a_r\varrho a_s \text{ if and only if } \{a_r, a_s\}\subseteq\{a_{n-1}, a_n\} \text{ or } a_r=a_s.$$

Obviously ϱ is a congruence relation of $\mathbf{A}$. Then the quotient automaton $\mathbf{A}/\varrho$ is definite and $A/\varrho=\{\{a_1\},\ldots, \{a_{n-2}\}, \{a_{n-1}, a_n\}\}$. Now let c be an arbitrary symbol $(c\notin A)$ and take the automaton $\mathbf{C}=(A/\varrho\times X, \{c, a_{n-1}, a_n\}, \delta)$ where

$$\delta(u, (\{a_i\}, x))=\begin{cases}\delta_{\mathbf{A}}(a_i, x) & \text{if} \quad \delta_{\mathbf{A}}(a_i, x)\in\{a_{n-1}, a_n\},\\ c & \text{otherwise,}\end{cases}$$

$$\delta(u, (\{a_{n-1}, a_n\}, x))=\begin{cases}\delta_{\mathbf{A}}(a_n, x) & \text{if} \quad \delta_{\mathbf{A}}(a_n, x)\in\{a_{n-1}, a_n\},\\ c & \text{otherwise,}\end{cases}$$

for any $u\in\{c, a_{n-1}, a_n\}$, $(\{a_i\}, x)\in\{\{a_1\}, \ldots, \{a_{n-2}\}\}\times X$, $(\{a_{n-1}, a_n\}, x)\in\{\{a_{n-1}, a_n\}\}\times X$. Now consider the α_0-product $\mathbf{A}/\varrho\times\mathbf{C}(X, \varphi)$ where $\varphi_1(x)=x$ for any $x\in X$ and $\varphi_2(\{a_i\}, x)=(\{a_i\}, x)$, $\varphi_2(\{a_{n-1}, a_n\}, x)=(\{a_{n-1}, a_n\}, x)$ for any $x\in X$ and $1\leqq i\leqq n-2$. Then it is not difficult to see that the correspondence

$$\nu(a_i)=\begin{cases}(\{a_i\}, c) & \text{if} \quad 1\leqq i\leqq n-2,\\ (\{a_{n-1}, a_n\}, a_i) & \text{if} \quad i\in\{n-1, n-2\},\end{cases}$$

is an isomorphism of $\mathbf{A}$ into the α_0-product $\mathbf{A}/\varrho\times\mathbf{C}(X, \varphi)$. On the other hand, observe that $\mathbf{C}$ is a reset automaton and it is a well-known fact that any reset automaton can be embedded isomorphically into a direct product of reset automata with two states. Therefore, by our induction hypothesis and Lemma, we have a required decomposition of $\mathbf{A}$. This completes the proof of Theorem 2.

Regarding α_i-products with $i\geqq 1$ we have the following statement.

Theorem 3. A system Σ of automata is isomorphically complete for the class of all definite automata with respect to the α_i-product $(i\geqq 1)$ if and only if Σ contains an automaton which has two different states a, b and four input signs v, x, y, z (need not be different) such that $av=bx=b$ and $by=az=a$ hold.

Proof. The necessity of the condition is obvious. To prove the sufficiency, by Theorem 2, it is enough to show that an α_0-product of α_1-products with single factors is an α_1-product which follows from the definition of the α_i-products.

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

References

[1] GÉCSEG, F., Composition of automata, Proceedings of the 2nd Colloquium on Automata, Languages and Programming, Saarbrücken, Springer Lecture Notes in Computer Science v. 14, 1974, pp. 351—363.

[2] IMREH, B., On α_i-products of automata, *Acta Cybernet.*, v. 3, 1978, pp. 301—307.

[3] IMREH, B., On finite nilpotent automata, *Acta Cybernet.*, v. 5, 1981, pp. 281—293.

[4] ITO, M. and DUSKE, J., On cofinal and definite automata, *Acta Cybernet.*, v. 6, 1983, pp. 181—189.

[5] KATSURA, M., A characterization of partially ordered sets of automata, *Papers on Automata Theory IV*, K. Marx Univ. of Economics, Dept. of Math., Budapest, 1982, No. DM 82-1, pp. 45—57.

[6] STOKLOSA, J., On operation preserving functions of shift registers, *Found. Control. Engrg.*, v. 2, 1977, pp. 211—214.

(Received Dec. 21, 1983)

On involutorial automata and involutorial events

By M. ITO and J. DUSKE

1. Basic notions and facts

By an *automaton* we mean a triple $\mathcal{A}=(A, X, \delta)$, where A and X are finite nonempty sets, the set of *states* and the set of *inputs* of $\mathcal{A}$, and $\delta\colon A\times X\to A$ is a function, the *transition function* of $\mathcal{A}$.

Let X^* be the free monoid generated by X with identity element e. Then δ is assumed to be extended to $A\times X^*$ in the usual way. A finite nonempty set is called an *alphabet*. Each subset $E\subseteq X^*$ is called an *event* or a *language* over the alphabet X. Let us now define the notion of an involutorial automaton.

1.1. Definition. An automaton $\mathcal{A}=(A, X, \delta)$ is called *involutorial* iff $\delta(a, xx)=$ $=a$ holds for all $a\in A$, $x\in X$.

The following lemma is a trivial but useful one.

1.2. Lemma. Let $\mathcal{A}=(A, X, \delta)$ be cyclic and involutorial. Then $\mathcal{A}$ is strongly connected.

(For the automata-theoretic notions not defined in this paper see [6] and [3].)

Let $\mathcal{A}=(A, X, \delta)$ be an arbitrary automaton and define the congruence ϱ on X^* by:

$$\forall w_1, w_2\in X^*\colon (w_1, w_2)\in\varrho \quad \text{iff} \quad \delta(a, w_1)=\delta(a, w_2) \quad \text{for all} \quad a\in A.$$

The quotient $S(\mathcal{A})=X^*/\varrho$ is called the *characteristic semigroup* of $\mathcal{A}$. Let us use the notation $[w]_\varrho$ for the set $\{w'|(w', w)\in\varrho\}$. Concerning characteristic semigroups of involutorial automata, we can state:

1.3. Theorem. Let $\mathcal{A}=(A, X, \delta)$ be an involutorial automaton. Then $S(\mathcal{A})$ is an involutorial generated group.

Proof. Let $x\in X$. Then $(xx, e)\in\varrho$, therefore $[x]_\varrho[x]_\varrho=[e]_\varrho$, hence $[x]_\varrho$ is an involutorial element of $S(\mathcal{A})$. Now let $[w]_\varrho\in S(\mathcal{A})$ with $w=x_1\dots x_n$. Denote $x_n\dots$ $\dots x_1$ by w^R. Then $[w]_\varrho[w^R]_\varrho=[w^R]_\varrho[w]_\varrho=[e]_\varrho$.

With the aid of the following well known lemma, automata, which are involutorial and commutative, can be characterized.

1.4. Lemma. A group, in which each element is involutorial, is an abelian group.

1.5. Theorem. An automaton $\mathcal{A}=(A, X, \delta)$ is involutorial and commutative iff $\delta(a, ww)=a$ holds for all $a\in A$ and $w\in X^*$.

1.6. Example. An important example of an involutorial automaton is the *T-Flip-Flop (trigger)* $\mathcal{T}=(\{0, 1\}, \{0, 1\}, \delta)$ with $\delta(z, 0)=z$ and $\delta(z, 1)=\bar{z}$ for all $z\in\{0, 1\}$. Here $\bar{0}=1$ and $\bar{1}=0$. $\mathcal{T}$ is involutorial and commutative.

If we generalize the notion of a trigger, we arrive at the following: Let X be an alphabet and let $y\in X$. Set $\mathcal{T}_y^X=(\{0, 1\}, X, \delta_y^X)$ with

$$\delta_y^X(z, x) = z \quad \text{for all} \quad z\in\{0, 1\} \quad \text{and} \quad x\in X \quad \text{with} \quad x \neq y \quad \text{and}$$

$$\delta_y^X(z, y) = \bar{z} \quad \text{for all} \quad z\in\{0, 1\}.$$

We will call these automata generalized triggers.

If we now specialize the proof of Theorem 1 in [5] to the involutorial and commutative case, we obtain:

1.7. Theorem [Gécseg]. Every commutative involutorial automaton is the homomorphic image of a subdirect product of finitely many generalized triggers.

We can characterize commutative involutorial automata from another point of view. It is easy to see that every commutative involutorial automaton is a finite direct sum of cyclic commutative involutorial automata. From the basis theorem for abelian groups (see e.g. [12], p. 121) and Fleck's result ([4], Theorem 6), we obtain the following results, which were suggested by B. Imreh.

1.8. Theorem. Every cyclic commutative ivolutorial automaton is a one-state automaton or a direct product of finitely many two-state cyclic involutorial automata.

1.9. Corollary. Every cyclic commutative involutorial automaton has 2^n states, where n is a nonnegative integer.

2. The minimal involutorial congruence on X^*

Let X be an alphabet. A finite subset T of $X^*\times X^*$ is called a *Thue-system over* X^*. T defines a relation $\varrho_T\subseteq X^*\times X^*$ in the following way:

$$\forall v, w\in X^*\colon (v, w)\in\varrho_T \quad \text{iff} \quad v = v_1v_2v_3, \quad w = v_1w_2v_3 \quad \text{and} \quad (v_2, w_2)\in T$$
$$\text{or} \quad (w_2, v_2)\in T.$$

The congruence generated by the Thue-system T is the reflexive and transitive closure of ϱ_T.

Let us now consider the Thue-system $T=\{(aa, e)|a\in X\}$ over X^*. The relation ϱ_T will be denoted by $\xleftrightarrow{(i)}$ and the congruence generated by T will be denoted by $\xleftrightarrow{i}$ and called *minimal involutorial congruence on* X^*. Obviously, $X^*/\xleftrightarrow{i}$ is an involutorial generated group. In order to investigate the congruence $\xleftrightarrow{i}$, we will first establish some properties of the context-free language $L(G_i)$ generated by the context-free grammar $G_i=(\{S\}, X, P, S)$ with the set of productions $P=\{S\to e, S\to SS\}\cup\cup(S\to aSa|a\in X\}$.

(For the notions of formal language theory not defined in this paper see [1], [7], and [8].)

2.1. Lemma. Let G_i be given as above. Then we have for all $w_1, w_2, w, u_1, u_2 \in X^*$:
(1) If $w_1, w_2 \in L(G_i)$, then $w_1 w_2 \in L(G_i)$,
(2) If $w \in L(G_i)$, then $awa \in L(G_i)$ for all $a \in X$,
(3.a) If $w \in L(G_i)$, $w \neq e$, then $w = aw_1aw_2$ with $w_1, w_2 \in L(G_i)$ and $a \in X$,
(3.b) If $w \in L(G_i)$, $w \neq e$, then $w = w_1aw_2a$ with $w_1, w_2 \in L(G_i)$ and $a \in X$,
(4) If $u_1aau_2 \in L(G_i)$ with $a \in X$, then $u_1u_2 \in L(G_i)$ (involutorial cancellation),
(5) If $u_1u_2 \in L(G_i)$ then $u_1aau_2 \in L(G_i)$ for all $a \in X$ (involutorial extension).

Proof. (1) and (2) are trivial. To prove (3.a), consider a leftmost derivation $S \overset{*}{\Rightarrow} S^k \Rightarrow aSaS^{k-1} \overset{*}{\Rightarrow} \dots \overset{*}{\Rightarrow} aw_1aw_2 = w$ of w, and to prove (3.b), consider a rightmost derivation $S \overset{*}{\Rightarrow} S^k \Rightarrow S^{k-1}aSa \overset{*}{\Rightarrow} \dots \overset{*}{\Rightarrow} w_1aw_2a = w$ of w. Now let us consider (4). Let $S \Rightarrow v_1 \Rightarrow v_2 \Rightarrow \dots \Rightarrow v_n = u_1aau_2$ be a derivation of u_1aau_2. We will prove the assertion by induction on n. The assertion holds for $n=1$. Let $S \Rightarrow v_1 \Rightarrow v_2 \Rightarrow \dots \dots \Rightarrow v_{n+1} = u_1aau_2$ be a derivation of u_1aau_2 of length $n+1$. We have to consider the following cases:

(a) Let $S \Rightarrow v_1 = SS$. Then there exist derivations $S \overset{*}{\Rightarrow} r_1$ and $S \overset{*}{\Rightarrow} r_2$ of length n_1 and n_2 with $n_1, n_2 \leqq n$ and $r_1r_2 = u_1aau_2$. If aa occurs in r_1 or r_2, then the assertion follows from the induction hypothesis. It remains to consider the case $r_1 = u_1a$ and $r_2 = au_2$. Here the application of (3.a), (3.b) and (1), (2) yields the assertion.

(b) Let $S \Rightarrow v_1 = bSb$ with $b \in X$. Then there exists a derivation $S \overset{*}{\Rightarrow} w$ of length $n_1 \leqq n$ and $u_1aau_2 = bwb$ holds. The case $w=e$ is trivial, therefore let us assume $w \neq e$. If aa occurs in w, then the assertion follows from the induction hypothesis and (2). If aa are the first two letters of u_1aau_2 $(a=b, u_1=e)$, then a is the leftmost letter of w. According to (3.a) we have $w = aw_1aw_2$ with $w_1, w_2 \in L(G_i)$, and therefore $u_1u_2 = w_1aw_2a \in L(G_i)$. The case that aa are the last two letters of u_1aau_2 $(a=b, u_2=e)$ is proved similarly.

Now let us prove (5) by induction on the length of u_1u_2. The case $|u_1u_2| = 0$ is trivial, therefore let us assume $|u_1u_2| > 0$.

According to (3.a) we have $u_1u_2 = bw_1bw_2$ with $b \in X$ and $w_1, w_2 \in L(G_i)$. If $|u_1| \geqq |bw_1b|$, then $u_1 = bw_1br$ and $w_2 = ru_2$. From the induction hypothesis we conclude $raau_2 \in L(G_i)$, and therefore $u_1aau_2 = bw_1braau_2 \in L(G_i)$ for all $a \in X$.

If $|u_1| < |bw_1b|$, then $bw_1b = u_1r_2b$. The case $|u_1| = 0$ is trivial. Therefore assume $u_1 = br_1$. Then $w_1 = r_1r_2$, and with the aid of the induction hypothesis we conclude $r_1aar_2 \in L(G_i)$, and hence we have $u_1aau_2 = br_1aar_2bw_2 \in L(G_i)$ for all $a \in X$. This completes the proof of the lemma.

Now we can prove the following theorem.

2.2. Theorem. Let G_i be the context-free grammar given above and let $w = x_1x_2 \dots x_k \in X^*$ with $x_j \in X$ for $j \in [1: k]$, $x_{j-1} \neq x_j$ for $j \in [2: k]$, and $k \geqq 0$. If

(I) $w = w_0 \xleftrightarrow{(i)} w_1 \xleftrightarrow{(i)} w_2 \xleftrightarrow{(i)} \dots \xleftrightarrow{(i)} w_n = w'$ holds, where $n \geqq 0$, then we have $w' = \alpha_0x_1\alpha_1x_2\alpha_2 \dots \alpha_{k-1}x_k\alpha_k$ with $\alpha_i \in L(G_i)$ for $i \in [0: k]$.

Proof. The assertion holds for $n=0$ and $n=1$. Now let $j \in [0: n]$ be maximal with the following property:

(II) There are words $\alpha^j_0, \alpha^j_1, \ldots, \alpha^j_k \in L(G_i)$ such that

$$w_j = \alpha^j_0 x_1 \alpha^j_1 x_2 \alpha^j_2 \ldots \alpha^j_{k-1} x_k \alpha^j_k \quad \text{holds.}$$

We will show $j=n$. Assume $j<n$. Let us consider the following two cases:

(1) w_{j+1} can be derived from w_j by insertion of aa, $a\in X$ (involutorial extension). Then, according to (5) of 2.1. Lemma, (II) holds for $j+1$.

(2) w_{j+1} can be derived from w_j by deletion of aa, $a\in X$ (involutorial cancellation). If this aa can be chosen in an α^j_l, $l\in[0{:}\ k]$, then, according to (4) of 2.1. Lemma, (II) holds for $j+1$. It remains to consider $aa=x_lx_l$, $l\in[1{:}\ k]$, and

$$\alpha^j_{l-1}x_l\alpha^j_l = \overline{\alpha^j_{l-1}}x_lx_l\alpha^j_l \quad \text{or} \tag{2.1}$$

$$\alpha^j_{l-1}x_l\alpha^j_l = \alpha^j_{l-1}x_lx_l\overline{\alpha^j_l}, \tag{2.2}$$

and w_{j+1} can be derived from w_j by cancellation of the occurrences of x_lx_l in the right sides of (2.1) or (2.2).

(2.1) According to (3.b) of 2.1. Lemma, divide α^j_{l-1} in $\alpha^j_{l-1}=w_1x_lw_2x_l$ with $w_1, w_2\in L(G_i)$. Then we have $\overline{\alpha^j_{l-1}}\alpha^j_l=w_1x_lw_2\alpha^j_l$, and therefore (II) holds for $j+1$.

(2.2) According to (3.a) of 2.1. Lemma, divide α^j_l in $\alpha^j_l=x_lw_1x_lw_2$ with $w_1, w_2\in$ $\in L(G_i)$. Then we have $\alpha^j_{l-1}\overline{\alpha^j_l}=\alpha^j_{l-1}w_1x_lw_2$, and therefore (II) holds for $j+1$.

We must have $j=n$, which proves the theorem.

2.3. Corollary. Under the assumptions of 2.2. Theorem either $|w'|>|w|$ or $w=w'$ holds.

2.4. Corollary. The context-free language $L(G_i)$ is the congruence class of e w.r.t. $\xleftrightarrow{i}$.

Now let us consider the local regular language $X^*\setminus X^*VX^*$ with $V=\{xx|x\in X\}$. We have:

2.5. Corollary.

(1) If w is a word of minimal length in a congruence class of $\xleftrightarrow{i}$, then $w\in X^*\setminus$ $\setminus X^*VX^*$.

(2) If $w\in X^*\setminus X^*VX^*$, then w is a word of minimal length in a congruence class of $\xleftrightarrow{i}$.

(3) Two different words of $X^*\setminus X^*VX^*$ are in different congruence classes of $\xleftrightarrow{i}$.

2.6. Corollary. Each congruence class of $\xleftrightarrow{i}$ contains exactly one word of minimal length.

2.7. Corollary. If $X=\{x\}$ is a one element alphabet, then $\xleftrightarrow{i}$ contains exactly two classes, namely $\{e, x^2, x^4, \ldots\}$ and $\{x, x^3, x^5, \ldots\}$. If $|X|\geqq 2$, then the index of $\xleftrightarrow{i}$ is infinite.

We will denote the unique word of minimal length in the congruence class of w w.r.t. $\xleftrightarrow{i}$ by $w_{l\,min}$. If $K\subseteq X^*$, then we denote the set $\{w_{l\,\min}|w\in K\}$ by $L_{min}(K)$.

The function ϱ: $X^*\to X^*$ with $\varrho(w)=w_{l\,\min}$ for $w\in X^*$ is a *Dyck-simplification* in the sense of [10], which implies the following theorem.

2.8. Theorem. (Sakarovitch [10]). If $R\subseteq X^*$ is regular, then $L_{\min}(R)$ is regular.

3. Involutorial events and involutorial closure

In this section we will introduce and investigate involutorial events and involutorial closure. Let us first define these notions.

3.1. Definition. Let X be an alphabet and let $E \subseteq X^*$ be an event (subset of X^*). The set $E^i = \{u' | u' \in X^*, \exists u \in E$ with $u \xleftrightarrow{i} u'\}$ is called the *involutorial closure* of E. An event $E \subseteq X^*$ is called *involutorial* iff $E = E^i$. E is called *i-regular* iff E is involutorial and regular.

Obviously, for $E \subseteq X^*$ we have $E^i = L_{\min}(E)^i$, and for $E, K \subseteq X^*$ we have $E^i = K^i$ iff $L_{\min}(E) = L_{\min}(K)$.

Let $\mathscr{A} = (A, X, \delta, a_0, F)$ be a recognizer. Here (A, X, δ) is an automaton, $a_0 \in A$ is the initial state and $F \subseteq A$ is the set of final states of $\mathscr{A}$. $T(\mathscr{A}) = \{w | \delta(a_0, w) \in \in F\}$ is the event recognized by $\mathscr{A}$. $\mathscr{A}$ is called *involutorial* iff (A, X, δ) is an involutorial automaton.

Now let $E \subseteq X^*$ be an arbitrary event, and let $\equiv$ be the *Nerode right congruence associated with E,* i.e. right congruence defined by:

$$\forall v, w \in X^*: \ v \equiv w \quad \text{iff} \quad \forall u \in X^*: \ (vu \in E \quad \text{iff} \quad wu \in E).$$

E is regular iff $\equiv$ is of finite index (see e.g. [9])

We can now prove the following theorem.

3.2. Theorem. Let X be an alphabet and $E \subseteq X^*$ be an event. Then E is i-regular iff E is recognized by an involutorial recognizer $\mathscr{A}$.

Proof. If $E = T(\mathscr{A})$ for an involutorial recognizer $\mathscr{A}$, then obviously E is involutorial. Conversely, let E be i-regular. Since E is regular, the Nerode right congruence $\equiv$ of E is of finite index. Construct the recognizer $\mathscr{A} = (A, X, \delta, a_0, F)$ with $A = \{[w]_\equiv | w \in X^*\}$, $\delta([w]_\equiv, x) = [wx]_\equiv$, $a_0 = [e]_\equiv$, and $F = \{[w]_\equiv | w \in E\}$. We have $E = T(\mathscr{A})$. Since E is involutorial, the congruence $\xleftrightarrow{i}$ is contained in $\equiv$. Since $wxx \xleftrightarrow{i} w$ for all $w \in X^*$ and $x \in X$, we have $\delta([w]_\equiv, xx) = [wxx]_\equiv = [w]_\equiv$, i.e., $\mathscr{A}$ is involutorial.

The recognizer constructed in this proof is the *complete minimal (accessible and reduced) recognizer* (see [3]) which accepts the i-regular set E. Therefore we can state:

3.3. Corollary. If E is an i-regular set, then the complete minimal recognizer for E is involutorial.

3.4. Example. Let $\mathscr{A} = (A, \{x\}, \delta, a_0, \{a_2\})$ be a recognizer with

$$A = \{a_0, a_1, a_2\}, \quad \delta(a_0, x) = a_2, \quad \delta(a_2, x) = a_1, \quad \text{and} \quad \delta(a_1, x) = a_2.$$

Then $T(\mathscr{A}) = \{x, x^3, \ldots, x^{2n+1}, \ldots\}$ is involutorial, but $\mathscr{A}$ is not involutorial. The complete minimal recognizer for $T(\mathscr{A})$ is

$$\mathrm{A}_r = (\{a_0, a_1\}, \ \{x\}, \ \delta_r, \ a_0, \ \{a_1\}) \quad \text{with} \quad \delta_r(a_0, x) = a_1 \quad \text{and} \quad \delta_r(a_1, x) = a_0.$$

which is obviously involutorial.

The following two theorems state some closure and nonclosure properties of i-regular sets.

3.5. Theorem. Let X be an alphabet.

(1) The family of i-regular sets of X^* is a Boolean algebra of sets,

(2) If E is an i-regular set, then the transpose E^R of E is also an i-regular set. (Recall that the transpose of a word $w=a_1a_2\dots a_n$ is the word

$$w^R = a_n \dots a_2a_1, \quad \text{and} \quad E^R = \{w^R|w\in E\}.)$$

Proof. (1): Let E, E_1, and E_2 be i-regular sets.

Then $(E_1\cup E_2)^i=E_1^i\cup E_2^i=E_1\cup E_2$. Let $\mathscr{A}=(A, X, \delta, a_0, F)$ be an involutorial recognizer with $E=T(\mathscr{A})$. Then $\bar{E}=T(\bar{\mathscr{A}})$, where $\bar{\mathscr{A}}=(A, X, \delta, a_0, A\setminus F)$ is an involutorial recognizer. The proof of (2) is trivial.

3.6. Theorem. The product of two i-regular sets and the iteration (star operation) of an i-regular set are not necessarily i-regular.

Proof. Consider $X=\{x\}$ and $E=\{x, x^3, \dots, x^{2n+1}, \dots\}$. Then E is i-regular. But $E^2=\{x^2, x^4, \dots, x^{2n}, \dots\}$ is not i-regular, since $(E^2)^i=\{e, x^2, x^4, \dots, x^{2n}, \dots\}$. This shows nonclosure under product. Now let $\mathscr{A}=(A, X, \delta, a_0, F)$ be the recognizer given by

$$A = \{a_0, a_1, a_2\}, \quad X = \{x, y\}, \quad F = \{a_2\}, \quad \text{and} \quad \delta(a_0, y) = \delta(a_1, x) = a_0,$$

$$\delta(a_0, x) = \delta(a_2, y) = a_1, \quad \text{and} \quad \delta(a_1, y) = \delta(a_2, x) = a_2.$$

$\mathscr{A}$ is an involutorial recognizer, therefore $T(\mathscr{A})$ is an i-regular set. If $w\in T(\mathscr{A})$, then $|w|\geqq 2$, and $x^2\notin T(\mathscr{A})$. This implies $x^2\notin T(\mathscr{A})^*$. But $e\in T(\mathscr{A})^*$ implies $x^2\in (T(\mathscr{A})^*)^i$, therefore $T(\mathscr{A})^*\neq (T(\mathscr{A})^*)^i$.

We will now consider the formation of the involutorial closure E^i of an event E. Let us first investigate those events E for which E^i is regular.

Remember that in contrast to the above mentioned (complete deterministic) recognizer, the transition function δ of an *incomplete (deterministic) recognizer* $\mathscr{A}=(A, X, \delta, a_0, F)$ is a partial function from $A\times X$ to A. We assume that δ is extended to $A\times X^*$ in the usual manner. We will call an incomplete (deterministic) recognizer $\mathscr{A}$ a *trim* recognizer, if each state of $\mathscr{A}$ is accessible and coaccessible [3]. If $E\neq\emptyset$ is regular, then there is a trim recognizer $\mathscr{A}$ with $E=T(\mathscr{A})$. Now we can prove:

3.7. Theorem. For each alphabet X there is a nonregular event $E\subseteqq X^*$ such that E^i is regular.

Proof. Let $X=\{x\}$ be a one-element alphabet. Then $E=\{x^{2n^2}|n\geqq 1\}$ is not regular, but $E^i=\{x^{2n}|n\geqq 0\}$ is regular. Now consider the case $|X|\geqq 2$. $L_{\min}(X^*)$ is regular. Let $\mathscr{A}=(A, X, \delta, a_0, F)$ be a trim recognizer such that $T(\mathscr{A})=L_{\min}(X^*)$. According to the structure of the words of $L_{\min}(X^*)$ and the fact that $\mathscr{A}$ is trim, $a'=\delta(a, x)$ for $a, a'\in A$, $x\in X$, implies $\delta(a', x)=\emptyset$. In particular we have $a\neq\delta(a, x)$ for all $a\in A$, $x\in X$. Now choose $x, y\in X$ with $x\neq y$. Since $x\in L_{\min}(X^*)$, there is a final state $a'=\delta(a_0, x)$.

Define an incomplete recognizer $\mathscr{B}=(B, X, \beta, a_0, F)$, whose state set is infinite, in the following way: Set $B=A\cup\{a_i|i\geqq 1\}\cup\{b_i|i\geqq 1\}$ (disjoint union) and

$$\beta(a, z) = \delta(a, z) \quad \text{for all} \quad a\in A, \quad a \neq a', \quad z\in X,$$

$$\beta(a', z) = \delta(a', z) \quad \text{for all} \quad z\in X, \quad z \neq x, \quad \text{and}$$

$$\beta(a', x) = a_1, \ \beta(a_1, x) = a'.$$

Furthermore set

$$\beta(a_i, y) = b_i, \quad \beta(b_i, x) = a_{i+1}, \quad \text{and}$$
$$\beta(b_i, y) = a_i, \quad \beta(a_{i+1}, x) = b_i \quad \text{for all} \quad i \geqq 1.$$

Then it is easy to see that $L_{\min}(X^*)=T(\mathcal{A})\subseteqq T(\mathcal{B})$. Set $E=T(\mathcal{B})$, then $E^i==X^*$, i.e., E^i is regular. We will now show that E is not regular. Assume the contrary. Then there exists a recognizer $\mathcal{C}$ such that $E=T(\mathcal{C})$. Let n be the number of states of $\mathcal{C}$. If $vw\in T(\mathcal{C})$, $v, w\in X^*$, then there exists some $\theta\in X^*$, $|\theta|\leqq n$, such that $v\theta\in\in T(\mathcal{C})=E$.

Consider a word $x(xy)^p$ with $p>n$. Then $\beta(a_0, x(xy)^p(yx)^p)=a'\in F$, hence $x(xy)^p(yx)^p\in E$. But obviously, for all $\xi\in X^*$ with $|\xi|\leqq n$ we have $x(xy)^p\xi\notin T(\mathcal{B})=E$, which yields a contradiction. Hence E is not regular.

Events $E\neq\emptyset$ over an alphabet X with $|X|\geqq 2$, for which E^i is regular, possess an interesting property w.r.t. $L_{\min}(X^*)$.

Let us first give the following definition.

3.8. Definition. Let E, $F\subseteqq X^*$ be events with $E\subseteqq F$. E is said to be *dense in F* iff there exists a nonnegative integer k such that the following holds:

$\forall v\in F\,\exists w\in X^*$ with $|w|\leqq k$ such that $vw\in E$. (Note that $k=0$ implies $E=F$.)

Now we can prove:

3.9. Theorem. Let X be an alphabet with $|X|\geqq 2$ and let $E\subseteqq X^*$ with $E\neq\emptyset$ such that E^i is regular. Then $L_{\min}(E)$ is dense in $L_{\min}(X^*)$.

Proof. Let $\mathcal{A}=(A, X, \delta, a_0, F)$ with $|A|=n$ be an involutorial recognizer such that $T(\mathcal{A})=E^i$. Since $E^i\neq\emptyset$ we have $F\neq\emptyset$, and furthermore, according to 1.2. Lemma, we can assume that (A, X, δ) is strongly connected. We have to show that there exists a nonnegative integer k such that, for all $v\in L_{\min}(X^*)$ there exists $w\in X^*$, $|w|\leqq k$, with $vw\in L_{\min}(E)$. First assume $v=v'x\in L_{\min}(X^*)$, where $x\in X$. Choose $y\in X$ with $x\neq y$ and set $u=(yx)^n$. Since $|A|=n$, $F\neq\emptyset$, and (A, X, δ) is strongly connected, there exists $u'\in X^*$ with $|u'|\leqq n$ such that $\delta(a_0, vuu')=\delta(a_0, v'x(yx)^nu')\in\in F$. Set $\alpha=v'x(yx)^nu'$, then $\alpha_{I\min}=v'xw$ with $|w|\leqq 3n=k$. Since $\mathcal{A}$ is an involutorial recognizer, we have $\delta(a_0, vw)=\delta(a_0, vuu')\in F$, i.e. $vw\in E^i$. Furthermore $vw\in E^i\cap L_{\min}(X^*)=L_{\min}(E)$. The case $v=e$ is trivial.

3.10. Corollary. Let X be an alphabet with $|X|\geqq 2$ and let $E\subseteqq X^*$ be a nonempty event. If $L_{\min}(E)$ is finite, then E^i is not regular. In particular, if E is finite, E^i is not regular.

The converse of 3.9. Theorem does not hold. Namely, we have:

3.11. Theorem. The involutorial closure E^i of an event E such that $L_{\min}(E)$ is dense in $L_{\min}(X^*)$ need not be regular.

Proof. Let X be an alphabet with $|X|\geqq 2$. Set $E=L_{\min}(X^*)\setminus\{x\}$, where $x\in X$. Obviously, $L_{\min}(E)$ is dense in $L_{\min}(X^*)$ and $\{x\}^i=X^*\setminus E^i$. Assume that the theorem does not hold. Then, E^i becomes regular. Since X^* and E^i are regular, so is $\{x\}^i$. This contradicts 3.10. Corollary. Hence, the theorem has to hold.

3.10. Corollary shows the existence of involutorial events, which are not regular. This situation is impossible for events which are involutorial and commutative.

An event $E \subseteqq X^*$ is called *commutative*, if E is a union of congruence classes of a congruence $\varkappa$, which is defined by:

$$\forall v, w \in X^*: (v, w) \in \varkappa \quad \text{iff} \quad v = y_1 \dots y_n, \quad w = y_{i_1} \dots y_{i_m},$$

where $i_1, \dots, i_m$ is a permutation of $1, \dots, m$ and $y_i \in X$ for $i \in [1: m]$ with $m \geqq 0$.

It can easily be shown that the congruence $\varkappa + \overset{i}{\longleftrightarrow}$ (the sum of the congruences $\varkappa$ and $\overset{i}{\longleftrightarrow}$) is of finite index. Therefore we have:

3.12. Theorem. If $E \subseteqq X^*$ is involutorial and commutative, then E is regular.

Let us now consider those events E for which E^i is context-free. We will prove a theorem characterizing these events, part of which can be viewed as a special case of a theorem due to Sakarovitch [11].

3.13. Theorem. Let $E \subseteqq X^*$ be an event. Then E^i is context-free iff $L_{\min}(E)$ is context-free.

Proof. Let $L_{\min}(E)$ be a context-free language and $G = (V, X, S, P)$ be a context-free grammar in Greibach normal form with $L(G) = L_{\min}(E)$. Each production of G is of the form $A \to a\alpha$ with $a \in X, \alpha \in V^*$. If $e \in L_{\min}(E)$, then there is a production $S \to e$, and S does not occur on the right-hand side of any production.

Construct a context-free grammar $G_1 = (V_1, X, S, P_1)$ with $V_1 = V \cup \{S_a | a \in \in X\} \cup \{S_1\}$ (disjoint union) and furthermore (1) if $e \notin L(G)$ then

$$P_1 = \{A \to S_a \alpha | A \to a\alpha \in P\} \cup \{S_a \to S_1 a S_1 | a \in X\} \cup \bar{P} \quad \text{with}$$

$$\bar{P} = \{S_1 \to e, \quad S_1 \to S_1 S_1\} \cup \{S_1 \to a S_1 a | a \in X\},$$

(2) if $e \in L(G)$ then extend P_1 of (1) with the production $S \to S_1$.

Since $L_{\min}(E) \subseteqq L_{\min}(X^*)$, $L(G_1) = L_{\min}(E)^i = E^i$ can easily be shown with the aid of 2.2. Theorem. Hence E^i is a context-free language.

Conversely, let E^i be a context-free language. Since the intersection of a context-free language with a regular set is context-free, $L_{\min}(E) = E^i \cap L_{\min}(X^*)$ is context-free.

In analogy to regular involutorial closures, we can state:

3.14. Theorem. For each alphabet X there is a non context-free event $E \subseteqq X^*$ such that E^i is context-free.

Proof. Let $x \in X$. It is easy to see that $E = \{x^{n!} | n \geqq 1\}$ is not context-free. Since $L_{\min}(E) = \{e, x\}$ is regular, E^i is context-free.

On the other hand, we have:

3.15. Theorem. The involutorial closure of a context-free event need not be context-free.

Proof. We have to show the existence of a context-free event E such that $L_{\min}(E)$ is not context-free. Consider the context-free event $E_1 = \{(ca)^n cb(ac)^{2n} | n \geqq 1\}$ over the alphabet $X = \{a, b, c\}$, and set $E = E_1^+$.

Then E is context-free and each word $w \in E$ is of the form

$$w = (ca)^{i_1} cb(ac)^{2i_1} (ca)^{i_2} cb(ac)^{2i_2} \dots (ca)^{i_k} cb(ac)^{2i_k}$$

with $i_j \geqq 1$ for $j \in [1: k]$ and $k \geqq 1$.

Assume that $L_{\min}(E)$ is context-free. Then $E_2=L_{\min}(E)\cap(ca)(cb)^+(ac)^+$ has to be context-free. Each word $v\in E_2$ is of the form $v=(ca)(cb)^k(ac)^{2^k}$ with $k\geqq 1$. Consider the homomorphism $h\colon X^*\to a^*$ given by $h(a)=a$ and $h(b)=h(c)=e$.

Then $h(E_2)=\{a^{2^k+1}|k\geqq 1\}$ has to be a context-free language, which is a contradiction. Therefore, $L_{\min}(E)$ is not context-free.

4. The structure of some special recognizers

We will start this section with the investigation of the structure of any trim recognizer accepting $L_{\min}(E)$, where $E\subseteqq X^*$ ($|X|\geqq 2$) is a nonempty involutorial regular event.

Let us first introduce the following notation. If $\mathscr{A}=(A, X, \delta, a_0, F)$ is a recognizer and $a\in A$, then we set

$$I_a=\{x|x\in X \text{ and } \delta(a', x)=a \text{ for an } a'\in A\} \text{ and}$$

$$0_a=\{x|x\in X \text{ with } \delta(a, x)\neq\emptyset\}.$$

Now we can state:

4.1. Theorem. Let $X=\{x_1, \ldots, x_n\}$ be an alphabet with $n\geqq 2$, let $E\subseteqq X^*$ be a nonempty involutorial regular set, and let $\mathscr{A}=(A, X, \delta, a_0, F)$ be a trim recognizer with $T(\mathscr{A})=L_{\min}(E)$. Then we have

(1) $\delta(a_0, x)\neq\emptyset$ for all $x\in X$. Furthermore, if $x, y\in X$ with $x\neq y$, then $\delta(a_0, x)\neq\delta(a_0, y)$ holds,

(2) The set I_{a_0} is empty. In particular $\delta(a_0, x)\neq a_0$ holds for all $x\in X$.

(3) Let $a\in A$ with $a\neq a_0$. Then $|I_a|=1$, i.e. a can be reached by exactly one $x\in X$, and $I_a\cap 0_a=\emptyset$, $I_a\cup 0_a=X$, i.e. a can be leaved by exactly those $y\in X$ with $y\neq x$. Furthermore we have $\delta(a, x)\neq\delta(a, y)$ for all $x, y\in 0_a$ with $x\neq y$.

Proof. (1) Each letter x is an element of $L_{\min}(X^*)$, therefore, according to 3.9. Theorem, there exists $w\in X^*$ such that $xw\in L_{\min}(E)$, which implies $\delta(a_0, x)\neq\emptyset$. Now let $x, y\in X$ with $x\neq y$, and assume $\delta(a_0, x)=\delta(a_0, y)$. Since $xy\in L_{\min}(X^*)$, then, with the same argument, there exists a word $w\in X^*$ such that $xyw\in L_{\min}(E)$, i.e. $\delta(a_0, xyw)\in F$. Since $\delta(a_0, xy)=\delta(a_0, yy)$, we conclude $\delta(a_0, yyw)\in F$, i.e. $yyw\in L_{\min}(E)$, which is a contradiction. (2) Assume that there exist $x\in X$, $a\in A$ such that $\delta(a, x)=a_0$ holds. Since a is accessible, there exists $u\in X^*$ with $\delta(a_0, u)=a$. Hence $\delta(a_0, ux)=a_0$. With the aid of (1) we conclude $\delta(a_0, uxx)=\delta(a_0, x)\neq\emptyset$. Since $\delta(a_0, x)$ is coaccessible, there exists $v\in X^*$ such that $\delta(\delta(a_0, x), v)\in F$, i.e. $\delta(a_0, uxxv)\in F$. This means that $uxxv\in L_{\min}(E)$, which yields a contradiction. (3) Assume that there exist $x, y\in I_a$ with $x\neq y$. Then there are $a', a''\in A$ with $\delta(a', x)=\delta(a'', y)=a$. Since $\mathscr{A}$ is trim, there are $v', v''\in X^*$ with $\delta(a_0, v')=a'$ and $\delta(a_0, v'')=a''$, which implies $\delta(a_0, v'x)=\delta(a_0, v''y)=a$. Since $v''y\in L_{\min}(X^*)$ and $x\neq y$, we have $v''xy\in L_{\min}(X^*)$. There exists $w\in X^*$ with $v''yxw\in L_{\min}(E)$, i.e. $\delta(a_0, v''yxw)=\delta(a_0, v'xxw)\in F$. Therefore $v'xxw\in L_{\min}(E)$, which is a contradiction. Notice that, since $\mathscr{A}$ is trim, there exist $x\in X$, $a'\in A$ with $\delta(a', x)=a$. This, together with the foregoing, shows $|I_a|=1$.

$I_a\cap 0_a=\emptyset$ follows from the fact that $\mathscr{A}$ is trim and $T(\mathscr{A})=L_{\min}(E)$. Now let $I_a=\{x\}$ and $\delta(a', x)=a$. There exists $v\in X^*$ with $\delta(a_0, v)=a'$, and we have

$vx \in L_{\min}(X^*)$. Let $y \in X$ with $x \neq y$. Then $vxy \in L_{\min}(X^*)$ holds, too. There exists $w \in X^*$ such that $vxyw \in L_{\min}(E)$, i.e. $\delta(a_0, vxyw) \in F$, and therefore we have $\delta(a, y) \neq \emptyset$. The rest of the assertion can be proved in a way similar to the corresponding part of (1).

4.2. Example. For each alphabet X we will introduce an automaton, named $\mathscr{L}_X$, which accepts exactly $L_{\min}(X^*)$. To this end, set

$$\mathscr{L}_X = (L_X, X, \lambda_X, l_0, L_X) \quad \text{with} \quad L_X = \{l_0\} \cup \{l_x | x \in X\},$$
$$\lambda_X(l_0, x) = l_x \quad \text{for all} \quad x \in X, \quad \text{and} \quad \lambda_X(l_x, y) = l_y \quad \text{for all} \quad x, y \in X$$
$$\text{with} \quad x \neq y.$$

It is easy to see that $\mathscr{L}_X$ is trim and $T(\mathscr{L}_X) = L_{\min}(X^*)$ holds.

Recall that an incomplete recognizer is called *minimal*, if it is trim and reduced (see [3]). In the following we will construct for a given nonempty involutorial regular event $E \subseteq X^*$ ($|X| \geqq 2$) a minimal recognizer which accepts $L_{\min}(E)$. To this end, we first need two lemmas.

4.3. Lemma. Let $\mathscr{A} = (A, X, \delta, a_0, F)$ with $|X| \geqq 2$ and $F \neq \emptyset$ be a cyclic involutorial recognizer. Then for all $x, y \in X$ with $x \neq y$ and all $a \in A$ there exist $u, v \in X^*$ such that $uxyv \in L_{\min}(T(\mathscr{A}))$ and $\delta(a_0, ux) = a$ holds.

Proof. Since $\mathscr{A}$ is cyclic and involutorial, $\mathscr{A}$ is strongly connected (see 1.2. Lemma). Let $a \in A$, choose $m > |A|$ and set $c = \delta(a, (xy)^m)$. Since $\mathscr{A}$ is involutorial, we have $a = \delta(c, (yx)^m)$. Since $\mathscr{A}$ is strongly connected, there exists $u' \in X^*$ with $|u'| \leqq m$ and $\delta(a_0, u') = c$. Therefore we have $\delta(a_0, u'(yx)^m) = a$. Set $\bar{u} = u'(yx)^m$. Then, from $|u'| \leqq m$, we conclude $\bar{u}_{l_{\min}} = ux$. Since $\mathscr{A}$ is involutorial, we have $\delta(a_0, ux) = a$. In a similar way we can show that there exists $\bar{v}$ with $\bar{v}_{l_{\min}} = yv$ and $\delta(a, yv) \in F$. Then $\delta(a_0, uxyv) \in F$, and since $x \neq y$, we have $uxyv \in L_{\min}(T(\mathscr{A}))$.

The following lemma states a property of accessibility and coaccessibility in $\mathscr{A} \times \mathscr{L}_X$, where $\mathscr{A}$ is a complete minimal involutorial recognizer which satisfies the conditions of the foregoing lemma, and $\mathscr{L}_X$ is the trim recognizer of 4.2. Example.

4.4. Lemma. Let $\mathscr{A} = (A, X, \delta, a_0, F)$ be a complete minimal involutorial recognizer with $|X| \geqq 2$ and $F \neq \emptyset$. Then, in the product automaton $\mathscr{A} \times \mathscr{L}_X$, the following holds:

(1) (a, l_0) is not accessible for all $a \in A$ with $a \neq a_0$,

(2) All the other states of $\mathscr{A} \times \mathscr{L}_X$ are accessible and coaccessible.

Proof. (1) is trivial according to the fact that $\mathscr{L}_X$ is a trim recognizer accepting $L_{\min}(X^*)$ and (2) of 4.1. Theorem. To prove (2), we have to consider the set of states $\{(a_0, l_0)\} \cup \{(a, l_x) | a \in A, x \in X\}$. It is easy to see that (a_0, l_0) is accessible and coaccessible. Consider a state (a, l_x), $a \in A$, $x \in X$. By the foregoing lemma, for all $y \in X$ with $x \neq y$ there exist $u, v \in X^*$ with $uxyv \in L_{\min}(T(\mathscr{A}))$ and $\delta(a_0, ux) = a$. This implies $(\delta \times \lambda_X)((a_0, l_0), ux) = (\delta(a_0, ux), \lambda_X(l_0, ux)) = (a, l_x)$, i.e. (a, l_x) is accessible, and $(\delta \times \lambda_X)((a, l_x), yv) \in F \times L_X$, i.e. (a, l_x) is coaccessible.

4.5. Theorem. Let $E \subseteq X^*$, $|X| \geqq 2$, be a nonempty involutorial regular event and $\mathscr{A} = (A, X, \delta, a_0, F)$ be a complete minimal recognizer with $T(\mathscr{A}) = E$. Then $(\mathscr{A} \times \mathscr{L}_X)^t$ is a minimal recognizer with $T((\mathscr{A} \times \mathscr{L}_X)^t) = L_{\min}(E)$. (Here $(\mathscr{A} \times \mathscr{L}_X)^t$ denotes the trim recognizer associated with $\mathscr{A} \times \mathscr{L}_X$ (see [3]).)

Proof. According to 3.3. Corollary, $\mathscr{A}$ is an involutorial recognizer, and since $E\neq\emptyset$, we have $F\neq\emptyset$. Furthermore, $T((\mathscr{A}\times\mathscr{L}_X)^t)=T(\mathscr{A}\times\mathscr{L}_X)=T(\mathscr{A})\cap T(\mathscr{L}_X)= =E\cap L_{\min}(X^*)=L_{\min}(E)$. From the foregoing lemma we know that $\{(a_0, l_0)\}\cup \cup\{(a, l_x)|a\in A,\ x\in X\}$ is the set of states of $(\mathscr{A}\times\mathscr{L}_X)^t$. If we show that $(\mathscr{A}\times\mathscr{L}_X)^t$ is reduced, then the theorem is proved.

Let us show that (a_0, l_0) is not equivalent to any state (a, l_y), $a\in A$, $a\neq a_0$, and $y\in X$. With the aid of 4.3. Lemma, we can find $v\in X^*$ such that $\delta(a_0, vy)\in F$ and $yv\in L_{\min}(X^*)$. Therefore $(\delta\times\lambda_X)((a_0, l_0), yv)\in F\times L_X$, but $(\delta\times\lambda_X)((a, l_y), yv)$ is not defined. Now choose $x, y\in X$ with $x\neq y$ and consider two states (a, l_x) and (a', l_y) with $a, a'\in A$. Again, with the aid of 4.3. Lemma, we can find a word $v\in X^*$ with $\delta(a, yv)\in F$ and $yv\in L_{\min}(X^*)$: Therefore, $(\delta\times\lambda_X)((a, l_x), yv)\in F\times L_X$, but $(\delta\times\lambda_X)((a', l_y), yv)$ is not defined. It remains to show that (a, l_x) and (a', l_x) are not equivalent for all $x\in X$ and $a, a'\in A$ with $a\neq a'$. To this end choose $y\in X$ with $y\neq x$ and $m>|A|^2$. Set $b=\delta(a, (yx)^m)$ and $b'=\delta(a', (yx)^m)$. Since an involutorial recognizer is obviously a permutation recognizer, we have $b\neq b'$. Since $\mathscr{A}$ is reduced, there exists $u\in X^*$ such that $\delta(b, u)\in F$ and $\delta(b', u)\notin F$ (or vice versa). Here we can assume that $|u|\leqq|A|^2$ holds. Hence we have $\delta(a, (yx)^m u)\in F$ and $\delta(a', (yx)^m u)\notin F$ (or vice versa). Set $w=(yx)^m u$. Since $|u|<m$, we have $w_{l_{\min}}=yv$ for a suitable $v\in X^*$. Consequently, we have $\delta(a, yv)\in F$ and $\delta(a', yv)\notin F$ (or vice versa). Since $y\neq x$, we have $(\delta\times\lambda_X)((a, l_x), yv)\in F\times L_X$ and $(\delta\times\lambda_X)((a', l_x), yv)\notin F\times L_X$ (or vice versa). This ends the proof of the theorem.

The proof of the following corollary now is trivial.

4.6. Corollary. Let $\mathscr{A}=(A, X, \delta, a_0, F)$ be a complete minimal involutorial recognizer with $|X|\geqq 2$ and $F\neq\emptyset$. Let $\mathscr{A}'=(A', X, \delta', a_0', F')$ be a minimal recognizer such that $T(\mathscr{A}')=L_{\min}(T(\mathscr{A}))$ holds. Then we have $|A'|=|A|\,|X|+1$.

5. Decidability results

In this section we will first investigate the decidability of the question "$T(\mathscr{A})^i= =T(\mathscr{B})^i$", where $\mathscr{A}$ and $\mathscr{B}$ are two given recognizers.

To treat this problem we first need, for a given alphabet X and a language $L\subseteq X^*$, the operator λ_L mapping subsets of X^* to subsets of X^* (see e.g. [1]). λ_L is defined as follows: Let $w, w'\in X^*$. Then $w'\in\lambda_L(w)$ iff $w=w_0x_1w_1x_2\ldots w_{r-1}x_rw_r$ with $w_i\in L$ for $i\in[0:r]$, $x_j\in X$ for $j\in[1:r]$, $r\geqq 0$, and $w'=x_1x_2\ldots x_r$.

It is known that, if $R\subseteq X^*$ is a regular event, then $\lambda_L(R)$ is regular for arbitrary languages $L\subseteq X^*$. Taking into consideration of this fact given in [1], p. 60, it can be seen that, if L is a given context-free language and R is a given regular language, one can effectively construct a recognizer accepting $\lambda_L(R)$. If we choose $L=\{e\}^i$, then, according to 2.2. Theorem and 2.5. Corollary, we have $w_{l_{\min}}\in\lambda_L(w)$ for all $w\in X^*$.

Now we can prove:

5.1. Theorem. Let two recognizers $\mathscr{A}$ and $\mathscr{B}$ be given. Then $T(\mathscr{A})^i=T(\mathscr{B})^i$ is decidable.

Proof. $T(\mathscr{A})^i=T(\mathscr{B})^i$ is equivalent to $L_{\min}(T(\mathscr{A}))=L_{\min}(T(\mathscr{B}))$. From $L_{\min}(T(\mathscr{A}))=\lambda_L(T(\mathscr{A}))\cap L_{\min}(X^*)$ with $L=\{e\}^i$ and similarly $L_{\min}(T(\mathscr{B}))=$

$=\lambda_L(T(\mathscr{B}))\cap L_{\min}(X^*)$ and the fact that we can construct recognizers for $\lambda_L(T(\mathscr{A}))$, $\lambda_L(T(\mathscr{B}))$, and $L_{\min}(X^*)$ the theorem follows.

We already mentioned that the involutorial closure of a regular set is a deterministic context-free language. If we specialize the proof of Theorem 3.3 in [2], then we can construct for a given regular set E, a deterministic pushdown acceptor for E^i. Since it is decidable whether the language accepted by a deterministic pushdown automaton is regular (see e.g. [8], p. 246), we conclude:

5.2. Theorem [Book]. Let a recognizer $\mathscr{A}=(A, X, \delta, a_0, F)$ be given. Then it is decidable whether $T(\mathscr{A})^i$ is regular or not.

Using our structure results in section 4, we are able to give an algorithm for this problem, which does not use deterministic pushdown automata, but finite automata only.

5.3. Algorithm.
Input: A recognizer $\mathscr{A}=(A, X, \delta, a_0, F)$.
Output: "YES", if $T(\mathscr{A})^i$ is regular, "NO" otherwise.
Method:

(1) If $T(\mathscr{A})=\emptyset$ or $|X|=1$, go to (5)

(2) Now we have $|X|\geqq 2$ and $F\neq\emptyset$.

Construct a minimal recognizer $\mathscr{A}'=(A', X, \delta', a_0', F')$ with $T(\mathscr{A}')=$ $=L_{\min}(T(\mathscr{A}))=\lambda_L(T(\mathscr{A}))\cap L_{\min}(X^*)$, where $L=\{e\}^i$.

(3) Construct the set $\mathfrak{C}=\Big\{\mathscr{C}\,|\,\mathscr{C}$ is a complete minimal involutorial recognizer with $\dfrac{|A'|-1}{|X|}$ states$\Big\}$.

(4) For all $\mathscr{C}\in\mathfrak{C}$ decide whether $T(\mathscr{A})^i=T(\mathscr{C})^i$ holds. If this is the case for a $\mathscr{C}\in\mathfrak{C}$, go to (5), otherwise go to (6).

(5) Output "YES".

(6) Output "NO".

5.4. Theorem. The output of 5.3. Algorithm is "YES" iff $T(\mathscr{A})^i$ is regular.

Proof. The case $T(\mathscr{A})=\emptyset$ or $|X|=1$ are trivial. Therefore we can assume that $T(\mathscr{A})\neq\emptyset$, which implies $F\neq\emptyset$, and $|X|\geqq 2$ holds. If the output is "YES" then there exists a $\mathscr{C}\in\mathfrak{C}$ with $T(\mathscr{A})^i=T(\mathscr{C})^i=T(\mathscr{C})$, i.e. $T(\mathscr{A})^i$ is regular.

Conversely, assume that $T(\mathscr{A})^i$ is regular. Consider the complete minimal involutorial recognizer $\bar{\mathscr{A}}=(\bar{A}, X, \bar{\delta}, \bar{a}_0, \bar{F})$ with $T(\bar{\mathscr{A}})=T(\mathscr{A})^i$. According to 4.6. Corollary we have $|\bar{A}|=\dfrac{|A'|-1}{|X|}$, i.e. $\bar{\mathscr{A}}\in\mathfrak{C}$, and therefore the output is "YES"..

Abstract

In this paper we will study a special class of automata and events or languages, called involutorial. An automaton with input alphabet X is involutorial iff the double input of one input sign $x\in X$ induces the identity mapping of the state set, an event over an alphabet X is involutorial iff it is saturated w.r.t. to a special (the minimal) involutorial congruence on X^*. This congruence is investigated in section 2. In section 3 we will treat involutorial events and the involutorial closure of arbitrary events. In particular we will study those events, whose involutorial closure is regular or context-free. In section 4 the structure of some special recognizers is determined, and in section 5 we shall give with the aid of these results an algorithm based on finite automata, to decide for a given regular event, whether the involutorial closure is regular or not.

Acknowledgement

The authors would like to thank Prof. Dr. R. Parchmann for valuable discussions and suggestions.

FACULTY OF SCIENCE
KYOTO SANGYO UNIVERSITY
603 KYOTO

INSTITUT FÜR INFORMATIK
UNIVERSITÄT HANNOVER
WELFENGARTEN 1
D-3000 HANNOVER 1

References

[1] BERSTEL, J., Transductions and Context-Free Languages, Teubner, Stuttgart, 1979.
[2] BOOK, R. V., Confluent and other types of Thue systems, J. ACM, v. 29, 1982, pp. 171—182.
[3] EILENBERG, S., Automata, Languages and Machines, v. A, Academic Press, 1974.
[4] FLECK, A. C., Isomorphism groups of automata, J. ACM, v. 12, 1965, pp. 566—569.
[5] GÉCSEG, F., On subdirect representations of finite commutative unoids, Acta Sci. Math., v. 36, 1974, pp. 33—38.
[6] GÉCSEG, F. and PEÁK, I., Algebraic Theory of Automata, Akadémiai Kiadó, Budapest, 1972.
[7] HARRISON, M. A., Introduction to Formal Language Theory, Addison-Wesley, Reading, Mass. 1978.
[8] HOPCROFT, J. E. and ULLMANN, J. D., Introduction to Automata Theory, Languages, and Computation, Addison-Wesley, Reading, Mass., 1979.
[9] RABIN, M. O. and SCOTT, D., Finite automata and their decision problems, IBM J. Res. Develop., v. 3, 1959, pp. 114—125.
[10] SAKAROVITCH, J., Un théorème de transversale rationelle pour les automates à piles déterministes, Proc. of the 4th GI conference on Theoretical Computer Sci., K. Weihrauch, ed., (Lect. Notes in Computer Sci., 67), Springer, 1979, pp. 276—285.
[11] SAKAROVITCH, J., Description des monoides de type fini, Elektronische Informationsverarbeitung und Kybernetik, v. 17, 1981, pp. 417—434.
[12] ZASSENHAUS, H. J., The Theory of Groups, Chelsea, New York, 1958.

A solution of the early bird problem in an n-dimensional cellular space

By T. Legendi and E. Katona

The early bird problem has been defined in [1] and has been solved in [1], [2], [3], [5]. In this paper the problem is generalized for the n-dimensional cellular space, and a real time solution is given using $0(mn)$ steps in an n-dimensional cellular field of edge length m. The solution will be shown in detail in the two-dimensional case. The basic idea of the solution is to reduce the n-dimensional early bird problem to the $(n-1)$-dimensional one, using special signals and applying the one-dimensional early bird algorithm from [3].

1. Introduction

Let us consider a cellular space in which any cell in quiescent state may be excited from the outside world. The excitations result in special "bird" states instead of the quiescent states. The task is to give a transition function ensuring after a certain time the first excitation(s) may be distinguished from the later ones. This is the early bird problem defined originally by Rosenstiehl et al. [1] for an elementary cyclic graph where to each of the m vertices an automaton (cell) is assigned. In [1] a $2m$-step solution is given on condition that at each time-step maximum one excitation occurs.

Vollmar [2] defined and solved the problem for a one-dimensional cellular space allowing more than one excitation at each time-step. The solution is based on the "age of waves" concept and requires a high number of cell-states.

In [3] a simplified solution has been given for the one-dimensional case using only 5 cell-states. Kleine Büning [4] proved that the early bird problem is unsolvable with 4 states in a one-dimensional cellular space, that is, the solution in [3] is optimal considering the number of states.

In [5] the early bird problem is solved for a two-dimensional cellular space in nonlinear time (for an $m \cdot m$ space $0(m^2)$ steps are needed.

2. Exact definition of the n-dimensional early bird problem

Let (I^n, S, N, f) be an n-dimensional cellular space, where

I is the set of integers and to each point of I^n a *cell* is assigned;

S is the finite set of *cell-states* containing 3 special states, the passive state #, the quiescent state q and the "bird" state β;

N is the *neighbourhood index*, in this paper the von Neumann neighbourhood index is assumed: $N=((0,0),(0,1),(0,-1),(1,0),(-1,0))$;

f: $S^5 \to S$ is the *local transition function* satisfying $f(q, \ldots, q)=q$ and $f(\#, a, b, c, d)=\#$ for any $a, b, c, d \in S$.

In the cellular space an n-dimensional cube K of edge length m is assigned: $K=\{(i_1, \ldots, i_n) | 0 \leqq i_j \leqq m-1, j=1, \ldots, n\}$. In the initial configuration the cells in K are in state q (active space), all other cells are in state # (passive space).*

The cellular space works, as usual, in discrete time-steps $t=0, 1, 2, \ldots$ using the local transition function f, but between two steps (that is to say, at time $t+1/2$) any quiescent cell (in state q) may change spontaneously into the bird state β (excitation). The problem is to define a transition function f ensuring that after a certain time the bird(s) arisen at first is (are) in a distinguished state, and all other cells in other states.

To simplify the solution, the excitation of the border cells (which have a neighbour in state #) will be prohibited.

For an easier explanation, first the solution will be presented in the two-dimensional case (point 3, 4). The generalization for n-dimensions will be discussed in point 5.

3. The sketch of the solution in the two-dimensional case

The basic idea of the algorithm is that any bird sends out a special signal in horizontal direction, which arrives at the leftmost column z steps after its origin where z is independent of the place of the bird (Fig. 1). That is, any excitation in the two-dimensional field at time t, may generate a "secondary excitation" in the leftmost column at time $t+z$. In this way the two-dimensional early bird problem can be "projected" into a one-dimensional one.

A bird will be called a *local early bird* if there is no earlier bird in its row, and it will be called a *global early bird* if there is no earlier bird in the cellular space. Using a one-dimensional early bird algorithm in each row, the local early birds can be marked. Using a one-dimensional early bird algorithm in the leftmost column, we can decide that which rows contain the global early birds. It is clear that the vertical and the horizontal early bird algorithms together may select the global early birds.

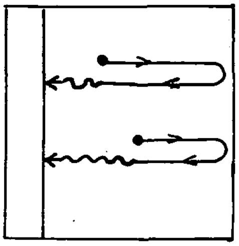


Fig. 1

The main problem in the above solution is to ensure a constant z delay for any signal sent by a bird. The solution of this problem is explained in a time-space diagram (Fig. 2).

[*] This partition of the cellular space corresponds to the "retina" conception of [7].

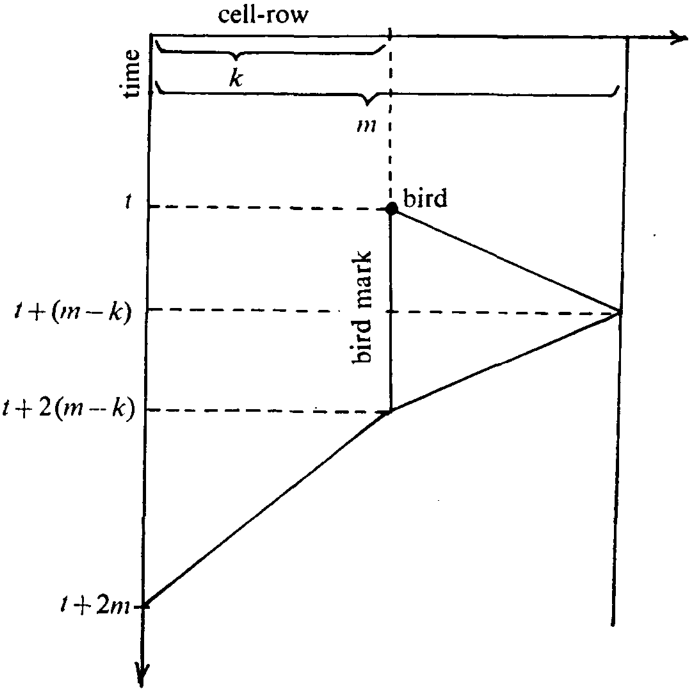


Fig. 2
The time-space diagram of a row of the cellular space

Let us consider a bird arisen at time t in the k-th cell of a cell-row. This bird sends a full-speed signal to the right (the signal steps right in each step) and in the k-th cell there remains a special sign, a so-called "bird mark". The full-speed signal is reflected at the rightmost cell at time $t+(m-k)$ and moves left until it reaches the bird mark (at time $t+2(m-k)$). Here it cancels the bird mark (the bird itself survives), and the full-speed signal alters into a half-speed signal moving to the left. This signal reaches the leftmost cell at time $t+2(m-k)+2k=t+2m$, thus the constant delay of the signal is ensured.

Considering more than one bird, there arises the question whether each full-speed signal will cancel its own bird mark. This question can be answered with "yes" because the following property holds.

Proposition. Let b_1 and b_2 two bird marks in the k_1-th and k_2-th cells of the cell-row, respectively. If $k_1<k_2$ then the full-speed signal s_2 (of b_2) goes before the full-speed signal s_1 (of b_1).

Proof. The definition of the early bird problem contains the restriction that only quiescent cells may alter into the bird state. Considering the 5-state early bird algorithm of [3] it is clear that the cells between the bird mark b_1 and its signal s_1 cannot be in quiescent state, therefore the excitation in this range is prohibited. This fact implies the above proposition. □

It results from the proposition that the first reflected signal has been sent by the rightmost bird, the second reflected signal by the next rightmost bird, etc., thus the cancellation of the bird marks always will be correct.

Summarizing, *the sketch of the two-dimensional algorithm* is as follows:

(i) Any new bird in the active cellular field sends out a signal which arrives at the leftmost column with a constant delay. At the same time, a one-dimensional early bird algorithm is executed in each row which cancels all "local late birds".

(ii) The signals arriving at the leftmost column appear as "secondary birds" and a vertical one-dimensional early bird algorithm is applied among them. If a secondary bird proved to be later then it is killed and a "cancel" signal starts moving to the right in that row which kills all the birds there.

After 6m steps (see the time estimation in point 5) only the global early birds exist in the cellular space.

4. Detailed description of the solution in the two-dimensional case

The state set $S'=S-\{\#\}$ is composed of two components: $S'=S_1\times S_2$. Component S_1 serves for the 5-state early bird algorithm of [3], that is, $S_1=\{Q, B, R, L, N\}$ where the states are named "quiescent", "bird", "right wave", "left wave", "neutral", respectively.

The component S_2 ensures the movement of signals according to Fig. 2. It consists of 5-bit words: $S_2=\{(s_1, \ldots, s_5)|s_i\in \{0, 1\}, i=1, \ldots, 5\}$, where

$s_1=1$ means "bird mark",
$s_2=1$ means "full-speed signal moving to the right",
$s_3=1$ means "full-speed signal moving to the left",
$s_4=1$ means "half-speed signal in phase 1",
$s_5=1$ means "half-speed signal in phase 2".

The quiescent state q and the bird state β (see point 2) are defined as $q=(Q,00000)$, $\beta=(B,11000)$.

The active cellular space is divided into three areas (Fig. 3). The leftmost column is called as area A_1, the inner cells as area A_2 and the rightmost column as area A_3. The transition function f involves three subfunctions f_1, f_2, f_3 according to A_1, A_2, A_3. (The cells in different areas may be distinguished by their left and right neighbours.) In the sequel the areas A_1, A_2, A_3 will be discussed separately.

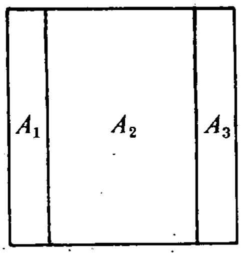


Fig. 3

Area A_2 (inner cells).

The transition function f_2 is composed of two functions. For the component S_1 the 5-state early bird function of [3] is used defined below without any explanation. (The terms have "left-own-right→new state" structure. An expression (B, R) means "state B or state R", points mean arbitrary states. In the undefined cases the new state must be equal to the old own state.)

$$\begin{array}{cccl}
(B, R) & Q & (Q, N) & \rightarrow R \\
(Q, N) & Q & (B, L) & \rightarrow L \\
(B, R) & Q & (B, L) & \rightarrow N \\
. & R & L & \rightarrow N \\
R & L & . & \rightarrow N \\
R & (B, N) & (\text{not } L) & \rightarrow R \\
. & R & (B, N) & \rightarrow N \\
(\text{not } R) & (B, N) & L & \rightarrow L \\
(B, N) & L & . & \rightarrow N \\
R & (B, N) & L & \rightarrow N
\end{array}$$

For the component S_2 the transition function can be defined by two terms (an expression in parentheses is a 5-bit word, the points mean arbitrary bits):

Term 1. Shift of signals ($a \cdot c \neq 1$ supposed):
$(.b...)(a..e.)(..c.d) \rightarrow (abcde)$.

Term 2. Cancellation of the bird mark:
$(.b...)(1..e.)(..1..) \rightarrow (0b01e)$.

Area A_3 (rightmost column)

This area serves for reflecting the full-speed signals. The state of the component S_1 is constantly N in each cell (according to [3]), and the transition of the component S_2 is defined by

Term 3: $(.b...)(00.00)(\#) \rightarrow (00b00)$.

Area A_1 (leftmost column)

This area requires only 5 states according to the vertical one-dimensional early bird algorithm. These 5 states are chosen from the state-set $S' = S_1 \times S_2$ as follows:

"quiescent" $= (N{,}00000)$,
"bird" $= (N{,}10000)$,
"right wave" $= (R{,}00000)$,
"left wave" $= (R{,}10000)$,
"neutral" $= (R{,}00100)$.

This choice of states solves two problems:

(i) If in the vertical early bird algorithm a cell is in state "right wave", "left wave" or "neutral", then in the row of this cell all the birds should be cancelled, because they cannot be early birds. Using the above choice of states, a cell in state "right wave", "left wave" or "neutral" shows a state $R(\in S_1)$ for the horizontal early bird algorithm. As a consequence, R states will be generated in the cell-row moving right and cancelling all the birds there.

(ii) It is easy to see that in all cases except (i), a cell in A_1 shows an indifferent state for its right neighbour, because the state $N(\in S_1)$ ensures a correct horizontal early bird algorithm, and the states 10000, 00100 $(\in S_2)$ do not disturb the movement of signals in the cell-row.

Using the above choice of states, the cells in A_1 work with a vertical one-dimensional early bird function, but instead of the spontaneous excitation a "secondary bird generation term" is introduced:

Term 4:
$$\begin{array}{l} \quad (N,00000) \\ \#(N,00000)(.,\dots .1)\rightarrow(N,10000). \\ \quad (N,00000) \end{array}$$

The solution described above requires $\|S'\|=5\cdot 32=160$ states. Note that the state-set can be reduced to $5\cdot 14=70$ states, but this reduction results in a more complicated transition function therefore its discussion is omitted.

5. The n-dimensional case

Let us consider the n-dimensional cube K defined in point 2. The inner cells of K form an n-dimensional cube K_n of edge length $m-2$: $K_n=\{(i_1, \ldots, i_n)|1\leqq i_j\leqq m-2, j=1, \ldots, n\}$. Using the signals of figure 2, any excitation in K_n can be projected into an $(n-1)$-dimensional cube $K_{n-1}=\{(i_1, \ldots, i_{n-1}, 0)|1\leqq i_j\leqq m-2, \ j=1, \ldots, n-1\}$, in K_{n-1} secondary excitations are induced. At the same time, in each cell-row of K_n a one-dimensional early bird algorithm is executed to select the local early birds.

For the secondary excitations in K_{n-1} a similar process is used reducing the task to an $(n-2)$-dimensional cube $K_{n-2}=\{(i_1, \ldots, i_{n-2}, 0, 0)|1\leqq i_j\leqq m-2, \ j=1, \ldots, \ldots, n-2\}$; similar reduction can be made for K_{n-3}, etc. In $K_1=\{(i_1, 0, \ldots, 0)|1\leqq \leqq i_1\leqq m-2\}$ only a one-dimensional early bird algorithm is needed.

For any $i=1, \ldots, n-1$, if a bird in K_i is cancelled then all birds in the corresponding row of K_{i+1} will be cancelled, similarly to the two-dimensional solution. It is not difficult to prove that after a certain time only the global early birds exist in K_n.

Time estimation: Let t be the time-point when the global early birds arise. (Before t the cellular space is in quiescent state.) At time $t+2m$ the signals of the early birds arrive at K_{n-1}, at time $t+4m$ the signals of the secondary birds arrive at K_{n-2}, etc. Thus the earliest excitations appear in K_1 at time $t+(n-1)2m$. The one-dimensional early bird algorithm of [3] requires $3m$ steps, therefore at time $t+(n-1)2m+3m$ all one-dimensional early bird algorithms in $K_1, \ldots, K_n$ are terminated. The cancellation process from K_1 to K_n needs maximum $(n-1)m$ steps, thus at time $t+(n-1)2m+ +3m+(n-1)m=t+3mn$ all late birds are cancelled in K_n. That is, *the n-dimensional early bird algorithm requires 3mn steps.*

Remark: For the cube K a transition function f is used which is composed of different subfunctions. The subfunctions of the cubes $K_2, \ldots, K_n$ are similar to the function f_2 in point 4, but at the boundary of K_i and K_{i-1} a coding problem arises (see the choosing problem of states in the area A_1). This problem can be solved by increasing the number of states. As a consequence, the state-set S may grow if the dimension degree n is increased.

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

References

[1] ROSENSTIEHL, P., J. R. FIKSEL and A. HOLLIGER, Intelligent graphs: Networks of finite automata capable of solving graph problems, in: Read, R. C. (ed.), *Graph Theory and Computing*, Academic Press, New York, 1972, pp. 219—265.

[2] VOLLMAR, R., On two modified problems of synchronization in cellular automata, *Acta Cybernet.*, v. 3, 1978, pp. 293—300.

[3] LEGENDI, T. and E. KATONA, A 5-state solution of the early bird problem in a one-dimensional cellular space, *Acta Cybernet.*, v. 5, 1981, pp. 173—179.

[4] KLEINE BÜNING, H., The early bird problem is unsolvable in a one-dimensional cellular space with 4 states, *Acta Cybernet.*, v. 6, 1983, pp. 23—31.

[5] PRIESE, L., Über das Early Bird Problem in planaren Zellularräumen, *Informatik-Skripten der Technischen Universität Braunschweig*, Nr. 2, Braunschweig, 1982, pp. 105—110.

[6] LEGENDI, T. and E. KATONA, A solution of the early bird problem in an n-dimensional cellular space (preliminary version), *Informatik-Skripten der Technischen Universität Braunschweig*, Nr. 2, Braunschweig, 1982, pp. 60—64.

[7] VOLLMAR, R., *Algorithmen in Zellularautomaten*, B. G. Teubner, Stuttgart, 1979.

(Received Febr. 14, 1984)

Language extension in the HLP/SZ system

By E. Simon

1. Introduction

It is the intent of this paper to describe a practicable modification of the HLP lexical metalanguage for specification of the language extension. Programmers should never be satisfied with languages which permit them to program everything. Therefore, there is a need for languages which help the programmer to find the most natural representation for the data structures he uses and the operations he wants to perform on them. This is clearly illustrated by the current trends in the evaluation of programming languages [4].

In order to achieve flexibility and power of expressions in programming languages, we must pay the price of greater complexity. In the 70's there was a tendency to retrench towards simpler languages such as PASCAL, even at the price of restricted flexibility and power of expressions. An alternative solution was the development of *extensible languages* too. The classification of language extension can be made on the basis of the stage of the translation process during which the definition of an extension is processed and the augment text is converted into a text in the base language.

In compiler-writing systems based on attribute grammars, two natural means are given to introduce extensions. In the first case the augment text is processed *during* the *lexical analysis,* and therefore the extension mechanism is a part of the generated lexical analyzer. Information which controls the recognition of the augment texts and which prescribes the generation of the definition texts can be obtained from the lexical metalanguage description. In this case the extension mechanism is generally similar to macrogenerators. Those lexical analyzers which have an extension mechanism must contain special stack automata to store parameters computed during isolation of a token. Extension mechanisms are generally implemented by recursive procedures. Adapted from [6], the extension executed during the lexical analysis is called the type-A extension. From the user's point of view, the type-A extension can be useful for a large class of problems, but there are some other classes where delay of the extension is needed until the translation process. For example, the recursive procedure call in BASIC language can not be introduced by a type-A extension, because the stack manipulations must be implemented at the target code level. It should be noted that the other type of language extension is based on *attributed tree transfor-*

mations. These are executed during syntax/semantic analysis of the token stream generated by the lexical analyzer. The attributed tree transformations are controlled by those attributes which are computed before transformations. In practice the application of both techniques is suitable.

In the case of the type-A extension there are two different methods to give definitions of the augments. In our HLP/SZ system the recursive definitions are given by generator expressions in lexical metalanguage, while the augments are contained by source programs. Hence, in all cases a new lexical analyzer must be generated on the basis of the new lexical description. If the generated lexical analyzer contains the total extension mechanism, including the parser generator, definition texts are given in the source programs. An extension executed by attributed tree transformations is now under development.

In the next part of this paper the modified lexical metalanguage is presented through examples. It is followed by a short description of our implementation based on SIMULA 67 language. In the following, a knowledge of the original HLP system cf. [1], [2]) is assumed.

2. Extension description in lexical metalanguage

A lexical description of a prgamming language defines the way the programs are written in that language. The streams of characters are divided into syntactically meaningful entities called *tokens.* Token classes are defined by regular expressions which are built up from character sets, terminals and tokens. In our system, five predefinite character sets are introduced with conventional meaning. These character sets are: LETTER, DIGIT, ANY, ENDOFLINE and SPACE. These names can be overdefined in the set declaration list. An essential difference between the original and the HLP/SZ system is that the *transformations* are related to the character set names and token class names only. The transformations are given after the token class description in the following way:

```
TRANSFORMATIONS ARE
identifier⇒; or                                         (1)
identifier_1⇒terminal_symbol;
END OF TRANSFORMATIONS.
```

In the first case, character sets assigned to the "identifier" are deleted during isolation of a token. In the second case, characters are changed by the terminal_symbol. If "identifier_1" is defined as the character set name, then the first character of the terminal_symbol will be inserted after deletion of the last input character. Applications of transformations can be found in the examples of this section.

In order to extend a language it is enough to create one or more generator expressions without any knowledge of the original parts of the lexical description. For introducing the generator expression, let us take the terminology of macrogenerators into consideration.

An *action block* consists of a collection of screen clauses in the original lexical metalanguage. Each screen clause denotes an identifier which is declared as a token class name in the lexical description. Generator expressions are assigned to token class names in action blocks. Character strings belonging to the token classes are treated as augment texts or *macro calls. Macro definitions* are given by generator expressions. Let us assume that, in the definition of a token class assigned to a generator expression, token class names occur. Character strings isolated to these token class names are called *parameters.* References to these parameters in the generator expressions are given by the following syntax rules.

```
formal_parameter = ≠↓≠ unsigned_integer;                    (2)
formal_parameter = ≠↓≠generator_expression;
```

In the first case, parameters are *positionated,* that is the "unsigned_integer" designates the serial number of a parameter. Token class names occuring on the left-hand side of token class definitions are regarded as 0th parameters. Numbering of parameters on the right-hand side is defined from left to right. The serial number of the leftmost token class name, as parameter, is equal to 1. In the second case, the generator expression generates a token class name, which is called *keyword parameter.* We suppose that the generated token class name can be found on the right-hand side of the token definition assigned to the generator expression in which the original formal parameter reference occurred. It should be noted that the token class definition above does not contain the same token class names twice.

In the original lexical metalanguage, the control can be transferred to an other block using GOTO. After a token is processed according to a screen clause, the GOTO part appearing on its left-hand side indicates the block where the next token will be recognized. The default rule states that the block is not changed. In the case of an extension description, the GOTO part can appear after the token class name assigned to a generator expression. The GOTO part occuring behind the generator expression indicates the block where the generated definition text will be recognized.

Generator expressions are built up from *terminal symbols, parameter references* and *functions.* The functions typed as string are the following: BLANK, SUC, PRED and IT_IND. The first function has no argument, while SUC and PRED have one string argument which must be converted into an integer. The function SUC is assumed to be an incrementing function, and PRED to be a decrementing one. The meaning of the function IT_IND can be found in the following part of this section.

In order to build up a generator expression from terminals parameters and functions, string operations are needed. Similarly to regular expressions, the descriptive power of generator expressions is based on the operations *iteration* (∗), *concatenation* (juxtaposition) and *McCarthy expression.* The operator ∗ has the highest precedence and the McCarthy expression the lowest. The use of subexpressions enclosed in parentheses is permitted.

For example, we could have the description to be found in Figure 1. The name of this description is FACTORIAL and no character set definitions are needed. The strings belonging to the token class FACT begin with terminal *FACT.* The left parenthesis is followed by 1—2 digits and the right parenthesis. After the right parenthesis some blank characters can follows, which are deleted during recognition of the token (augment text) called FACT. During evaluation, the generator expression assigned to the token class name FACT, ↓1 contains the substring passed to INT_PARAM as integer parameters of the factorial call. Parameter reference ↓0 assigns the total augment text without last spaces. It can be seen that the extension is *recursive.*

```
        LANGUAGE EXTENSION AT LEXICAL LEVEL
        THE AUGMENT TEXT IS: FACT( INT_PARAM )

LEXICAL DESCRIPTION FACTORIAL
    TOKEN CLASSES
    NUMBER          =DIGIT+ [4];
    MULT            =≠*≠;
    FACT            =≠FACT≠ ≠(≠ INT_PARAM ≠)≠ SPACE*;
    INT_PARAM       =DIGIT+ [2] ;
    END OF TOKEN CLASSES
    TRANSFORMATIONS ARE
    SPACE           ⇒ ;
    END OF TRANSFORMATIONS
BLOCK:BEGIN
    FACT [BLOCK]    →(↓1 EQ ≠1≠ --→ ≠1≠ /
                      ↓0 EQ  ↓0  --→  ↓1  ≠*≠ ≠FACT(≠ PRED(↓1)
                                                        ≠)≠)
                    [BLOCK_1] ;
    END OF BLOCK
BLOCK_1:
    BEGIN
    NUMBER          ⇒NUMBER ;
    MULT [BLOCK]    ⇒MULT_OP ;
    END OF BLOCK_1
END OF LEXICAL DESCRIPTION FACTORIAL.

    FINIS
```

Figure 1

A part of the generated text will be recognized by the automata assigned to the token class FACT. During the processing of an augment text (macro call), a sequence of tokens called NUMBER and MULT_OP are generated in action block BLOCK_1. After the processing, the next active block will be the BLOCK by GOTO transfer assigned to token class name FACT.

In order to create macro calls which have other calls in their own parameter list, the following agreement is reached. If none of the token classes currently under consideration has a right-hand side from which a token class name TO can be derived, the string generated by TO will be normally processed; otherwise, the generated string will be strored *in the stack*.

As an example, let us consider the description of HANOI TOWERS in Figure 2. In this case the string generated by PARAM must be stored in the stack as a parameter of the augment text HANOI. Naturally, theere is a posibility to continue the example found in Figure 2 by creating an attribute grammar to synthesize the original HANOI call on the basis of the token stream generated by the lexical analyzer HANOI_TOWERS.

```
LEXICAL DESCRIPTION HANOI_TOWERS
CHARACTER SETS
    ABC          = ≠ABC≠ ;
END OF CHARACTER SETS
TOKEN CLASSES
    IDENTIFIER = LETTER (LETTER/DIGIT)* ;
    INTEGER     = DIGIT+ [2] ;
    SPACES      = SPACE+ (/ENDOFLINE) ;
    DOT         = ≠.≠ ;
    NUMBER      = DIGIT+ ;
    FROM        = ABC ;
    TO          = ABC ;
    MOVE        = ≠XMOVE≠ ≠(≠ NUMBER ≠,≠ FROM ≠,≠ TO ≠)≠;
    SER_NUMB    = DIGIT+ ;
    PARAM2      = ABC ;
    PARAM1      = ABC ;
    PARAM       = ≠PARAM≠ ≠(≠ PARAM1 ≠,≠ PARAM2 ≠)≠ ;
    ARGUMENT = (ABC/PARAM) ;
    HANOI       = ≠HANOI≠ ≠(≠ SER_NUMB (≠,≠ ARGUMENT)*
                                                      ≠)≠ ;
END OF TOKEN CLASSES
SCREEN_A:
BEGIN
    IDENTIFIER ⇒ KEYSTRINGS ;
    INTEGER     ⇒ INTEGER ;
    SPACES      ⇒ ;
    DOT         = DOT ;
    MOVE        → ≠MOVE≠ ≠ ≠ ≠THE≠ ↓1≠TH≠ ≠ ≠ ≠DISC≠ ≠ ≠
                  ≠FROM≠ ≠ ≠ ≠THE≠ ≠ ≠ ↓2 ≠.≠ ≠ ≠
                  ≠PLACE≠ ≠ ≠ ≠TO≠ ≠ ≠ ≠THE≠ ≠ ≠ ↓3 ≠.≠
                  ≠ ≠ ≠PLACE≠ ;
    HANOI       → (↓1 EQ ≠1≠ --→ ≠XMOVE≠ ≠(≠ ↓1 ≠,≠ ↓2 ≠,≠ ↓3
                  ≠)≠ /
                   ↓0 EQ ↓0 --→
                   ≠HANOI≠ ≠(≠ PRED(↓1) ≠,≠ ↓2 ≠,≠
                           ≠PARAM≠ ≠(≠ ↓2 ≠,≠ ↓3 ≠)≠ ≠)≠
                   ≠XMOVE≠ ≠(≠ ↓1 ≠,≠ ↓2 ≠,≠ ↓3 ≠)≠
                   ≠HANOI≠ ≠(≠ PRED(↓1) ≠,≠
                           ≠PARAM≠ ≠(≠ ↓2 ≠,≠ ↓3 ≠)≠ ≠,≠ ↓3
                                                      ≠)≠) ;
    PARAM       → (↓1 EQ ≠A≠ --→
                  (↓2 EQ ≠B≠ --→ ≠C≠ / ↓0 EQ ↓0 --→ ≠B≠) /
                   ↓1 EQ ≠B≠ --→
                  (↓2 EQ ≠A≠ --→ ≠C≠ / ↓0 EQ ↓0 --→ ≠A≠) /
                   ↓0 EQ ↓0 --→
                  (↓2 EQ ≠A≠ --→ ≠B≠ / ↓0 EQ ↓0 --→ ≠A≠)) ;
END OF SCREEN_A
END OF LEXICAL DESCRIPTION HANOI_TOWERS.

    FINIS
```

Figure 2

```
HLP/SZEGED (0.2)                                  * *SOURCE TEXT* *
------------------------------------------------------------------------
 1
 2           EXTENSION OF AN ALGOLW SUBSET
 3             BY CASE AND FOR STATEMENTS
 4               5 OCTOBER 1983. SZEGED
 5
------------------------------------------------------------------------
 6
 7  LEXICAL DESCRIPTION EXTENDED_ALGOLW_SUBSET
 8
 9  CHARACTER SETS
10* *   PERCENT        = ≠%≠ ;
11      UNDERSCORE     = ≠_≠ ;
12      STMT_CHAR      = ANY − ≠;≠ / ENDOFLINE ;
13      END OF CHARACTER SETS
14
15  TOKEN CLASSES
16      IDENTIFIER     = LETTER (LETTER/DIGIT/UNDERSCORE)*
                         [16] ;
17      NUMBER         = DIGIT+ [8] ;
18      COMMENT        = PERCENT ANY* ENDOFLINE ;
19      SPECIALS       = DEFAULT TOKENS ;
20* *   SPACES         = SPACE+ ;
21      LINE_SKIP      = SPACE* ENDOFLINE ;
22      CASE           = ≠XCASE≠ ≠ ≠INT_PAR≠ ≠ ≠OF≠ ≠ ≠
23                       (STATEMENT ≠;≠)* (SPACE/ENDOFLINE)*
                         ≠END≠ ;
24      FOR            = ≠XFOR≠ ≠ ≠ CYCLE_PAR ≠ ≠ ≠:=≠ ≠ ≠
                         FIRST_VALUE ≠ ≠ ≠STEP≠ ≠ ≠

25                       STEP_VALUE ≠ ≠ ≠UNTIL≠ ≠ ≠
26                       LAST_VALUE ≠ ≠ ≠DO≠ ≠ ≠
                         STATEMENT_1 ;
27      INT_PAR        = DIGIT+ [2] ;
28      STATEMENT      = STMT_CHAR* ;
29      CYCLE_PAR      = LETTER(LETTER/DIGIT/UNDERSCORE)* [16] ;
30* *   FIRST_VALUE    = DIGIT+ [8] ;
31      STEP_VALUE     = DIGIT+ [8] ;
32      LAST_VALUE     = DIGIT+ [8] ;
33      STATEMENT_1    = STMT_CHAR* ;
34      STATEMENT_1    = ≠BEGIN≠ (ANY/ENDOFLINE)* ≠END≠ ;
35  END OF TOKEN CLASSES
36
37      TRANSFORMATIONS ARE
38      UNDERSCORE  ⇒ ;
39      END OF TRANSFORMATIONS
40* *
41  SCAN: BEGIN
```

```
      IDENTIFIER     => IDENTIFIER/KEYSTRINGS;
      NUMBER         => NUMBER ;
      COMMENT        => ;
      SPECIALS       => KEYSTRINGS ;
      SPACES         => ;
      LINE_SKIP      => ;
      CASE           → (≠IF≠ ↓1 ≠=≠ ITIND ≠THEN≠ ↓SUC(ITIND)
                       ≠ELSE≠)* [1] ≠;≠ ;
**    FOR            → ≠BEGIN≠ ↓1 ≠:=≠ ↓2 ≠;≠ ≠LABEL:≠
                       ≠IF≠ ↓1 ≠LE≠ ↓4 ≠THEN≠ ≠BEGIN≠ ↓5≠;≠
                                ↓1 ≠:=≠ ↓1 ≠+≠ ↓3 ≠;≠
                                 ≠GOTO≠ ≠LABEL;≠ ≠END≠ ≠;≠
                       ≠END≠ ≠;≠ [SCAN] ;
   END OF SCAN

   END OF LEXICAL DESCRIPTION EXTENDED_ALGOLW_SUBSET.


-----------------------------------------------------------------
**    FINIS
```

Figure 3

In the following we define the format in which the number of iterations are given to an elementary subexpression of a generator expression. The number of an iteration is equal to 1 if the iteration specification is omitted. If it occurs, then it must be written in the following format:

iteration_number = ≠*≠ ≠[≠ generator_expression ≠]≠ ; or
iteration_number = ≠*≠ ≠[≠ unsigned_integer ≠]≠ ; (3),

assuming that the string generated by the "generator_expression" can be converted into an integer. It should be noted on the basis of (3) that *[2] and *[≠2≠] are (in the same way) syntactically correct items with different meanings. In the first case the value of the "generator_expression" designates the number of iterations to be executed. To illustrate the meaning of "iteration_number" defined secondly, let us consider a token class which is defined by iterations. Let us number the occurrences of iterations from left to right. In this case *[i] has the following meaning. The index i denotes a * symbol on the right-hand side of the token class definition, assigned to the generator expression in which *[i] occurs. In this way the item *[i] means the number of the i-th iterations performed by the automata during isolation of a token. In order to use the iteration number in the form *[i], practical additional features are required. Firstly, during execution of an iteration the elementary string expressions must be evaluated in each step. Secondary, we must introduce a new elementary string expression called IT_IND, which produces the value of the iteration index.

We are now ready to give two extensions of an ALGOLW subset in HLP/SZ lexical metalanguage. The original lexical description [3] is increased by some token class definitions to introduce the CASE and FOR statements into the base language. Additional token class definitions, such as STATEMENT, INT_PAR etc., are needed to describe the parameters of the augments. It seems to me that the definition of a statement list by iteration, which can be seen in the description of the CASE statement (cf. Figure 3), is very useful for giving the extension assigned to the CASE. The generator expression is a very good example for application of the function IT_IND too. The meaning of the elementary string expression ↓SUC(IT_IND) is not trivial. During the first step of the interation cycle prescribed by ∗[1], the value of the iteration index is equal to 1. On the other hand, the serial number of the first STATEMENT parameter is equal to 2.

3. Implementation

Our implementation is based on SIMULA 67 language and it is now running on the CDC 3300 computer [5]. The system can generate two types of lexical analyzer, such as a lexical analyzer with an extension and without an extension facility. The type of generated lexical analyzer is given at the job control level.

There is a further facility too. If the source text is given without augments, the lexical analyzer containing procedures to execute extensions can be controlled so that, during its running, additional information for extensions will not be created. The hand-written analyzer of the HLP/SZ lexical metalanguage has no extension facilities, because in this case the processing time was the primary point of view. The generated lexical analyzer has automatic error-correction routines, which discontinue "one distance" lexical errors if it is possible. Therefore, a restriction is needed for the first characters of augments. The reader may imagine what happens if, for example, the new tokens called IDENTIFIER and CASE are being recognized in the same action block and the next four input characters are CAPE. In this case it can not be decided whether an error correction must be made or not. In this manner, the first character of an augment text must differ from the first characters of the other tokens assigned to the same block. In exchange for this, the augment text can be more exactly recognized. Information, needed to the error-correcting, parameter-generating and iteration-calculating routines, can be computed during the generation of the final states. Generator expressions are embedded into the generated lexical analyzer in a special tree language, to promote the fast evaluation of these.

Abstract

The Hungarian version of the Helsinki Language Processor, HLP/SZ [1], consists of modules for the lexical, syntactic and semantic processing of programming languages. The lexical metalanguage of our system has been modified so as to introduce the possibility of language extension. Augment texts, expressed in terms of constructs which are not part of the base language, are defined by token classes. The generator expressions, which are assigned to the token class names in action blocks, generate the definition texts. Definition texts can contain additional augments in a recursive way, and it will be processed according to an optional action block. Generator expressions built up from terminals, parameters and functions are controlled by "McCarthy expressions". The system implemented in SIMULA 67 language is now running on the CDC 3300 computer.

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

References

[1] GYIMÓTHY, T., E. SIMON and Á. MAKAY, An implementation of the HLP, Acta Cybernetica, Tom. 6, Fasc. 3, pp. 315—327.

[2] RÄIHÄ, K. J., M. SAARINEN, M. SARJAKOSKI, S. SIPPU, E. SOISALON-SOININEN and M. TEINARI, Revised report on the compiler writing system HLP78, University of Helsinki, Report A—1983-1, 130 pp.

[3] SIPPU, S., Syntax error handling in compilers, University of Helsinki, Report A-1981-1, 100 pp.

[4] SIMON, E., Language design objectives and the CHANGE system, Computational Linguistics and Computer Languages, v. 15, 1982, pp. 229—247.

[5] SIMON, E. and T. GYIMÓTHY, Using attribute grammars to generate compilers, Információ Elektronika, v. 1984, 2, in Hungarian.

[6] SOLNTSEFF, N. and A. YEZERSKY, A survey of extensible programming languages, McMaster University Hamilton, Technical Report No. 71—7, 143 pp.

(Received Jan. 26, 1984)

Some results about keys of relational schemas

HO THUAN and LE VAN BAO

1. §.

In this section we recall some important notions and results in the theory of relational data base needed in subsequent sections.

In this paper, when we talk about a set of tuples the word relation is used while talking about structural description of sets of tuples we use the word relational schema [1]. With this approach a relation is an instance of a relational schema.

A *relation* on the set of attributes $\Omega=\{A_1, A_2, \ldots, A_n\}$ is a subset of the cartesion product Dom $(A_1)\times$Dom $(A_2)\times\ldots\times$Dom (A_n) where Dom (A_i) — the domain of A_i — is the set of possible values for that attribute. The elements of the relation are called *tuples* and will be denoted by $\langle t\rangle$.

A constraint involving the set of attributes $\{A_1, A_2, \ldots, A_n\}$ is a predicate on the collection of all relations on this set. A relation $R(A_1, A_2, \ldots, A_n)$ obeys the constraint if the value of the predicate for R is "true".

We shall restrict ourselves to the case of functional dependencies.

A *functional dependency* (abbr. FD) is a sentence denoted $\sigma\colon X\rightarrow Y$ where σ is the name of the functional dependency and X and Y are sets of attributes. A functional dependency $\sigma\colon X\rightarrow Y$ holds in R (Ω) where X and Y are subsets of Ω, if for every tuple u and $v\in R$, $u[X]=v[X]$ implies $u[Y]=v[Y]$ ($u[X]$ denotes the projection of the tuple u on X).

Let F be a set of functional dependencies. A relation R defined over the attributes $\Omega=\{A_1, A_2, \ldots, A_n\}$ is said to be an *instance* of the *relational schema* $S=\langle\Omega, F\rangle$ iff each functional dependency $\sigma\in F$ holds in R.

There are inference rules — Armstrong's axioms — which can be applied to deriving further functional dependencies from F, and they are listed below. The system of Armstrong's rules is complete in the sense of Ullman [3].

Armstrong's axioms.[1]

For every $X, Y, Z\subseteq\Omega$,

[1] In fact we used here a system of axioms which is equivalent to that of Armstrong.

A1. (Reflexivity): if $Y \subseteq X$ then $X \to Y$.
A2. (Augmentation): if $X \to Y$ then $X \cup Z \to Y \cup Z$.
A3. (Transitivity): if $X \to Y$ and $Y \to Z$ then $X \to Z$.
From the Armstrong's axioms it is easy to prove the following inference rules:
Union rule: if $X \to Y$ and $X \to Z$ then $X \to Y \cup Z$.
Decomposition rule: if $X \to Y$ and $Z \subset Y$ then $X \to Z$.

Let F be a given set of FDs. The closure F^+ of F is the set of FDs which can be derived from F through Armstrong's inference rules.

It is shown in [3] that

$$(X \to Y) \in F^+ \Leftrightarrow Y \subseteqq X^+$$

where

$$X^+ = \{A_i/(X \to A_i) \in F^+\}$$

is the closure of X w.r.t. F.

There is a linear-time algorithm, proposed by Beeri and Bernstein [4], for computing the closure X^+ of a given set X (w.r.t F):

1) Establish the sequence $X^{(0)}, X^{(1)}, \ldots,$ as follows:

$$X^0 \equiv X.$$

Suppose $X^{(i)}$ is computed then

$$X^{(i+1)} = X^{(i)} \cup Z^{(i)}$$

where

$$Z^{(i)} = \bigcup_{X_j \subseteqq X^{(i)}} Y_j$$

$$(X_j \to Y_j) \in F$$

2) In view of the construction, it is obvious that

$$X^{(0)} \subseteqq X^{(1)} \subseteqq X^{(2)} \subseteqq \ldots$$

Since Ω is a finite set, there exists a smallest non negative integer t such that

$$X^{(t)} = X^{(t+1)}$$

3) We have $X^+ = X^{(t)}$.

Keys of a relational schema.

Let $S = \langle \Omega, F \rangle$ be a relational schema and let X be a subset of Ω.

X is a key of S if it satisfies the following two conditions:

(i) $$(X \to \Omega) \in F^+,$$

(ii) $$\bar{\exists} X' \subset X: (X' \to \Omega) \in F^+.$$

The subset X which satisfies only (i) is called a *super key* of S.

In the following, instead of $(X \to Y) \in F^+$, we shall write $X \xrightarrow{*} Y$.

2. §.

Let $S=\langle\Omega, F\rangle$ be a relational schema, where

$$\Omega = \{A_1, A_2, \dots, A_n\},$$

$$F = \{L_i \to R_i/i = 1, 2, \dots, k; \quad L_i, R_i \subseteqq \Omega\}.$$

Let us denote

$$L = \bigcup_{i=1}^{k} L_i \quad \text{and} \quad R = \bigcup_{i=1}^{k} R_i.$$

Without loss of generality, in this paper we assume that

$$L_i \cap R_i = \emptyset, \quad i = 1, 2, \dots, k.$$

(Results without this assumption can be found in [5], [6].) We have the following lemmas.

Lemma 1. Let $S=\langle\Omega, F\rangle$ be a relational schema.
If $A\bar{\in}L$ and $X\xrightarrow{*}Y$ then

$$X\setminus\{A\} \xrightarrow{*} Y\setminus\{A\}.$$

Proof. From the algorithm for computing the closure X^+ of X w.r.t. F (see 1. §.) it is easy to prove by induction that if $A\bar{\in}L$ then $(X\setminus A)^+$ is equal either to X^+ or to $(X^+\setminus A)$.

On the other hand, $X\xrightarrow{*}Y$ implies $Y\subseteqq X^+$.

Hence $$Y\setminus A \subseteqq X^+\setminus A.$$

(i) The case $(X\setminus A)^+=X^+$. From $Y\setminus A\subseteqq X^+\setminus A$, it is clear that

$$Y\setminus A \subseteqq X^+ = (X\setminus A)^+.$$

Hence

$$X\setminus A \xrightarrow{*} Y\setminus A.$$

(ii) The case $(X\setminus A)^+=X^+\setminus A$.
From $Y\setminus A\subseteqq X^+\setminus A$, it is obvious that

$$Y\setminus A \subseteqq (X\setminus A)^+ = X^+\setminus A.$$

Hence

$$X\setminus A \xrightarrow{*} Y\setminus A.$$

Lemma 2. Let $S=\langle\Omega, F\rangle$ be a relational schema, $X\subseteqq\Omega$. If $A\in X$ and $X\setminus A\xrightarrow{*} A$ then X is not a key of S.

Proof. By the hypothesis of the lemma,

$$X\setminus A \xrightarrow{*} A.$$

On the other hand, it is obvious that

$$X\setminus A \xrightarrow{*} X\setminus A.$$

Applying the union rule, we obtain

$$X\setminus A \xrightarrow{*} X.$$

Since $A\in X$ then $X\setminus A\subset X$, showing that X is not a key.

We are now in a position to prove the following theorem.

Theorem 1. Let $S=\langle\Omega, F\rangle$ be a relational schema and X a key of S. Then

$$\Omega\setminus R \subseteqq X \subseteqq (\Omega\setminus R)\cup(L\cap R).$$

Proof. We shall begin with showing that

$$\Omega\setminus R \subseteqq X.$$

First we observe that $X^+\subseteqq X\cup R$. Since X is a key, obviously $X^+=\Omega$.

Hence $X\cup R=\Omega$. This implies that

$$\Omega\setminus R \subseteqq X.$$

It remained to show that:

$$X \subseteqq (\Omega\setminus R)\cup(L\cap R). \tag{1}$$

It is clear that

$$X \subseteqq \Omega = (\Omega\setminus R)\cup(L\cap R)\cup(R\setminus L). \tag{2}$$

To obtain (1), it is therefore sufficient to prove that

$$X\cap(R\setminus L) = \emptyset.$$

Assume the contrary. Then, there would exist an attribute $A\in X$, $A\in R$ and $A\bar{\in} L$. Since X is a key, we have $X\xrightarrow{*}\Omega$. Since $A\bar{\in} L$, we refer to Lemma 1 to deduce

$$X\setminus\{A\} \xrightarrow{*} \Omega\setminus\{A\}.$$

On the other hand, from $A\bar{\in} L$ and $L\subseteqq\Omega$, we have $L\subseteqq\Omega\setminus A$. Hence $\Omega\setminus A\xrightarrow{*}L$.

Applying the transitivity rule for the sequence $X\setminus A\xrightarrow{*}\Omega\setminus A\xrightarrow{*}L\xrightarrow{*}R\xrightarrow{*}A$ (since $A\in R$) we obtain

$$X\setminus A\xrightarrow{*} A \quad \text{with} \quad A\in X.$$

By virtue of Lemma 2, this contradicts the hypothesis that X is a key.

Thus we have proved that if X is a key, then $X\cap(R\setminus L)=\emptyset$.

From (2) we deduce that

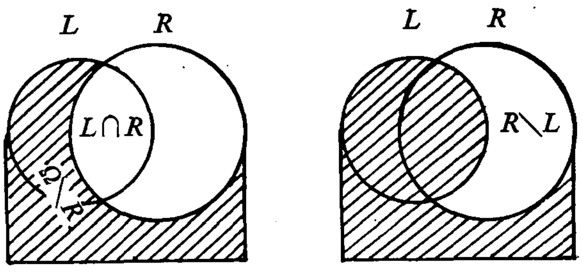
$$X \subseteqq (\Omega\setminus R)\cup(L\cap R).$$

The proof is complete.

Remark 1. Theorem 1 can be deduced from Lemma 6 in Békéssy's and Demetrovics' paper [9]. Here another formal proof has been given. Theorem 1 is illustrated by Fig. 1, where X is an arbitrary key of the relational schema $S=\langle\Omega, F\rangle$.
In view of Theorem 1, it is seen that the keys of $S=\langle\Omega, F\rangle$ are different only on the attributes of $L\cap R$. In other words, if X_1 and X_2 are two different keys of S, then

$$X_1\setminus X_2\subset L\cap R \quad \text{and} \quad X_2\setminus X_1\subset L\cap R.$$

Let $\mathscr{K}(\Omega, F)$ denote the set of all keys of $S=\langle\Omega, F\rangle$, and $\mathscr{S}(Z)$ — the maximal cardinality Sperner system on a set Z [7].



$\Omega\setminus R\subseteq X$

$X\subseteq(\Omega\setminus R)\cup(L\cap R)$

Fig. 1

As an immediate consequence of Theorem 1 and results in [8], [9], we have the following corollaries.

Corollary 1. Let $S=\langle\Omega, F\rangle$ be a relational schema. Then

$$\#\mathcal{K}(\Omega, F)\leqq\#\mathcal{S}(L\cap R)=C_h^{[h/2]}$$

where $h=\#(L\cap R)$.

Corollary 2. Let $S=\langle\Omega, F\rangle$ be a relational schema, and X a key of S. Then

$$\#(\Omega\setminus R)\leqq\#X\leqq\#(\Omega\setminus R)+\#(L\cap R).$$

Corollary 3. Let $S=\langle\Omega, F\rangle$ be a relational schema.
If $R\setminus L\neq\emptyset$ then there exists a key X such that $X\neq\Omega$ (non trivial key).

Corollary 4. Let $S=\langle\Omega, F\rangle$ be a relational schema.
If $L\cap R=\emptyset$ then $\#\mathcal{K}(\Omega, F)=1$ and $\Omega\setminus R$ is the unique key of S.

Theorem 2. Let $S=\langle\Omega, F\rangle$ be a relational schema, where

$$L\cap R=\{A_{t_1}, A_{t_2}, \ldots, A_{t_h}\}\subseteqq\{A_1, A_2, \ldots, A_n\}=\Omega.$$

Let us define

$$K(1)=(\Omega\setminus R)\cup(L\cap R),$$

$$K(i+1)=\begin{cases}K(i)\setminus A_{t_i} & \text{if } K(i)\setminus A_{t_i}\xrightarrow{*}A_{t_i},\\ K(i) & \text{if } K(i)\setminus A_{t_i}\overset{*}{\not\rightarrow}A_{t_i}\end{cases}$$

$$\text{with } i=1, 2, \ldots, h.$$

Then $K(h+1)$ is a key of $S=\langle\Omega, F\rangle$.

Proof. We shall begin with showing that

$$K(i+1)\xrightarrow{*}K(i).$$

Two cases can occur:

a) If $K(i)\setminus A_{t_i}\overset{*}{\not\rightarrow}A_{t_i}$ then from the definition of $K(i+1)$ we have $K(i+1)=$ $=K(i)$ and it is obvious that

$$K(i+1)\xrightarrow{*}K(i).$$

b) If $K(i)\setminus A_{t_i} \xrightarrow{*} A_{t_i}$, we have

$$K(i+1) = K(i)\setminus A_{t_i}.$$

On the other hand, it is obvious that

$$K(i)\setminus A_{t_i} \xrightarrow{*} K(i)\setminus A_{t_i}.$$

Applying the union rule, we get:

$$K(i)\setminus A_{t_i} \xrightarrow{*} K(i).$$

Therefore

$$K(i+1) \xrightarrow{*} K(i).$$

So we have:

$$K(h+1) \xrightarrow{*} K(h) \xrightarrow{*} \dots \xrightarrow{*} K(1).$$

From the above definition of $K(i+1)$ it is clear that

$$K(h+1) \subseteqq K(h) \subseteqq \dots \subseteqq K(1).$$

We are now in a position to prove the theorem. As an immediate consequence of Theorem 1, $K(1)=(\Omega\setminus R)\cup(R\cap L)$ is a superkey of $\langle\Omega, F\rangle$. On the other hand $K(h+1) \xrightarrow{*} K(1)$ showing that $K(h+1)$ is a superkey. To complete the proof, it remains to show that $K(h+1)$ is a key.

Were it false, there would exist a key $\bar{X}$ such that $\bar{X} \subset K(h+1)$, and using the result of Theorem 1 we find

$$\Omega\setminus R \subseteqq \bar{X} \subset K(h+1) \subseteqq (\Omega\setminus R)\cup(L\cap R)$$

clearly, there exist

$$A_{t_i} \in K(h+1)\cap(L\cap R)\setminus\bar{X}$$
$$\text{with} \quad 1 \leqq i \leqq h.$$

From the definition of $K(i+1)$, we find $K(i)\setminus A_{t_i} \overset{*}{\nrightarrow} A_{t_i}$. Since $K(h+1) \subseteqq K(i)$ it follows that $K(h+1)\setminus A_{t_i} \overset{*}{\nrightarrow} A_{t_i}$. On the other hand $\bar{X} \subseteqq K(h+1)\setminus A_{t_i}$. Therefore $\bar{X} \overset{*}{\nrightarrow} A$ which conflicts with the fact that $\bar{X}$ is a key of $\langle\Omega, F\rangle$.

The proof is complete.

It is natural to ask whether the results formulated in Theorem 1 can be improved. The answer is in the affirmative as the following lemmas and theorems show.

Lemma 3. Let $S=\langle\Omega, F\rangle$ be a relational schema and X a key of S. Then

$$X\cap R\cap(L\setminus R)^+ = \emptyset.$$

Proof. Suppose it were not so then there would exist on attribute A such that $A \in X\cap R\cap(L\setminus R)^+$, thus $A \in X$, $A \in R$ and $L\setminus R \xrightarrow{*} A$. Since $A \in R$, it follows that $A \notin (L\setminus R)$.

On the other hand, it is clear that

$$L\setminus R \subseteqq \Omega\setminus R.$$

Taking Theorem 1 into account we get

$$L\setminus R \subseteqq \Omega\setminus R \subseteqq X.$$

Thus

$$L\setminus R \subseteqq X\setminus A \quad (\text{since } A \notin L\setminus R).$$

Evidently $X\setminus A \xrightarrow{*} L\setminus R \xrightarrow{*} A$, where $A\in X$.
By Lemma 2, this contradicts the hypothesis that X is a key of S. The proof is complete.
We define $a(L, R)=(L\setminus R)^{+}\cap(L\cap R)$.
It is clear that $a(L, R)\subseteqq(L\setminus R)^{+}\cap R$.
From this: $X\cap a(L, R)=\emptyset$.
Combining with Theorem 1, the following theorem is obvious.

Theorem 3. Let $S=\langle\Omega, F\rangle$ be a relational schema, and X a key of S. Then:

$$(\Omega\setminus R)\subseteqq X\subseteqq(\Omega\setminus R)\cup((L\cap R)\setminus a(L, R)).$$

Here is an example where $a(L, R)\neq\emptyset$.

Example 1. $\Omega=\{A, B, H, G, Q, M, N, V, W\}$

$$F=\{A\to B,\ B\to H,\quad G\to Q,\quad V\to W,\quad W\to V\}$$

From this we have

$L=ABGVW;\quad R=BHQVW,\quad L\cap R=BVW;$

$L\setminus R=AG;\quad (L\setminus R)^{+}=AGBHQ$

$a(L, R)=(L\setminus R)^{+}\cap(L\cap R)=\{B\}\neq\emptyset.$

Remark 2. It is worth noticing that $(\Omega\setminus R)^{+}=(\Omega\setminus(L\cup R))\cup(L\setminus R)^{+}$.
Therefore, if X is a key of S then obviously:

$$X\cap R\cap(\Omega\setminus R)^{+}=X\cap R\cap(L\setminus R)^{+}=\emptyset$$

and

$$(\Omega\setminus R)\cup\{(L\cap R)\setminus(\Omega\setminus R)^{+}\}=(\Omega\setminus R)\cup\{(L\cap R)\setminus a(L, R)\}.$$

Remark 3. Using Theorem 3, Corollaries 1, 2, 3 deduced from Theorem 1 above, can be improved, as well.

3. §.

Based on Theorems 1 and 2, we now propose some algorithms for the key finding and key recognition problem.

It is worth recalling that:
(i) X is a superkey of $S=\langle\Omega, F\rangle$ iff $X^{+}=\Omega$;
(ii) $X\xrightarrow{*}Y$ iff $Y\subseteqq X^{+}$.

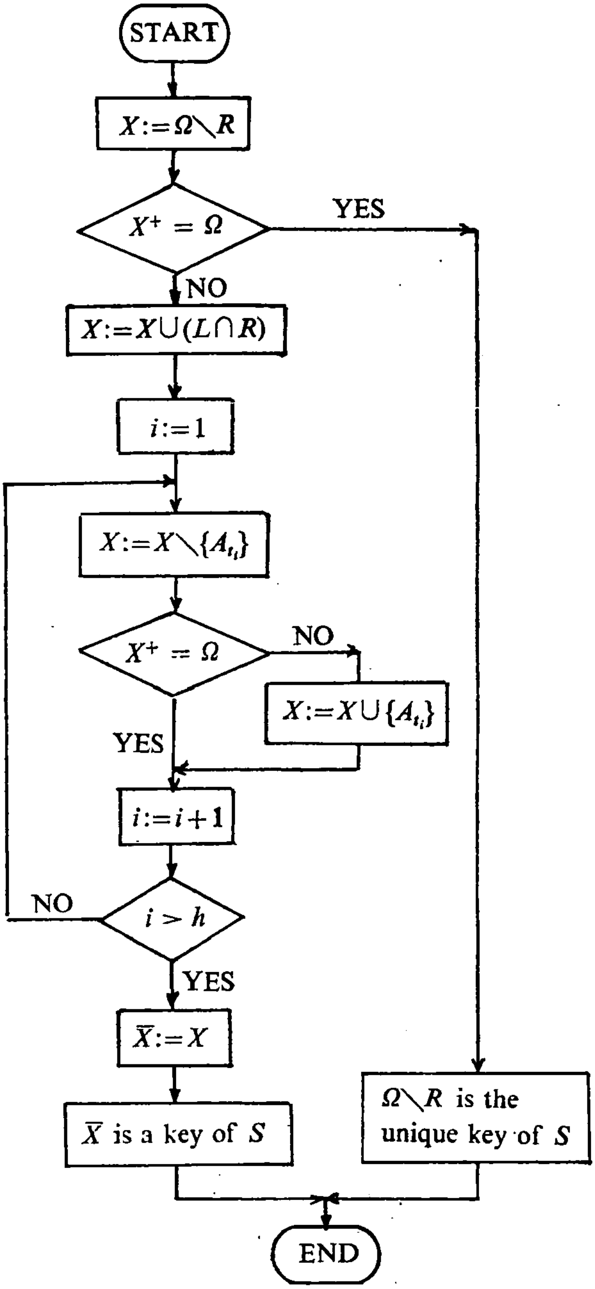
Algorithm 1.
Algorithm for finding one key of the relational schema $S=\langle\Omega, F\rangle$ where

$$\Omega=\{A_1, A_2, \ldots, A_n\},$$

$$F=\{L_i\to R_i/i=1, 2, \ldots, k;\ L_i, R_i\subseteqq\Omega\},$$

$$L=\bigcup_{i=1}^{k}L_i,\quad R=\bigcup_{i=1}^{k}R_i,$$

$$L\cap R=\{A_{t_1}, A_{t_2}, \ldots, A_{t_h}\}.$$


START
$X := \Omega \setminus R$
$X^+ = \Omega$
YES
NO
$X := X \cup (L \cap R)$
$i := 1$
$X := X \setminus \{A_{t_i}\}$
$X^+ = \Omega$
NO
$X := X \cup \{A_{t_i}\}$
YES
$i := i + 1$
NO
$i > h$
YES
$\bar{X} := X$
$\bar{X}$ is a key of S
$\Omega \setminus R$ is the unique key of S
END

Example 2.

The following example illustrates the performance of the Algorithm 1.

Let $S=\langle \Omega, F\rangle$ be a relation scheme with

$$\Omega = \{A, B, C, D, E, G\}$$

and

$$F = \{B \to C, \quad C \to B, \quad A \to GD\}.$$

From this we have

$$L = BCA, \quad R = BCGD,$$

$$\Omega \setminus R = EA, \quad L \cap R = BC.$$

Since $(\Omega\setminus R)^+ = (EA)^+ = EAGD \neq \Omega$, $(\Omega\setminus R)$ is not a key of $S=\langle \Omega, F\rangle$. From the block ③, the algorithm begins with the superkey $X=EABC$. With $A_{t_1}=B$ and $A_{t_2}=C$, we have the sequence

$$X := X\setminus\{B\} = EAC; \quad (EAC)^+ = EACBGD = \Omega,$$

$$X := X\setminus\{C\} = EA; \quad (EA)^+ = EAGD \neq \Omega,$$

$$X := X\cup\{C\} = EAC; \quad \bar{X} := EAC.$$

We obtained a key of S, being $\bar{X}=EAC$.

Similarly, if we start with the same superkey

$$X = EABC$$

but with $A_{t_1}=C$ and $A_{t_2}=B$, then after the termination of Algorithm 1, we obtain another key of the relational schema $S=\langle \Omega, F\rangle$, being EAB.

Algorithm 2.

Algorithm for finding one key of the relational schema $S=\langle \Omega, F\rangle$ included in a given superkey X.

Suppose that $\bar{X}$ is a key included in X.

Then $\bar{X} \subseteq X$.

On the other hand, from Theorem 1:

$$\Omega\setminus R \subseteqq \bar{X} \subseteqq (\Omega\setminus R)\cup(L\cap R).$$

Therefore

$$\bar{X} \subseteqq (\Omega\setminus R)\cup(X\cap(L\cap R)).$$

Thus we can start with the superkey

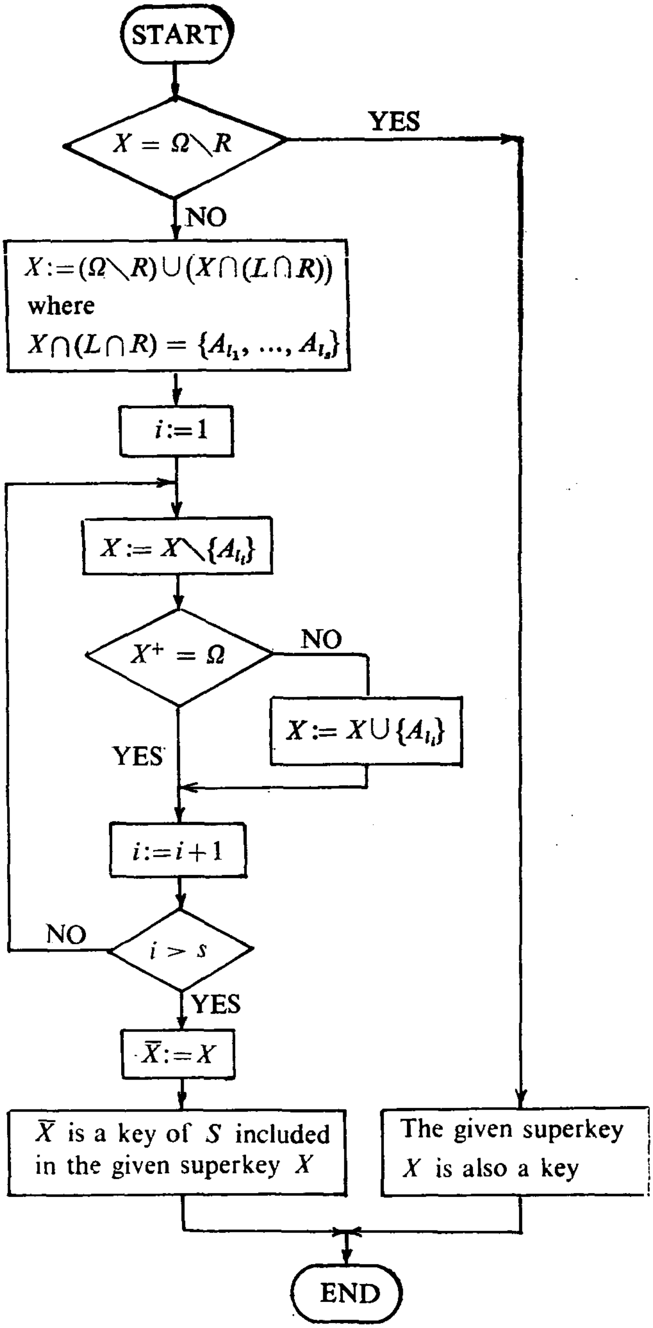
$$(\Omega\setminus R)\cup(X\cap(L\cap R))$$

for finding a key included in a given superkey X.

It is easily seen that Algorithm 2 is similar to Algorithm 1 but block ③ is replaced by the assignment

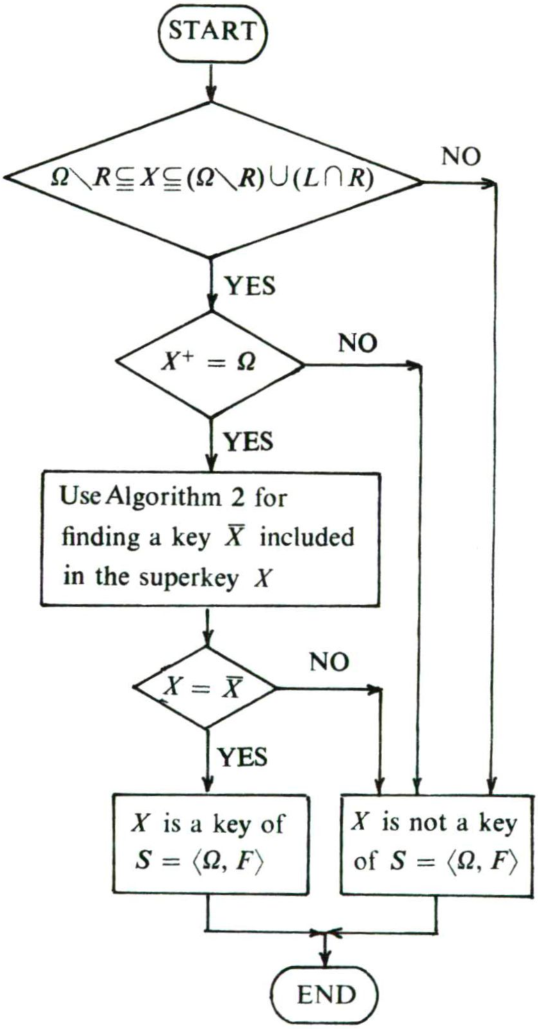
$$X := (\Omega \setminus R) \cup (X \cap (L \cap R))$$

with $X \cap (L \cap R) = \{A_{l_1}, A_{l_2}, \ldots, A_{l_s}\}$ and there are, in addition, some non significant modifications.


START
$X = \Omega \setminus R$
YES
NO
$X := (\Omega \setminus R) \cup (X \cap (L \cap R))$ where $X \cap (L \cap R) = \{A_{l_1}, \ldots, A_{l_s}\}$
$i := 1$
$X := X \setminus \{A_{l_i}\}$
$X^+ = \Omega$
NO
$X := X \cup \{A_{l_i}\}$
YES
$i := i+1$
NO
$i > s$
YES
$\bar{X} := X$
$\bar{X}$ is a key of S included in the given superkey X
The given superkey X is also a key
END

Algorithm 3.

Algorithm for recognition whether a given subset X $(X \subseteqq \Omega)$ is a key of $S = \langle \Omega, F \rangle$.


START
$\Omega \setminus R \subseteqq X \subseteqq (\Omega \setminus R) \cup (L \cap R)$
NO
YES
$X^+ = \Omega$
NO
YES
Use Algorithm 2 for finding a key $\bar{X}$ included in the superkey X
$X = \bar{X}$
NO
YES
X is a key of $S = \langle \Omega, F \rangle$
X is not a key of $S = \langle \Omega, F \rangle$
END

Remark 4. The Algorithms 1, 2, and 3 can easily be improved using Theorem 3.

Lemma 4. Let $S=\langle \Omega, F\rangle$ be a relational schema.
Then

$$G\cap R=\emptyset,$$

where $G=\bigcap_{X_i\in\mathscr{X}\langle\Omega, F\rangle} X_i$ is the intersection of all keys of S.

Proof. It is sufficient to prove that for each $A\in R$ there exists a key X of S such that $A\overline{\in}X$.

In fact, from $A\in R$ we deduce that A belongs to some R_i. Consider the functional dependency

$$L_i\to R_i,\quad (L_i\cap R_i=\emptyset)$$

therefore

$$A\overline{\in}L_i.$$

It is easily seen that:

$$L_i\cup\{\Omega\setminus(L_i\cup R_i)\}\xrightarrow{*}\Omega$$

and

$$A\overline{\in}L_i\cup\{\Omega\setminus(L_i\cup R_i)\},$$

showing that $L_i\cup\{\Omega\setminus(L_i\cup R_i)\}$ is a superkey of S. This superkey includes a key X such that $A\overline{\in}X$.

From this $G\cap R=\emptyset$.

Theorem 4. Let $S=\langle\Omega, F\rangle$ be a relational schema. Then

$$G=\Omega\setminus R.$$

Proof. As an immediate consequence of Lemma 4 we have

$$G\subseteqq\Omega\setminus R.$$

On the other hand it is easily seen by Theorem 1 that

$$\Omega\setminus R\subseteqq G.$$

Hence

$$G=\Omega\setminus R.$$

The proof is complete.

In most cases it is easier to compute $\Omega\setminus R$ than to compute first all keys directly and take their intersection.

Theorem 5. Let $S=\langle\Omega, F\rangle$ be a relational schema satisfying the following condition

$$\forall_i(R_i\cap L\neq\emptyset\Rightarrow L_i\cap R=\emptyset).$$

Then S has exactly one key and $\Omega\setminus R$ is this unique key.

Proof. Let $C=\Omega\setminus(L\cup R)$. Since $L\xrightarrow{*}R$, consequently

$$L\cup C\xrightarrow{*}L\cup C\cup R=\Omega.$$

Let $I=\{i, R_i\cap L\neq\emptyset\}$.

Evidently

$$\bigcup_{i\in I} L_i \cap R = \emptyset \tag{3}$$

and

$$L\cap R \subseteqq \bigcup_{i\in I} R_i. \tag{4}$$

It is obvious that

$$\bigcup_{i\in I} R_i \xrightarrow{*} L\cap R.$$

On the other hand we have

$$\bigcup_{i\in I} L_i \xrightarrow{*} \bigcup_{i\in I} R_i.$$

From (4), clearly

$$\bigcup_{i\in I} L_i \xrightarrow{*} L\cap R.$$

From (3) we have

$$\bigcup_{i\in I} L_i \subseteqq L\setminus R.$$

Hence

$$L\setminus R \xrightarrow{*} \bigcup_{i\in I} L_i \xrightarrow{*} L\cap R.$$

From this we get

$$L\setminus R \xrightarrow{*} (L\setminus R)\cup(L\cap R).$$

That is $L\setminus R \xrightarrow{*} L$.

Using $L\cup C \xrightarrow{*} \Omega$, we have

$$(L\setminus R)\cup C \xrightarrow{*} \Omega.$$

Evidently $(L\setminus R)\cup C = \Omega\setminus R$ is a superkey of S. By Theorem 1, $S=\langle\Omega, F\rangle$ has $(\Omega\setminus R)$ as the unique key.

Theorem 6. Let $S=\langle\Omega, F\rangle$ be a relational schema, X a superkey of S. If $X\cap R=\emptyset$ then X is the unique key of S.

Proof. From $X\cap R=\emptyset$, it is obvious that $X\subseteqq \Omega\setminus R$. Since X is a superkey of S, there exist a key $\overline{X}\subseteqq X$. Using Theorem 1, clearly

$$(\Omega\setminus R) \subseteqq \overline{X} \subseteqq X \subseteqq (\Omega\setminus R)$$

showing that $\Omega\setminus R$ is the unique key of S.

Theorem 7. Let $S=\langle\Omega, F\rangle$ be a relational schema and X a superkey. Then X is a unique key of S iff $X\cap R=\emptyset$.

Proof. The sufficiency of this theorem is essentially Theorem 6. We have only to prove the necessity. Let X be the unique key of $S=\langle\Omega, F\rangle$. Assume the contrary that $X\cap R\neq\emptyset$. Then we should have $A\in X\cap R$.

Evidently $A\in R$ and $A\in X$. In view of Lemma 4, there exists a key $\overline{X}$ such that $A\overline{\in}\overline{X}$. Thus $X, \overline{X}$ are different keys of S, which contradicts the condition that X is the unique key of S.

Theorem 8. Let $S=\langle\Omega, F\rangle$ be a relational schema and let $A\in\Omega$ satisfies the following conditions: for all L_i,

(i) $$A\in L_i \Rightarrow L_i\setminus A \xrightarrow{*} A, \quad \text{and}$$

(ii) $$A\overline{\in} L_i \Rightarrow A\in L_i^+.$$

Then A is a non prime attribute, that is $A\overline{\in} H$, where

$$H = \bigcup_{X_i\in \mathcal{X}(\Omega,F)} X_i.$$

Proof. The proof is by contradiction. Assume the contrary that $A\in H$. Then there would exist a key X of S such that $A\in X$, and an L_i such that $L_i \subseteqq X$.

(i) If $A\in L_i$ then by the hypothesis of the theorem, we have

$$L_i\setminus A \xrightarrow{*} A.$$

Consequently

$$X\setminus A \xrightarrow{*} L_i\setminus A \xrightarrow{*} A$$

which, by Lemma 2, contradicts the fact that X is a key of S.

(ii) If $A\overline{\in} L_i$ then by the hypothesis of the theorem, we have $A\in L_i^+$.

Since $A\overline{\in} L_i$, consequently

$$L_i \subseteqq X\setminus A.$$

Hence

$$X\setminus A \xrightarrow{*} L_i \xrightarrow{*} A$$

which contradicts the fact that X is a key of S.

Thus $A\overline{\in} H$. The proof is complete.

Example 3.

$$\Omega = \{A_1, A_2, A_3, A_4, A_5, A_6\} \quad \text{and}$$

$$F = \{A_1 \rightarrow A_3A_5;\ A_3A_4 \rightarrow A_1A_6;\ A_1A_5A_6 \rightarrow A_2A_4\}.$$

It is easy to verify that A_5 satisfies all conditions of Theorem 8. Therefore $A_5\overline{\in} H$.

Acknowledgment

The authors would like to take this opportunity to express their sincere thanks to Prof. Dr. J. Demetrovics and Dr. A. Békéssy for their valuable comments and suggestions.

Abstract

In this paper we investigate some characteristic properties of a given relational schema $S=\langle\Omega, F\rangle$, in particular the necessary conditions under which a subset X of Ω is a key.

Basing on these results, some effective algorithms are proposed for the key finding problem and key recognition problem. Moreover, a simple explicit formula is given for computing the intersection of all keys of S, as well as sufficient conditions for which a relational schema has exactly one key, and a criterion for which an attribute is a non prime one.

AUTHOR'S ADDRESS:
HO THUAN, LE VAN BAO,
· COMPUTER AND AUTOMATION INSTITUTE OF THE HUNGARIAN
ACADEMY OF SCIENCES, BUDAPEST VICTOR HUGO STR. 18—22.
· INSTITUTE OF COMPUTER SCIENCES AND CYBERNETICS,
BA DINH—LIEU GIAI—HANOI—VIET-NAM

References

[1] DELOBEL, C., Theoretical aspects of modeling in relational data base. Rapport de recherche № 177, July 1979, Université scientifique et médicale et Institut national polytechnique de Grenoble.

[2] ARMSTRONG, W. W., Dependency structures of database relationships, Information processing 74. North Holland Publ. Co. Amsterdam 1974.

[3] ULLMAN, J., Principles of data base systems. Computer Science Press, 1980.

[4] BEERI, C. and BERNSTEIN, P. A., Computational problems related to the design of normal forms for relational schemas, ACM Transactions on Data base systems, vol 4, № 1 March 1979.

[5] LE VAN BAO and HO THUAN, On some properties of keys of relational schemas, Preprint series, № 9, 1983, Hanoi. Institute of Mathematics and Institute of Computer science and Cybernetics.

[6] LE VAN BAO and HO THUAN, Sufficient conditions for which a relational schema has precisely one key, Preprint series, № 7, 1983. Hanoi.

[7] SPERNER, E., Ein Satz über Untermengen einer endlichen Menge, Math. Z. 27 (1928), 544—548.

[8] DEMETROVICS, J., On the number of candidate keys, Information processing letters, vol.7. number 6, October 1978.

[9] BÉKÉSSY A. and J. DEMETROVICS, Contributions to the theory of data bases relations, Discrete Math., 27 (1979) 1—10.

(Received Dec. 12, 1983)

Зависимости в реляционных структурах данных

Б. Тальхайм

I. Введение

В работе рассматривается одна из перспективных моделей обработки информации для банков данных, предложенная Э. Ф. Коддом (5). В этой модели информация представляется в виде отношений, имеющих форму двумерных таблиц, строки которых представляют собой конкретные записи, а стольбцы определяют некоторые области или же аттрибуты.

Поскольку здесь рассматриваем только зависимости (12), т. е. формулы, являющие независимыми от основного множества и верные в тривиальной струкуре, достаточно рассматривать однородные (неструктурированные) отношения.

Пусть G-основное множество. Подмножества R множество G^n называем n-арным отношением. Реляционная структура данных — это структура $\langle G, R\rangle$. В дальнейшем считаем n фиксированным. С любым отношением можно связывать множество $U=\{A_1, ..., A_n\}$ названий столбцов или как принято говорить в литературе множество аттрибутов.

С любым отношением связывается ограничения. Важными и интересными ограничениями являются разные типы зависимостей между типами записей. Не менее интересно нахождение тех аксиом, с помощъю которых можно задать эти зависимости в реляционных базах данных относительно этих вопросов можно посмотреть (1)—(9), (12). В настоящей работе изучается два типа зависимостей, функциональные и декомпозиционные. Изучены общие функциональные зависимости. Для них решена проблема аксиоматизации. Далее изучены бинарные, тернарные и иерархические декомпозиционные зависимости и решена проблема аксиоматизации.

2. Функциональные зависимости

В литературе сначала изучались Э. Ф. Коддом (5) и другими специальные функциональные зависимости, которые в дальнейщем были обобщены Г. Цедли, Я. Деметровичем и Д. Дьепеши (6), (9). Можно еще обобщить эти зависимости и этим упростить математический аппарат.

Для двух элементов $r=(r_1, \ldots, r_n)$, $r'=(r_1', \ldots, r_n')$ отношения R пишем

$$\sigma_i(r, r') = \begin{cases} 0, & \text{если } r_i \neq r_i', \\ I, & \text{если } r_i = r_i' \end{cases} \quad \text{и}$$

$$\sigma(r, r') = \big(\sigma_1(r, r'), \ldots, \sigma_n(r, r')\big).$$

Определение I. Пусть f, g — n-местные Булевы функции. Тогда пара (f, g) называется функциональной зависимостью. Пусть R -отношение и (f, g)-функциональная зависимость. Тогда R удовлетворяет функциональной зависимости $\big(R \models (f, g)\big)$, если для всех r, r' из R имеет место $f\sigma(r, r') \to g\sigma(r, r') = 1$.

Пусть $\mathfrak{R}$ множество в n-арных отношений, $\mathfrak{f}$ множество функциональных зависимостей, $R \in \mathfrak{R}$, $(f, g) \in \Sigma$.

Тогда определим

$$\mathfrak{K}\big((f, g)\big) = \{R' | R'(f, g)\},$$

$$\mathfrak{K}(\Sigma) = \bigcap_{(f,g)\in\Sigma} \mathfrak{K}\big((f, g)\big),$$

$$\mathfrak{M}\mathfrak{f}(R) = \{(f', g') | R \models (f', g')\},$$

$$\mathfrak{M}\mathfrak{f}(\mathfrak{R}) = \bigcap_{R\in\mathfrak{R}} \mathfrak{M}\mathfrak{f}(R).$$

Мы пишем $\Sigma \models (f', g')$, если $\mathfrak{K}(\Sigma) \subseteqq \mathfrak{K}\big((f', g')\big)$. Далее для подмножества $\mathfrak{f}'$ множества $\mathfrak{f}$ всех функциональных зависимостей пишем

$$\mathrm{Cn}_{\mathfrak{f}'}(\Sigma) = \{(f', g') \in \mathfrak{f}' | \Sigma \models (f, g)\}.$$

Для классов Поста (14) K_1, K_2 пусть $\mathfrak{f}(K_1, K_2) = K_1 \times K_2$. Очевидно все известные функциональные зависимости $\big((1), (5), (6), (9)\big)$ являются специальными случаями функциональных зависимостей. Обозначим (14) P_1 класс всех n-местных коньюнкций, S_1 класс всех n-местных дизьюнкции и A_1 класс всех n-местных монотонных функции. Тогда

Кодд-функциональные зависимости (5), (1):	$\mathfrak{f}(P_1, P_1)$,
дуальные зависимости (6), (9):	$\mathfrak{f}(S_1, S_2)$,
строгие зависимости (6), (9):	$\mathfrak{f}(S_1, P_1)$,
слабые зависимости (6), (9):	$\mathfrak{f}(P_1, S_1)$,
монотонные зависимости:	$\mathfrak{f}(A_1, A_2)$.

Пример. Расписание уроков ($U=\{$лектор, лекция, группа, место, время$\}$).

Ограничения:

1) Каждая группа посещает для любого момента времени не более одной лекции.

2) Любой лектор читает в любом моменте времени не более одной лекции.

3) Любое место не может быть занято два раза.

4) Если лекция читается разными лекторами, тогда и слушатели разные.

Эти ограничения можно и легко формулировать нашим аппаратом. Отношение.

лектор	лекция	группа	место	время
1	1	1	1	1
1	2	2	1	2
2	3	1	2	2
2	1	2	2	4
2	4	1	1	3
2	5	1	1	4
3	6	1	1	5
3	3	2	2	1
3	2	2	2	3
4	7	3	1	6
5	8	1	1	7
5	8	2	1	8
6	9	1	2	8
6	9	2	2	7

Эти специальные функциональные зависимости легко представить на языке $\{X \to Y | X, Y \subseteq U\}$. Для Кодд-функциональных зависимостей легко доказывается с помощью фольклорной теоремы Фреге полнота следующей системы (1) правил вывода:

($c\mathfrak{f}$-рефлексивность) $$\frac{}{X \cup Y \to Y}$$

($c\mathfrak{f}$-монотония) $$\frac{X \to Y}{X \cup W \cup Z \to Y \cup Z}$$

($c\mathfrak{f}$-транзитивность) $$\frac{X \to Y,\ Y \to Z}{X \to Z}$$

($c\mathfrak{f}$-объединение) $$\frac{X \to Y,\ X \to Z}{X \to Y \cup Z}$$

($c\mathfrak{f}$-декомпозиция) $$\frac{X \to Y \cup Z}{X \to Y}$$

Аналогичные системы правил вывода для специальных функциональных зависимостей быги представлены в (6).

Для n-местных Булевих функций f, g пишем $f \leqq g$, если для всех $\tilde{\sigma}$ из E_2^n из $f(\tilde{\sigma}) = 1$ следует, что $g(\tilde{\sigma}) = 1$.

Теорема I. Для n-местных Булевых функций $f, f_1, \ldots, f_m, g, g_1, \ldots, g_m$ имеет место $\{(f_i, g_i) | 1 \leqq i \leqq m\} \models (f, g)$ тогда и только тогда, когда имеет место $\mathop{\&}\limits_{i=1}^{m} (f_i \to g_i) \leqq f \to g$.

Доказательство. 1) Докажем утверждение теоремы сначала для $m = 1$.

1.1) Пусть $f_1 \to g_1 \not\models f \to g$, т. е. существует набор $\tilde{\sigma}$ со свойством $f(\tilde{\sigma}) = f_1(\tilde{\sigma}) = g_1(\tilde{\sigma}) = 1$ и $g(\tilde{\sigma}) = 0$ или $f_1(\tilde{\sigma}) = g(\tilde{\sigma}) = 0$ и $f(\tilde{\sigma}) = 1$. Тогда существует отношение R в $\mathfrak{R}((f_1, g_2))$ такая, что $R \notin \mathfrak{R}((f, g))$.

1.2) Допустим, что в $\mathfrak{R}(f_1, g_1) \setminus \mathfrak{R}((f, g))$ существует отношение R. Тогда существуют в R элементы r, r' такие, что $f\sigma(r, r') \to g\sigma(r, r') = 0$ и $f_1\sigma(r, r') \to g_1\sigma(r, r') = 1$.

2) Доказательство утверждения теоремы для $m = 2$ аналогично.

Из этой теоремы и теорем представления монотонных функций (14) вытекает следующее утверждение, подчеркивающее важность класса слабых зависимостей.

Следствие 1. Для любой зависимости (f, g) из $\mathfrak{f}(A_1, A_1)$ существует эквивалентная ей система Σ слабых зависимостей. Тем самым, поскольку аксиоматизация класса $\mathfrak{f}(A_1, A_1)$ намного сложнее аксиоматизации класса $\mathfrak{f}(P_1, S_1)$, вместо систем монотонных зависимостей удобнее рассматривать системы слабых зависимостей.

Следствие 2. Пусть f, f', g, g' n-местные функции.

1) $\models (f, 1)$, $\models (0, g)$.

2) Если имеют места $f' \leqq f$, $g' \geqq g$, то $(f, g) \models (f', g')$.

3) Если $f \leqq g$, то $\models (f, g)$.

4) $(f, g) \models (\bar{g}, \bar{f})$.

5) Пусть h m-местная монотонная функция и $\Sigma = \{(f_i, g_i) | I \leqq i \leqq m\}$. Тогда имеет место

$$\Sigma = \big(h(f_1, f_2, \ldots f_m),\ h(g_1, g_2, \ldots, g_m)\big).$$

Очевидно, что теорема I обусловливает простые аксиоматизации класса $\mathfrak{f}(K_1, K_2)$. Приведем два из них. Подмножество Σ множества $\mathfrak{f}(K_1, K_2)$ называется замкнутым, если верно

$$\Sigma = Cn_{\mathfrak{f}(K_I, K_2)}(\Sigma).$$

Следствие 3 (9). Подмножество Σ множества $\mathfrak{f}(P_1, S_2)$ является замкнутым тогда и только тогда, когда для лубой зависимости (f, g) из $\mathfrak{f}(P_1, S_1) \setminus \Sigma$ существует функция h из P_5 такая, что верны следующие условия:

1) $f \vee h = f$, $g \& h \neq h$,

2) если для $(f', g') \in \Sigma$ верно $f' \vee h = f$, то $g' \& h = h$.

Следствие 4. Подмножество Σ множества $\mathfrak{f}(A_1, A_1)$ является замкнутым множеством тогда и только тогда, когда для всех зависимостей (f, g) из $\mathfrak{f}(A_1, A_1) \setminus \Sigma$ существует функция h в P_5 такая, что имеет место:

1) $f \vee h = f$, $g \vee h \neq g$,

2) если $(f', g') \in \Sigma$ и $f' \vee h = f'$, то $g' \vee h = g'$.

В множестве $\mathfrak{f}$ можно ввести частичный порядок. Мы пишем $(f_1, g_1) \geqq (f_2, g_2)$, если $f_1 \geqq f_2$ и $g_1 \geqq g_2$. Любое замкнутое множество Σ имеет тогда некоторое подмножество максимальных элементов.

$$\max(f) = \mathop{\&}_{(f, g) \in \Sigma} g; \quad \min(g) = \bigvee_{(f, g) \in \Sigma} f;$$

$$\mathrm{Max}\,(\Sigma) = \{(f, g) \in \Sigma | g = \max(f),\ f = \min(g)\}.$$

Теорема 2. Пусть M замкнутое множество функциональных зависимостей. Зависимость (f, g) является элементом M тогда и только тогда, когда существует в $\mathrm{Max}(M)$ элемент (f', g') такой, что имеет место соотношения $f' \geqq f$, $g' \leqq g$.

Доказательство. Допустим, что (f, g) является элементом множества M. Пусть $f' = \min(g)$ и $g' = \max(\min(g))$. Поскольку $(\min(g), g) \in M$, $f \leqq \min(g)$ по следствию 2 имеют место $f' \geqq f$, $\max(\min(g)) \leqq g$ и таким образом $g' \leqq g$. Функция min является монотонной и поэтому верно $\min(\max(\min(g))) \leqq \leqq \min(g)$. С другой стороны, $(\min(g), \max(\min(g))) \in M$ и $\min(g) \leqq \leqq \min(\max(\min(g)))$, т. е. $\min(g) = \min(\max(\min(g)))$ (f', g') из $\mathrm{Max}(M)$. Если $(f', g') \in \mathrm{Max}(M)$ по следствию 2 имеем $(f, g) \in M$.

Следствие 5. Для всех замкнутых множеств Σ $\langle \mathrm{Max}(\Sigma), \cup, \cap \rangle$ является дистрибутивной структурой, где

$$(f_1, g_1) \cup (f_2, g_2) = (\min(g_1 \vee g_2), g_1 \vee g_2) \text{ и}$$

$$(f_1, g_1) \cap (f_2, g_2) = (f_1 \& f_2, \max(f_1 \& f_2)).$$

Армстронг изучал подобные структуры $\{(f, g) \in P_1 \times P_1 | g = \max(f)\}$ (1), (2). Из следствия 5 и теоремы 2 следует тогда очень важное свойство элементов множества $\mathrm{Max}(M)$.

Следствие 6. Любой элемент множества $\mathrm{Max}(M)$ имеет единственное несократимое представление в виде объединения неразложимых элементов.

Интересно также максимальное число неразложимых элементов структуры $\mathrm{Max}(M)$: 2^n.

Этими утверждениями легко выводить и решения алгоритмических проблем и оценки сложности решения алгоритмических проблем для функциональных зависимостей.

3. Декомпозиционные зависимости

Для определения этих зависимостей введем две операции над отношениями.

Определение 2. 1) Пусть $X = \{A_{i_1}, \ldots, A_{i_k}\} \subseteqq U$. Тогда называется

$$R[X] = \{(r_{i_1}, \ldots, r_{i_k}) | r' \in R\colon\ r_{i_j} = r'_{i_j}\}$$

проэкцией отношения R на (стольбцы из) X.

2) Пусть для $X = \{A_{i_1}, \ldots, A_{i_k}\}$, $Y = \{A_{j_1}, \ldots, A_{j_k}\}$ $X \cup Y = U$ две проекции отношения R. Тогда сверткой отношений $R[X]$ и $R[Y]$ называется множество

$$R[X] * R[Y] = \{(r_1, \ldots, r_n) | (r_{i_1}, \ldots, r_{i_k}) \in R[X],\ (r_{j_1}, \ldots, r_{j_l}) \in R[Y]\}.$$

Определение 3. 1) Покрытие $(X_1, \ldots, X_k)$ множества U называется k-арной декомпозиционной зависимостью.

2) Отношение R удовлетворяет k-арной декомпозиционной зависимости $(X_1, \ldots, X_k)$ $(R \models (X_1, \ldots, X_k))$, если имеет место равенство $R = R[X_1] * R[X_2] * \ldots \ldots * R[X_k]$.

Пусть ϑ_k-множество всех k-арных декомпозиционных зависимостей и ϑ-множество всех декомпозиционных зависимостей.

Как и для функциональных зависимостей можно определить для D, $D \subseteqq \vartheta$, $\tilde{X}$, $\tilde{X} \in \vartheta$, R, $\mathfrak{R}$ множества $\mathfrak{K}(\tilde{X})$, $\mathfrak{K}(D)$, $\mathfrak{N}\vartheta(R)$, $\mathfrak{N}\vartheta(\mathfrak{R})$, $Cn_{\vartheta}(D)$, $Cn_{\vartheta_k}(D)$ и соотношение $D \models \tilde{X}'$.

Удобно и легко формулировать эти зависимости на языке $\mathfrak{L}$ открытой логики предикатов с алфавитом $\{\{(x_1, \ldots, x_n) \in V_1 \times V_2 \times \ldots \times V_n\}, P, \&, \rightarrow, (,)$, где $V_i \cap V_j = \emptyset$ для $i \neq j$ и P n-местный символ.

Определение 4. Формула языка $\mathfrak{L}$ вида

$$P(\tilde{x}_1) \, \& \ldots \& \, P(\tilde{x}_k) \rightarrow P(\tilde{x})$$

называется предикативной зависимостью и называется декомпозиционной зависимостью, если выполнены следующие условия для всех i, j, $i \neq j$, $1 \leqq i \leqq k$, $1 \leqq j \leqq k$:

1) $$|\tilde{x}_i| \cap |\tilde{x}| \nsubseteqq |\tilde{x}_j| \cap |\tilde{x}|$$

2) $$(|\tilde{x}_j| \setminus |\tilde{x}|) \cap (|\tilde{x}_i| \setminus |\tilde{x}| = \emptyset,$$

где $|(y_1, \ldots, y_n)|$ множество $\{y_1, \ldots, y_n\}$.

Оба эти условия для декомпозиционных зависимостей существенные. Множества всех предикативных и всех декомпозиционных зависимостей отличаются существенным образом. Например, существуют в $\mathfrak{L}_{\text{П}}$ бесконечные цепы формул $\langle \alpha_i \rangle, \langle \beta_i \rangle$ такие, что имеют место $\mathfrak{K}(\alpha_i) \nsubseteqq \mathfrak{K}(\alpha_{i+1})$ и $\mathfrak{K}(\beta_i) \nsupseteqq \mathfrak{K}(\beta_{i+1})$ для всех натуральных чисел i. В множестве всех декомпозиционных зависимостей $\mathfrak{L}_{\text{Д}}$ существуют только такие цепы конечной длины и точно один максимальный элемент. Вместе с тем можно и доказать, что для любого подмножества Σ множества $\mathfrak{L}_{\text{Д}}$ существует в $\mathfrak{L}_{\text{П}}$ формула α, эквивалентная этому множеству Σ (т. е. $\mathfrak{K}(\Sigma) = \mathfrak{K}(\alpha)$).

Пусть далее $\mathfrak{L}_{\text{Д}}^k$-множество всех k-компонентных формул из $\mathfrak{L}_{\text{Д}}$.

Следует отметить, что для предикативных зависимостей существует очень простая аксиоматизация, известная как аксиоматизация Фреге—Лукасевича и вновь открытая разными авторами.

Предложение. Формула α из $\mathfrak{L}$ следует из подмножества Σ множества $\mathfrak{L}$ тогда и только тогда, когда она выводима с помощью правил отделения и подстановки из множества и множества аксиом

$$\{\alpha \rightarrow \beta \rightarrow \alpha, \ \alpha \& \beta \rightarrow \alpha, \ \alpha \& \beta \rightarrow \beta, \ \alpha \rightarrow \beta \rightarrow \alpha \& \beta\},$$

$$(\alpha \rightarrow \beta \rightarrow \gamma) \rightarrow (\alpha \rightarrow \beta) \rightarrow (\alpha \rightarrow \gamma).$$

Можно и в множестве ϑ ввести порядок. Для $\tilde{X} = (X_1, \ldots, X_k)$ и $\tilde{Y} = (Y_1, \ldots, Y_l)$ из ϑ пишем $\tilde{X} \leqq \tilde{Y}$, если для всех i, $I \leqq i \leqq k$, существует j, $I \leqq j \leqq l$, такое, что $X_i \subseteqq Y_j$.

Следствие 7. Для $\tilde{X}$, $\tilde{Y}$ из ϑ следует из $\tilde{X} \leqq \tilde{Y}$ что $\tilde{X} \models \tilde{Y}$. Изучим сейчас класс ϑ_2 всех бинарных декомпозиционных зависимостей.

Определение 5. 1) Система D, $D \subseteqq \vartheta_2$. называется монотонна, если $(U, U) \in D$ и из $\tilde{X} \in D$, $\tilde{X} \leqq \tilde{Y}$ следует $\tilde{Y} \in D$.

2) Пусть D-монотонная система. Если для всех (X_1, X_2) и (Y_1, Y_2) из D из того, что

а) $X_1 \cap X_2 = Y_1 \cap Y_2$ следует $(X_1 \cap Y_1, X_2 \cup Y_2) \in D$, то D называется слабо полной;

б) $Y_1 \cap Y_2 = X_2$ следует $(X_1 \cap Y_1, Y_2) \in D$, то D называется АД-полной (2);

в) $X_1 \cap X_2 \subseteqq Y_1$ и $X_2 \subseteqq Y_2$ следует $(X_1 \cap Y_1, Y_2) \in D$, то D называется полной;

г) $X_2 \subseteqq Y_2$ следует $(Y_1 \cap (X_1 \cup Y_2), Y_2) \in D$, то D называется сильно транзитивной;

д) $X_1 \cap X_2 \subseteqq Y_2$ и $Y_2 \subseteqq (X_1 \cap X_2) \cup Y_1$ следует $(Y_1 \cap (X_1 \cup Y_2), Y_2) \in D$, то D называется транзитивной (3).

3) Монотонная система $D, D \subseteqq \vartheta_2$, называется сильно полной, если для всех $(X_1, X_2), (Y_1, Y_2), (Z_1, Z_2)$ из D также и принадлежит D $(Y_1 \cap (X_2 \cup Y_1 \cup Z_1) \cup \cup Y_1 \cap (Y_2 \cup Z_2), X_2 \cap (X_1 \cup Y_2 \cup Z_1) \cup Y_2 \cap (Y_1 \cup Z_2))$.

Следствие 8. Сильно полная система D, $D \subseteqq \vartheta_2$, является полной. Полная система D, $D \subseteqq \vartheta_2$, является слабо полной и АД-полной системой.

Первое утверждеиие очевидно для $(Y_1, Y_2) = (\emptyset, U)$. Можно доказать, что полная система D, $D \subseteqq \vartheta_2$, также сильно полна (Лемма 7) и что существует слабо полная система D, $D \subseteqq \vartheta_2$, не являющаясь полной.

Определение 6. Система D, $D \subseteqq \vartheta$, называется k-замкнутой, если $D = = Cn_{\vartheta_k}(D)$.

Определение 7. Система D, $D \subseteqq \vartheta_2$, удовлетворяет D_2-аксиоме, если для всех $(Z_1, Z_2) \in \vartheta_2 \setminus D$ существует множество $B \subseteqq U$ такое, что

1) $Z_1 \cap Z_2 \subseteqq B$, $Z_1 \nsubseteqq B$, $Z_2 \nsubseteqq B$;

2) если $(X_1, X_2) \in D$, $X_1 \cap X_2 \subseteqq B$, то $X_1 \subseteqq B$ или $X_2 \subseteqq B$.

Определение 8. Пусть $\underline{E} = \{E_1, \ldots, E_k\}$-множество множеств. $\underline{E}$ называется Δ-системой, если для всех $i, j, I \leqq i < j \leqq k$, множество $E_\iota \cap E_j$ не зависит от выбора i и j.

Определение 9. Множество D, $D \subseteqq \vartheta_2$, удовлетворяет D_2'-аксиоме, если существуют натуральное число k и система $\underline{E} = \{E_{ij} \subseteqq U \mid I \leqq i < j \leqq k, \ |E_{ij}| \leqq \leqq n-2\}$, что

1) если $(X, Y) \notin D$, то существуют i, j $(I \leqq i < j \leqq k)$, что $X \cap Y \subseteqq E_{ij}$, $X \nsubseteqq E_{ij}$, $Y \nsubseteqq E_{ij}$;

2) если $(X, Y) \in D$ и $X \cap Y \subseteqq E_{ij}$ для некоторых i, j, то $X \subseteqq E_{ij}$ или $Y \subseteqq E_{ij}$;

3) для всех i, j, l $(I \leqq i < j < l \leqq k)$ $\{E_{ij}, E_{il}, E_{jl}\}$ является Δ-системой.

Теорема 3. Для системы D, $D \subseteqq \vartheta_2$, следующие утверждения эквивалентны:

(а) D 2-замкнута;

(б) D сильно полна;

(в) D полна;

(г) D АД-полна;

(д) D транзитивна;

(е) D сильно транзитивна;

(ё) D удовлетворяет D_2-аксиоме;

(ж) удовлетворяет D_2'-аксиоме.

Соотношение (а)⇒(г) доказано в (2). Там же по определению считали эквивалентными (а) и (г). Как и в (2) доказывается (а)⇒(б). Если (а) и (в) эквивалентны, то по следствию 8 (б) и (в) эквивалентны. По лемме 7 из (в) следует (б). Аналогично можно доказать, что (д), (е) и (в) эквивалентны. Докажем аналогичным способом как и в (9), что (а)⇒(в) (лемма 1), (в)⇒(ё) (лемма 2), (ё)⇒(ж) (лемма 3) и (ж)⇒(а) (лемма 4, 5, 6).

Лемма I. Если D 2-замкнутое множество, то D полна.

Лемма I$^+$. Если $\Sigma = Cn^2(\Sigma)$ для некоторого подмножества Σ множества $\mathfrak{L}^2_{\text{Д}}$, то имеет место для всех $P(\tilde{X}_1)\&P(\tilde{X}_2)\to P(\tilde{X}_3)$ и $P(\tilde{Y}_1)\&P(\tilde{Y}_2)\to P(\tilde{Y}_3)$ из Σ:

1) Если $|\tilde{Z}_{i_1}| \supseteqq |\tilde{X}_1|$ и $|\tilde{Z}_{i_2}| \supseteqq |\tilde{X}_2|$, то $P(\tilde{Z}_1)\&P(\tilde{Z}_2)\to P(\tilde{Z}_3) \in \Sigma$.

2) Если $|\tilde{X}_1| \cap |\tilde{X}_2| \subseteqq |\tilde{Y}_1|$ и $(|\tilde{Y}_1| \setminus |\tilde{Y}_2|) \cap (|\tilde{X}_1| \setminus |\tilde{X}_1|) = \emptyset$, то для $\tilde{Z}$ со свойством $\tilde{Z} \supseteqq |\tilde{X}_1| \cap |\tilde{Y}_1|$, $|\tilde{Z}| \setminus (|\tilde{X}_1| \cap |\tilde{Y}_1|) \cap |\tilde{Z}_2| = \emptyset$ имеет место $P(\tilde{Z})\&P(\tilde{Y}_2)\to P(\tilde{Y}_3)$.

Доказательство леммы I$^+$. Утверждение 1) очевидно. Утверждение 2) следует из истинности формулы $((\alpha_1\&\alpha_2\to\alpha_3)\&(\alpha_3\&\alpha_2\to\alpha_4))\to(\alpha_1\&\alpha_2\to\alpha_4)$. Очевидно, $P(\tilde{Z})\&P(\tilde{Y}_2)\to P(\tilde{Y}_3)$ из $\mathfrak{L}^2_{\text{Д}}$.

Лемма 2. Если множество D полное множество, то D удовлетворяет D_2-аксиоме.

Доказательство. Пусть D полное множество. Допустим, что $(X\cup V,\ V\cup Y) \notin D$ для разбиения $\{X, Y, V\}$ множества U. Тогда существует множество E, $E \subseteqq U$, что $V \subseteqq E$ и E максимально относительно (X, Y), т. е. $(X\cup E,\ Y\cup E) \notin D$ и для всех E', $E' \supsetneqq E$: $(X\cup E',\ Y\cup E') \in D$. Существование такого множества следует из свойства $(U, X) \in D$.

Докажем, что множество E удовлетворяет D_2-аксиоме. Во-первых $V \subseteqq E$. Есло было бы $X \subseteqq E$, то $(E, E\cup Y) \notin D$, т. е. $(E\cup Y) \notin D$, т. е. $(U) \notin D$. Поэтому $X \nsubseteqq E$ и $Y \nsubseteqq E$. Во-вторых, пусть для $(V'\cup X',\ V'\cup Y') \in D$ $V' \subseteqq E$. Допустим, что $X' \nsubseteqq E$ и $Y' \nsubseteqq E$. Тогда для $X'' = X' \setminus E$, $Y'' = Y' \setminus E$ $(E\cup X'',\ E\cup Y'') \in D$. Поэтому из D также $(E\cup X\cup X'',\ E\cup Y\cup X'')$ и $(E\cup X\cup Y'',\ E\cup Y\cup Y'')$. Из $(E\cup X'', E\cup Y'')$ $(E\cup X\cup Y'',\ E\cup Y\cup Y'') \in D$ следует, что $(E\cup X,\ E\cup Y\cup Y'')$, $(E\cup X\cup X'',\ E\cup Y\cup Y'') \in D$. Из $(E\cup Y'',\ E\cup X'')$, $(E\cup X\cup X'',\ E\cup Y\cup X'') \in D$ следует, что $(E\cup(X\cap Y''), E\cup Y\cup X'' \in D)$. Из $(E\cup Y\cup X'',\ E\cup(X\cap Y''))$, $(E\cup Y\cup Y'',\ E\cup X) \in D$ следует, что $(E\cup Y, E\cup X) \in D$. Это противоречит $(E\cup X,\ E\cup Y) \notin D$. Поэтому D удовлетворяет D_2-аксиоме.

Лемма 3. Если D удовлетворяет D_2-аксиоме, то D удовлетворяет D_2'-аксоме.

Доказательство. Для всех $(X, Y) \notin D$ напишем начиная с $i=2$ равенственные множества $E(X, Y) = X \cap Y = E_i$: $E_2, E_3, \ldots, E_k$. Пусть $E_{1j} = E_j$ $(I < j \leqq k)$ и $E_{ij} = E_i \cap E_j$ $(I < i < j \leqq k)$. Условия 1 и 2 уже выполнены множеством $\{E_2, \ldots$‘ $\ldots, E_k\}$. Докажем условие 3.

Пусть $i = I$. Тогда $E_{1j} = E_j$, $E_{1l} = E_l$, $E_{jl} = E_j \cap E_l$ и $\{E_{1j}, E_{1l}, E_{jl}\}$ является Δ-системой.

Пусть $i > I$. Тогда $E_{ij} = E_i \cap E_j$, $E_{jl} = E_j \cap E_l$, $E_{il} = E_i \cap E_l$. Поэтому $\{E_{ij}, E_{jl}, E_{il}\}$ Δ-система.

Лемма 4. Пусть R отношение и r_1, r_2, r_3 разные элементы R. Тогда $\{E(r_1, r_2), E(r_1, r_2), E(r_2, r_3)\}$ Δ-система, где

$$E(r, r') = \{A \in U | r(A) = r'(A)\}.$$

Доказательство очевидно.

Лемма 5. Пусть для всех i, j, l $(I \leqq i < j < l \leqq k)\{E_{ij}, E_{il}, E_{jl}\}$ является Δ-системой и $\underline{E} = \{E_{ij} \subseteqq U | I \leqq i < j \leqq k, |E_{ij}| \leqq n-2\}$. Тогда существует отношение R такое, что

$$\underline{E}(R) = \{E(r, r') | r, r' \in R \ r \neq r'\} = \underline{E}.$$

Доказательство (по индукции). Пусть для $m(<k)$ уже получены $r_1, \ldots, r_m$ со свойством: $(I \leqq i < j \leqq m)$ $E(r_i, r_j) = E_{ij}$. r_{m+1} определим следующим образом:

$$r_{m+1}(A) = \begin{cases} r_i(A), & \text{если } A \in E_{im+1} \quad I \leqq i \leqq m, \\ \max\left(r_i(B) | B \in U, \ I \leqq i \leqq m\right) + I & \text{иначе.} \end{cases}$$

Докажем, что $\{r_1, \ldots, r_{m+1}\}$ удовлетворяет условию леммы.

1) Если $A \in E_{im+1} \cap E_{jm+1}$, то $r_i(A) = r_j(A)$ и $\{E_{ij}, E_{im+1}, E_{jm+1}\}$ Δ-система.

2) Если для i, $I \leqq i \leqq m$, $A \notin E_{im+1}$, то $r_i(A) \neq r_{m+1}(A)$. Действительно, если для j $A \in E_{jm+1}$, то $r_{m+1}(A) = r_j(A)$ и $A \notin E_{ij}$. Поскольку $\{E_{ij}, E_{jm+1}, E_{im+1}\}$ Δ-система и $r_i(A) \neq r_j(A)$, $r_i(A) \neq r_{m+1}(A)$. Если $A \notin \bigcup\limits_{1 \leqq j \leqq m} E_{jm+1}$, то $r_{m+1}(A) \neq$ $\neq r_i(A)$ для всех i, $I \leqq i \leqq m$.

Поэтому существует $R = \{r_1, \ldots, r_k\}$ со свойством $\underline{E}(R) = \underline{E}$.

Лемма 6. Пусть D подмножество множества ϑ_2.

1) Если D удовлетворя D_2-аксиоме, то существует отношение R такое, что $\mathfrak{N}\vartheta_2(R) = D$.

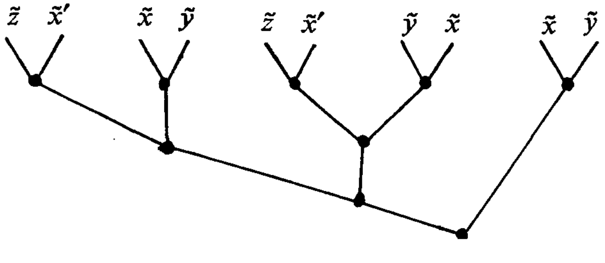
2) Если R отношение, то $\mathfrak{N}\vartheta_2(R)$ удовлетворяет D_2-аксиоме.

Доказательство. 1 Пусть D с $\underline{E} = \{E_{ij} | I \leqq i < j \leqq k\}$ удовлетворяет D_2-аксиоме. Тогда по лемме 5 существует отношение R такое, что $\underline{E}(R) = \underline{E}$ и по аксиоме $D = \mathfrak{N}\vartheta_2(R)$.

2) Пусть R отношение, $R = \{r_1, \ldots, r_k\}$ и $E_{ij} = \underline{E}(r_i, r_j)$. Множество $\{E_{ij} | I \leqq i < j \leqq k\}$ удовлетворяет аксиоме.

Лемма 7. Если D полная система, то D сильно полна.

Доказательство. Пусть $\tilde{X} = (X_1, X_2)$, $\tilde{Y} = (Y_1, Y_2)$, $\tilde{Z} = (Z_1, Z_2)$ из D. Тогда выберем минимальное $\tilde{X}' = (X'_1, X'_2)$ со свойством $\tilde{X} \geqq \tilde{X}$, $Z_1 \cap Z_2 \subseteqq X'_2$, $Z_2 \subseteqq X'_1$. Используя правило $\dfrac{(X_1, X_2), (Y_1, Y_2)}{(X_1 \cap (X_2 \cup Y_1), X_2 \cup Y_2)}$, которая следует из павила полноть следующее дерево вывода даст решение:



Аналогичным путем можно решать и проблему аксиоматизации для класса ϑ_k. Можно, например, доказать, что если система D 3-замкнута, то эта система монотонна и для (X_1, X_2, X_3), (Y_1, Y_2, Y_3) из D также и $(X_1 \cap Y_1, Y_2, Y_3)$ из D, если одно из следующих свойств верна:

1) $(X_i \cap X_j) \subseteqq (Y_i \cap Y_j)$, $(X_k \setminus X_1) \cap (Y_1 \setminus Y_k) = \emptyset$ для всех i, j, k $(I \leqq i < j \leqq 3, 2 \leqq k \leqq 3)$;

2) $Y_1 \setminus Y_2 \subseteqq X_1$, $X_1 \cap X_i \subseteqq Y_1 \cap Y_2$ для i $(2 \leqq i \leqq 3)$;

3) $(X_2 \cap Y_1) \cup (Y_1 \setminus Y_2) \subseteqq X_1$, $X_i \cap X_3 \subseteqq Y_1 \cap Y_2$ для i $(I \leqq i \leqq 2)$.

Определение 10. Зависимость $(X_1, \ldots, X_k)$ из ϑ называется иерархической (8), если $\{X_1, \ldots, X_k\}$ Δ-система.

Пусть $\mathfrak{H}$ множество всех иерархических зависимостей. Множество H иерархических зависимостей называется $\mathfrak{H}$-замкнутой, если $H = Cn_{\mathfrak{H}}(H)$.

Опpеделелние 11. Система H, $H \subseteqq \mathfrak{H}$, удовлетворяет H-аксиоме, если для всех $\tilde{X} = (X_1, \ldots, X_k) \in \mathfrak{H} \setminus H$ существует такое подмножество B, $B \subseteqq U$, что

1) $X_i \cap X_j \subseteqq B$ для некоторых i, j и дляа всех i $(I \leqq i \leqq k)$ $X_i \nsubseteqq B$;

2) если $\tilde{Y} = (Y_1, \ldots, Y_l) \in H$ и $Y_l \cap Y_j \subseteqq B$ для некоторых i, j, то $Y_1 \subseteqq B$ или $Y_2 \subseteqq B$ или … или $Y_k \subseteqq B$.

Совершенно аналогично теореме 3 доказывается следующее утверждение, доказательство которого мы поэтому отпустим.

Теорема 4. Система H, $H \subseteqq \mathfrak{H}$, является $\mathfrak{H}$-замкнутой тогда и только тогда, когда она удовлетворяет H-аксиоме.

Summary

General functional dependencies with a simple solution of completeness problem are discussed. Also the completeness problem in the class of binary decomposition dependencies and in the class of hierarchical decomposition dependencies is solved.

B. THALHEIM
TECHNISCHE UNIVERSITÄT DRESDEN
SEKTION MATHEMATIK
GDR 8027 DRESDEN
MOMMSENSTR. 13

Литература

[1] Armstrong, W. W., Dependency Structures of Data Base Relationships. Proc. IFIP 74, North Holland, 1974, 580—583.

[2] Armstrong, W. W., Delobel, C., Decompositions and Functional Dependencies in Relations. ACM TODS, 5, 4, 1980, 404—430.

[3] Beeri, C., Fagin, R., Howard, J. H., A complete axiomatization for functional and multivalued dependencies in database relations. Proc. 1977 ACM SIGMOD, Toronto, 1977, 47—61.

[4] Beeri, C., Vardi M. Y., On the properties of join dependencies. Advances in Database Theory, Vol. 1, Plenum Press, London, 1982.

[5] Codd, E. F., A Relational Model for Large Shared Data Bases. CACM, 13, 6, 1970, 377—387.

[6] Czedli, G., On Dependencies in the Relational Model of Data. EIK, 17, 1981, 2/3, 103—112.

[7] Date, C. J., An Introduction to Database Systems. Addison-Wesley, Reading, Mass., 1977.

[8] Delobel, C., An overview of the relational data theory. Proc. IFIP 1980, North Holland, 1981, 413—426.

[9] Demetrovics, J., Gyepesi, Gy.: On the functional dependency and some generalizations of it. Acta Cybernetica, Tom 5, Fasc. 3, 1981, 295—305.

[10] Erdős P., Rado R., Intersection theorems for systems of sets. Journal London Math. Soc. 35 (1960), 85—90; 44 (1969), 467—479.

[11] Kambayashi, Y., Database — a bibliography, Vol. 1. Springer-Verlag, 1981.

[12] Makowsky, J. A., Characterizing Data Base Dependencies, LNCS 115, 1981, 86—97.

[13] Ващенко В. П.: Теоретико-множественный подход к функциональной разделимости. Дискретный анализ 10, 1967, 9—22.

[14] Яблонский С. В., Гаврилов Г. П., Кудрявцев В. Б.: Функции алгебры логики и классы Поста. Наука, Москва, 1966.

(Received Febr. 16, 1984)

A queueing model for multiprogrammed computer systems with different I/O times

By J. Sztrik

1. Introduction. Multiprogrammed computer systems are important subject of research in modern queueing theory. We can model such a system as the collection of a Central Procęssor Unit (CPU) or CPU-s, terminals (e. g. rotating disk memory, magnetic tape, card reader, etc.) and the jobs. Each job is associated with a terminal at which it suffers no delay, queues of programs may occur only at the CPU (or CPU-s).

To analyse this kind of systems we often use the finite-source queueing model which is sometimes called the "machine interference model". For a FIFO multiprogrammed computer system a new mathematical model can be given in the following way. Let the number of jobs in the system be n $(n \geqq r)$. Jobs (or programs) emanate from the peripheral devices where various input-output (I/O) operations are carried out. An arriving program is immediately served by one of r CPU-s if there is an idle one, otherwise a waiting line is formed. The jobs are served in the order of their arrival, that is the service discipline is FIFO (first-in, first-out). The service times of the jobs are assumed to be exponentially and identically distributed random variables with mean $1/\mu$. After completing CPU operations the program i returns to its peripheral device and stays there for a random time having an arbitrary distribution function $F_i(x)$ with density $f_i(x)$. All random variables involved here are supposed to be mutually independent of each other.

Since there is a huge literature for the problem in question we refer only to the latest results. Bunday and Scraton [3] have recently proved that the probability distribution of the number of machines running in steady state is the same in the $M/M/r$ and $G/M/r$ cases. In connection with the mathematical description of multiprogrammed computer system for the interested reader the following papers can be recommended: Avi-Itzhak and Heyman [2], Asztalos [1], Csige and Tomkó [5] Gaver [6], Kameda [8], Schatte [10], Sztrik [11]. In Kleinrock's book [9] further models and good bibliography on this subject can be found.

The present paper deals with a possible generalization of the $G/M/r$ case and gives the main steady-state operational characteristics of the system, such as CPU utilization, mean waiting and response times of the jobs.

2. The mathematical model. Let the random variable $\nu(t)$ denote the number of jobs processing I/O operations at time t and $\big(\alpha_1(t), \dots, \alpha_{\nu(t)}(t)\big)$ indicate their indices ordered lexicographically. Let us denote by $\big(\beta_1(t), \dots, \beta_{n-\nu(t)}(t)\big)$ the indices of the jobs waiting or served at the CPU-s in the order of their arrival. Clearly the sets $\{\alpha_1(t), \dots, \alpha_{\nu(t)}(t)\big)$ and $\{\beta_1(t), \dots, \beta_{n-\nu(t)}(t)\}$ are disjoint.

Let us introduce the process

$$\underline{Y}(t) = \big(\nu(t);\ \alpha_1(t), \dots, \alpha_{\nu(t)}(t){:}\ \beta_1(t), \dots, \beta_{n-\nu(t)}(t)\big).$$

The stochastic process $\big(\underline{Y}(t), t \geqq 0\big)$ is not a Markovian one unless the distribution functions $F_i(x)$ are exponential, $i=1, \dots, n$.

Let us introduce the supplementary variable $\xi\alpha_l(t)$ denoting the random time that the job $\alpha_l(t)$ has been staying at a peripheral device in the time period $(0, t)$, $l=1, \dots, \nu(t)$. Define

$$X(t) =. \big(\nu(t);\ \alpha_1(t), \dots, \alpha_{\nu(t)}(t);\ \xi\alpha_j(t), \dots, \xi\alpha_{\nu(t)}(t);\ \beta_1(t), \dots, \beta_{n-\nu(t)}(t)\big).$$

the process $\big(\underline{X}(t), t \geqq 0\big)$ has the Markov property.

Let V_k^n and C_k^n denote the set of all variations and combinations, respectively, of order k of the integers $1, 2, \dots, n$ ordered lexicographically. Then the state space of the process $\big(\underline{X}(t), t \geqq 0\big)$ consists of the points $(i_1, \dots, i_k;\ x_1, \dots, x_k; j_1, \dots, j_{n-k})$, where $(i_1, \dots, i_k) \in C_k^n$, $(j_1, \dots, j_{n-k}) \in V_{n-k}^n$, $x_i \in \mathbf{R}_+$, $i=1, \dots, k$, $k=1, \dots, n$.

The process is in state $(i_1, \dots, i_k;\ x_1, \dots, x_k;\ j_1, \dots, j_{n-k})$ if k jobs with indices $(i_1, \dots, i_k)$ have been processing I/O operations for times $(x_1, \dots, x_k)$, respectively, while the rest of jobs need the CPU-s. The indices of these programs in the order of their arrival are $(j_1, \dots, j_{n-k})$.

To derive the Kolmogorov equations we should consider the transitions that can occur in an arbitrary time interval $(t, t+h)$. The transition probabilities are given in the following way

$$P\{\underline{x}(t+h) = (i_1, \dots, i_k;\ x_1+h, \dots, x_k+h;\ j_1, \dots, j_{n-k})/$$

$$\underline{x}(t) = (i_1, \dots, i_k;\ x_1, \dots, x_k;\ j_1, \dots, j_{n-k})\} =$$

$$\big(1-(n-k)\mu h\big) \prod_{l=1}^{k} \frac{1-F_{i_l}(x_l+h)}{1-F_{i_l}(x_l)} + o(h),$$

$$P\{\underline{x}(t+h) = (i_1, \dots, i_k;\ x_1+h, \dots, x_k+h;\ j_1, \dots, j_{n-k})/$$

$$\underline{x}(t) = (i_1', \dots, j_{n-k}', \dots, i_k';\ x_1', \dots, y', \dots, x_k';\ j_1, \dots, j_{n-k-1})\} =$$

$$\frac{f_{j_{n-k}}(y)h}{1-F_{j_{n-k}}(y)} \prod_{l=1}^{k} \frac{1-F_{i_l}(x_l+h)}{1-F_{i_l}(x_l)} + o(h),$$

$$\text{for} \quad 0 \leqq n-k < r,$$

where $(i_i', \ldots, j_{n-k}', \ldots, i_k')$ denotes the lexicographical order of the indices $(i_1, \ldots, \ldots, i_k, j_{n-k})$, while $(x_1', \ldots, y', \ldots, x_k')$ indicates the corresponding times.

$$P\{\underset{\sim}{x}(t+h) = (i_1, \ldots, i_k;\ x_1+h, \ldots, x_k+h;\ j_1, \ldots, j_{n-k})/$$
$$\underset{\sim}{x}(t) = (i_1, \ldots, i_k;\ x_1, \ldots, x_k;\ j_1, \ldots, j_{n-k})\} =$$
$$(1-r\mu h) \prod_{l=1}^{k} \frac{1-F_{i_l}(x_l+h)}{1-F_{i_l}(x_l)} + \sigma(h),$$

$$P\{\underset{\sim}{x}(t+h) = (i_1, \ldots, i_k;\ x_1+h, \ldots, x_k+h;\ j_1, \ldots, j_{n-k})/$$
$$\underset{\sim}{x}(t) = (i_1', \ldots, j_{n-k}', \ldots, i_k';\ x_1', \ldots, y', \ldots, x_k';\ j_1, \ldots, j_{n-k-1})\} =$$
$$\frac{f_{j_{n-k}}(y)h}{1-F_{j_{n-k}}(y)} \prod_{l=1}^{k} \frac{1-F_{i_l}(x_l+h)}{1-F_{i_l}(x_l)} + o(h).$$
$$\text{for} \quad r \leqq n-k \leqq n.$$

To calculate the distribution of $\underset{\sim}{X}(t)$ consider the following functions.

$$Q_{0;\, j_1, \ldots, j_n}(t) = P(v(t) = 0;\ \beta_1(t) = j_1, \ldots, \beta_k(t) = j_n),$$
$$Q_{i_1, \ldots, i_k;\, j_1, \ldots, j_{n-k}}(x_1, \ldots, x_k;\ t) =$$
$$P(v(t) = k;\ \alpha_1(t) = i_1, \ldots, \alpha_k(t) = i_k;\ \xi_i \leqq x_1, \ldots, \xi_{i_k} \leqq$$
$$\leqq x_k;\ \beta_1(t) = j_1, \ldots, \beta_{n-k}(t) = j_{n-k}) \tag{2.1}$$

Let λ_i be defined by $1/\lambda_i = \int\limits_0^\infty x\,dF_i(x)$, then we have

Theorem 1. If $1/\lambda_i < \infty$, $i=1, \ldots, n$, then the process $(\underset{\sim}{X}(t), t \geqq 0)$ possesses the unique limiting (stationary) ergodic distribution independently of the initial conditions, namely

$$Q_{0;\, j_1, \ldots, j_n} = \lim_{t\to\infty} Q_{0;\, j_1, \ldots, j_n}(t), \tag{2.2}$$

$$Q_{i_1, \ldots, i_k;\, j_1, \ldots, j_{n-k}}(x_1, \ldots, x_k) = \lim_{t\to\infty} Q_{i_1, \ldots, i_k;\, j_1, \ldots, j_{n-k}}(x_1, \ldots, x_k;\ t).$$

Note that $\underset{\sim}{X}(t)$ belongs to the class of piecewise-linear Markov processes subject to discontinuous changes. The proof follows immediately from a theorem on page 211 of Gnedenko—Kovalenko's [7] monograph. Theorem 1 provides the existence and uniqueness of the following limits

$$q_{i_1, \ldots, i_k;\, j_1, \ldots, j_{n-k}}(x_1, \ldots, x_k) =$$
$$\lim_{t\to\infty} P(v(t) = k;\ \alpha_1(t) = i_1, \ldots, \alpha_k(t) = i_k;\ x_l \leqq \xi_{i_l} < x_l+dx_l,\ l =$$
$$= \overline{1, k};\ \beta_1(t) = j_1, \ldots, \beta_{n-k}(t) = j_{n-k}) \tag{2.3}$$
$$\text{for} \quad k = 1, \ldots, n,$$

where $q_{i_1, \ldots, i_k;\, j_1, \ldots, j_{n-k}}(x_1, \ldots, x_k)$ denotes a state probability density associated with state $(i_1, \ldots, i_k;\ x_1, \ldots, x_k;\ j_1, \ldots, j_{n-k})$ as $t\to\infty$, $k=1, \ldots, n$. Note that we have assumed here that the ergodic distributions (2.2) of $\underline{X}(t)$ for fixed k have densities, $k=1, \dots, n$. This assumption is justified if we suppose for simplicity that $F_i(x)$ has density $f_i(x)$, $i=1, \dots, n$. (cf. Gnedenko—Kovalenko [7] pp. 224). We can make this assumption as in many applications we use distributions having density.

In order to formulate the following theorem introduce a further notation, namely

$$q^*_{i_1, \dots, i_k; j_1, \dots, j_{n-k}}(x_1, \dots, x_k) = \frac{q_{i_1, \dots, i_k; j_1, \dots, j_{n-k}}(x_1, \dots, x_k)}{(1-F_{i_1}(x_1)) \dots (1-F_{i_k}(x_k))} \tag{2.4}$$

which is the so-called normed density function, $k=1, \dots, n$. Then we have

Theorem 2. The normed density functions introduced above satisfy the following system of integro-differential equations (2.5,), (2.7) with boundary conditions (2.6), (2.8):

$$\left[\frac{\partial}{\partial x_1}+\dots+\frac{\partial}{\partial x_k}\right]^* q^*_{i_1, \dots, i_k; j_1, \dots, j_{n-k}}(x_1, \dots, x_k) =$$

$$= -(n-k)\mu q^*_{i_1, \dots, i_k; j_1, \dots, j_{n-k}}(x_1, \dots, x_k) + \int\limits_0^\infty q^*_{i'_1, \dots, j'_{n-k}, \dots, i'_k; j_1, \dots, j_{n-k-1}}$$

$$(x'_1, \dots, y', \dots, x'_k) f_{j_n}(y)\,dy \tag{2.5}$$

$$q^*_{i_1, \dots, i_k; j_1, \dots, j_{n-k}}(x_1, \dots, x_{l-1}, 0, x_{l+1}, \dots, x_k) =$$

$$\mu \sum_{V^{i_l}_{j_1, \dots, j_{n-k}}} q^*_{i_1, \dots, i_{l-1}, i_{l+1}, \dots, i_k; j_1, \dots, i_l, \dots, j_{n-k}}(x_1, \dots, x_{l-1}, x_{l+1}, \dots, x_k), \tag{2.6}$$

$$\text{for} \quad l = 1, \dots, k, \quad 0 \leqq n-k < r.$$

$$\left[\frac{\partial}{\partial x_1}+\dots+\frac{\partial}{\partial x_k}\right]^* q^*_{i_1, \dots, i_k; j_1, \dots, j_{n-k}}(x_1, \dots, x_k) =$$

$$= -r\mu q^*_{i_1, \dots, i_k; j_1, \dots, j_{n-k}}(x_1, \dots, x_k) + \int\limits_0^\infty q^*_{i'_1, \dots, j'_{n-k}, \dots i'_k; j_1, \dots, n-k-1}$$

$$(x'_1, \dots, y', \dots, x'_k) f_{j_{n-k}}(y)\,dy, \tag{2.7}$$

$$q^*_{i_1, \dots, i_k; j_1, \dots, j_{n-k}}(x_1, \dots, x_{l-1}, 0, x_{l+1}, \dots, x_k) =$$

$$\mu \sum_{V^{i_l}_{j_1, \dots, j_{r-1}}} q^*_{i_1, \dots, i_{l-1}, i_{l+1}, i_k; j_1, \dots, i_l, \dots, j_{n-k}}(x_1, \dots, x_{l-1}, x_{l+1}, x_k), \tag{2.8}$$

$$\text{for} \quad l = 1, \dots, k, \quad r \leqq n-k \leqq n-1.$$

$$r\mu Q_{0; j_1, \dots, j_n} = \int\limits_0^\infty q^*_{j_n; j_1, \dots, j_{n-1}}(y) f_{j_n}(y)\,dy.$$

Symbol [ ]* will be explained in the proof, while $V^{i_l}_{j_1, \dots, j_s}$ is defined as follows

$$V^{i_l}_{j_1, \dots, j_s} = \{(i_l, j_1, \dots, j_s), (j_1, i_l, j_2, \dots, j_s), \dots, (j_1, \dots, j_s, i_l) \in V^n_{s+1}\}.$$

Proof. Since the process $(\underline{X}(t), t \geqq 0)$ is Markovian its densities must satisfy the Chapman—Kolmogorov equations. A derivation is based on the examination of the sample paths of the process during an infinitesimal interval of width h. The following relations hold:

$$q_{i_1, \dots, i_k; j_1, \dots, j_{n-k}}(x_1+h, \dots, x_k+h) = q_{i_1, \dots, i_k; j_1, \dots, j_{n-k}}(x_1, \dots, x_k) \times$$

$$\times (1-(n-k)\mu h) \prod_{l=1}^{k} \frac{1-F_{i_l}(x_{l+h})}{1-F_{i_l}(x_l)} + \prod_{l=1}^{k} \frac{1-F_{i_l}(x_l+h)}{1-F_{i_l}(x_l)} \times \tag{2.9}$$

$$\times \int_0^\infty q_{i'_1, \dots, j'_{n-k}, \dots, i'_k; j_1, \dots, j_{n-k-1}}(x'_1, \dots, y', \dots, x'_k) \frac{f_{i_{n-k}}(y) h\, dy}{1-F_{i_l}(x_l)} + o(h),$$

$$q_{i_1, \dots, i_k; j_1, \dots, j_{n-k}}(x_1+h, \dots, x_{l-1}+h, x_{l+1}+h, \dots, x_k+h) h = \prod_{\substack{s=1 \\ s \neq l}}^{k} \frac{1-F_{i_s}(x_s+h)}{1-F_{i_s}(x_s)} \times$$

$$\times \mu h \sum_{V^{i_l}_{j_1, \dots, j_{n-k}}} q_{i_1, \dots, i_{l-1} i_{l+1}, \dots, i_k; j_1, \dots, i_l, \dots, j_{n-k}}(x_1, \dots, x_{l-1}, x_{l+1}, \dots, x_k) + o(h)$$

$$\text{for} \quad 0 \leqq n-k < r, \quad l = 1, \dots, k.$$

Similarly

$$q_{i_1, \dots, i_k; j_1, \dots, j_{n-k}}(x_1+h, \dots, x_k+h) = q_{i_1, \dots, i_k; j_1, \dots, j_{n-k}}(x_1, \dots, x_k) \times$$

$$\times (1-r\mu h) \prod_{l=1}^{k} \frac{1-F_{i_l}(x_l+h)}{1-F_{i_l}(x_l)} + \prod_{l=1}^{k} \frac{1-F_{i_l}(x_l+h)}{1-F_{i_l}(x_l)} \times$$

$$\times \int_0^\infty q_{i'_1, \dots, j'_{n-k}, \dots, i'_k; j_1, \dots, j_{n-k-1}}(x'_1, \dots, y', \dots, x'_k) \frac{f_{j_{n-k}}(y) h\, dy}{1-F_{j_{n-k}}(y)} + o(h), \tag{2.10}$$

$$q_{i_1, \dots, i_k; j_1, \dots, j_{n-k}}(x_1+h, \dots, x_{l-1}+h, 0, x_{l+1}+h, \dots, x_k+h) h = \prod_{\substack{s=1 \\ s \neq l}}^{k} \frac{1-F_{i_s}(x_s+h)}{1-F_{i_s}(x_s)} \times$$

$$\times \mu h \sum_{V^{i_l}_{j_1, \dots, j_{r-1}}} q_{i_1, \dots, i_{l-1}, i_{l+1}, \dots, i_k; j_1, \dots, i_l, \dots, j_{n-k}}(x_1, \dots, x_{l-1}, x_{l+1}, \dots, x_k) + o(h)$$

$$\text{for} \quad r \leqq n-k \leqq n-1, \quad l = 1, \dots, k.$$

Finally

$$Q_{0; j_1, \dots, j_n} = Q_{0; j_1, \dots, j_n}(1-r\mu h) + \int_0^\infty q_{j_n; j_1, \dots, j_{n-1}}(y) \frac{f_{j_n}(y) h\, dy}{1-F_{j_n}(y)} + o(h). \tag{2.11}$$

Hence the derivation of eq. (2.5), (2.7) and boundary conditions (2.6), (2.8) is quite simple. Indeed, dividing the lefthand side of eq. (2.9), (2.10), (2.11) by the factor $\prod_{l=1}^{k} (1-F_{i_l}(x_l+h))$, taking into account the definition of the normed densities (2.4) and taking the limits as $h \to 0$ we get the desired result.

In the lefthand side of (2.5), (2.7), used for the notation of the limit in the right-hand side, the usual notation for partial differential quotients has been applied. Strictly speaking this is not allowed since the existence of the individual partial differential quotients is not assured. This is why the operator is notated by []*. Actually this is a $(1, 1, \dots, 1) \in R^k$ directional derivative. (See Cohen [4] pp. 252).

In the following we solve eq. (2.5), (2.7) subject to boundary conditions (2.6), (2.8) to determine the ergodic distribution

$$(Q_{0;\, j_1, \dots, j_n}, Q_{i_1, \dots, i_k;\, j_1, \dots, j_{n-k}}),$$

$$(i_1, \dots, i_k) \in C_k^n, \quad (j_1, \dots, j_{n-k}) \in V_{n-k}^n, \quad k = 1, \dots, n.$$

If we set

$$Q_{0;\, j_1, \dots, j_n} = C_0,$$

$$q^*_{i_1, \dots, i_k;\, j_1, \dots, j_{n-k}}(x_1, \dots, x_k) = c_k, \quad k = 1, \dots, n,$$

then it can be shown by substitution that they satisfy the eq. (2.5), (2.7), and boundary conditions (2.6), (2.8). Moreover the sequence $\{c_k\}$ can be obtained in succession and expressed in a neat form by the help of c_n. Using the relations (2.5), (2.6), (2.7), (2.8) it is easy to see that

$$c_k = (r!\, r^{n-r-k} \mu^{n-k})^{-1} \cdot c_n \quad \text{for} \quad 0 \leqq k \leqq n-r,$$

and, similarly

$$c_k = ((n-k)!\, \mu^{n-k})^{-1} \cdot c_n \quad \text{for} \quad n-r \leqq k \leqq n.$$

Since these equations completly describe the system, this is the required solution.

Let $Q_{i_1, \dots, i_k;\, j_1, \dots, j_{n-k}}$ denote the steady state probability that jobs with indices $(i_1, \dots, i_k)$ are at the peripheral devices and the order of the rest arrival to the CPU-s is $(j_1, \dots, j_{n-k})$. Furthermore, denote by $Q_{i_1, \dots, i_k}$ the steady state probability that programs with indices $(i_1, \dots, i_k)$ are processing I/O operations. It can be verified that

$$Q_{i_1, \dots, i_k;\, j_1, \dots, j_{n-k}} = (\lambda_{i_1} \dots \lambda_{i_k})^{-1} c_k, \quad \text{for} \quad k = 1, \dots, n. \tag{2.12}$$

Using the relations for c_k we get

$$Q_{i_1, \dots, i_k} = (n-k)!\, [r!\, r^{n-k-r} \mu^{n-k} \cdot \lambda_{i_1} \dots \lambda_{i_k}]^{-1} \cdot c_n, \tag{2.13}$$

$$(i_1, \dots, i_k) \in C_k^n, \quad k = 0, 1, \dots, n-r.$$

Similarly

$$Q_{i_1, \dots, i_k} = [\mu^{n-k} \cdot \lambda_{i_1} \dots \lambda_{i_k}]^{-1} \cdot c_n, \tag{2.14}$$

$$(i_1, \dots, i_k) \in C_k^n, \quad k = n-r, \dots, n.$$

Let $\hat{Q}_k$ and $\hat{P}_l$ denote the steady state probabilities that k jobs are staying at the peripheral devices and l jobs need service at the CPU-s, respectively. Clearly

$$Q_{i_1, \dots, i_n} = Q_{1, \dots, n} = \hat{Q}_n = \hat{P}_0, \quad \hat{Q}_k = \hat{P}_{n-k},$$

$$\text{for} \quad k = 0, \dots, n.$$

It is easy to see that

$$C_n = \hat{Q}_n(\lambda_1 \dots \lambda_n),$$

and

$$\hat{Q}_k = \sum_{(i_1, \dots, i_k) \in C_k^n} Q_{i_1, \dots, i_k},$$

where $\hat{Q}_n$ can be obtained with the aid of the norming condition

$$\sum_{k=0}^{n} \hat{Q}_k = 1.$$

In the homogenenous case relations (2.13), (2.14) yield

$$\hat{Q}_k = \frac{n!}{k!\,r!\,r^{n-k-r}} \cdot \left(\frac{\lambda}{\mu}\right)^{n-k} \cdot \hat{Q}_n, \quad \text{for} \quad 0 \leqq k \leqq n-r,$$

$$\hat{Q}_k = \binom{n}{k}\left(\frac{\lambda}{\mu}\right)^{n-k} \cdot \hat{Q}_n, \quad \text{for} \quad n-r \leqq k \leqq n.$$

Thus

$$\hat{P}_k = \binom{n}{k}\left(\frac{\lambda}{\mu}\right)^{k} \cdot \hat{P}_0, \quad \text{for} \quad 0 \leqq k \leqq r, \quad \text{and}$$

$$\hat{P}_k = \frac{n!}{(n-k)!\,r!\,r^{k-r}} \left(\frac{\lambda}{\mu}\right)^{k} \cdot \hat{P}_0, \quad \text{for} \quad r \leqq k \leqq n.$$

This is exactly the same result obtained by Bunday and Scraton [3]. The equivalence of the $E_k/M/1$ and $M/M/1$ models and that of $G/M/r$ and $M/M/r$ models as noted by Benson, Bunday and Scraton, respectively, is just a special case of our more general result obtained here.

Before determining the operational characteristics of the system we need one more theorem. In order to formulate it we introduce some notations. Let $Q^{(i)}(P^{(i)})$ denote the stationary probability that job i is processing I/O operation (need service at the CPU-s) for $i=1, \ldots, n$. It is clear that the process $(\underline{Y}(t), t \geqq 0)$ is semi-Markovian with state space

$$\bigcup_{\substack{(i_1,\ldots,i_k)\in C_k^n,\ (j_1,\ldots,j_{n-k})\in V_{n-k}^n \\ \{i_1,\ldots,i_k\}\cap\{j_1,\ldots,j_{n-k}\}=\emptyset \\ k=0,1,\ldots,n,}} \{(i_1, \ldots, i_k;\ j_1, \ldots, j_{n-k})\}.$$

Let H_i be the event that job i is processing I/0 operation and $Z_{H_i}(t)$ its indicator function i.e.

$$Z_{H_i}(t) = \begin{cases} 1 & \text{if} \quad \underline{Y}(t) \in H_i, \\ 0 & \text{otherwise.} \end{cases}$$

Theorem 4.

$$\lim_{T\to\infty} \frac{1}{T} \int_0^T Z_{H_i}(t)\,dt = \frac{1/\lambda_i}{1/\lambda_i + W_i + 1/\mu} = Q^{(i)} = 1 - P^{(i)},$$

where W_i denotes the mean waiting time of program i.

Proof. The statement is a special case of a theorem concerning the expected sojourn time for semi-Markov processes, see Tomkó [12] pp. 297.

3. The main characteristics of the system

(i) *Utilizations.* Utilizations can now be considered for individual servers or for the system as a whole. The process $\underline{X}(t)$ is assumed to be in equilibrium. Considering the system as a whole it will be empty only when there are no jobs at the CPU-s and will be busy at other times. As usual, using renewal-theoretic arguments for the system utilization we have $U=1-\hat{Q}_n$, and

$$\hat{Q}_n = \frac{M\eta^*}{M\eta^*+M\delta},$$

where $\eta^*=\min(\eta_1, \dots, \eta_n)$, the random variable η_i denotes the I/0 times of program i, $i=1, \dots, n$, and $M\delta$ means the average busy period of the system, respectively. Thus the expected busy period lenght is given by

$$M\delta = M\eta^* \cdot \frac{1-\hat{Q}_n}{\hat{Q}_n}.$$

Specially, if $F_i(x)=1-\exp(-\lambda_i x)$, $i=1, \dots, n$ we get $M\delta=(1-\hat{Q}_n)\left(\hat{Q}_n \cdot \sum_{i=1}^{n} \lambda_i\right)^{-1}$. It is easy to see that for CPU utilization the following relation holds:

$$U_{\text{CPU}} = \frac{1}{r}\left(\sum_{k=1}^{r} k\hat{P}_k + r \sum_{k=r+1}^{n} \hat{P}_k\right) = \frac{\bar{r}}{r},$$

where $\bar{r}$ denotes the mean number of busy CPU-s.

(ii) *Mean waiting times.* By the virtue of Theorem 4 we have

$$Q^{(i)} = (1/\lambda_i)(1/\lambda_i+W_i+1/\mu)^{-1}.$$

Consequently, the expected waiting time of job i is

$$W_i = \frac{1}{\lambda_i}\frac{1-Q^{(i)}}{Q^{(i)}} - \frac{1}{\mu}, \quad \text{for} \quad i = 1, \dots, n.$$

It follows that the mean response time of program i, that is, the waiting and CPU time together, can be obtained by

$$T_i = W_i + 1/\mu = (1-Q^{(i)})(\lambda_i Q^{(i)})^{-1}, \quad \text{for} \quad i = 1, \dots, n. \tag{3.1}$$

Since

$$\sum_{i=1}^{n} (1-Q^{(i)}) = \bar{n}$$

where $\bar{n}$ denotes the mean number of jobs staying at the CPU-s, by reordering and adding (3.1) we have

$$\sum_{i=1}^{n} \lambda_i T_i Q^{(i)} = \bar{n} \tag{3.2}$$

which is Little's formula for the finite-source $\vec{G}/M/r$ queue. In particular, if $F_i(x)=F(x)$, $i=1, \dots, n$, (3.2) can be written as $\lambda T\bar{Q}=\bar{n}$ where $\bar{Q}$ denotes the average number of programs processing I/O operations.

Acknowledgement. I am very grateful to Dr. J. Tomkó for his helpful discussion. My special thanks are also due to Prof. M. Arató for providing valuable comments and very thorough readings of earlier version of the paper which have greatly improved the presentation.

Abstract

The aim of the present paper is to give a new queueing model for a multiprogrammed computer system, where r CPU-s serve the jobs according to the FIFO discipline. The programs are stochastically different, job i is characterised by exponentially distributed CPU time with rate μ and I/O time with an arbitrary distribution function $F_i(x)$ possessing density $f_i(x)$. In steady state we deal with the main performance measures, such as CPU utilization, mean waiting and response times of the jobs.

DEPARTMENT OF MATHEMATICS
UNIVERSITY OF DEBRECEN
PF. 12, DEBRECEN, HUNGARY
H—4010

References

[1] Asztalos, D., Finite source queueing systems and their applications to computer systems, Alk. Mat. Lapok, 5 (1979), 89—101 (in Hungarian).
[2] Avi-Itzhak, B. and D. P. Heyman, Approximate Queuing Models for Multiprogramming Computer Systems, Opns. Res. 21 (1973), 1212—1230.
[3] Bunday, B. D. and R. E. Scraton, The *G/M/r* machine interference model, Eur. J. Operation Res., 4 (1980), 399—402.
[4] Cohen, J. W., The multiple phase service network with generalised processor sharing, Acta Informatica, 12 (1979), 245—284.
[5] Csige, L. and J. Tomkó, The machine interference for exponentially distributed operating and repair times, Alk. Mat. Lapok, 8 (1982), 107—124 (in Hungarian).
[6] Gaver, D. P., Probability models for multiprogramming computer systems, J. ACM, 3 (1967), 423—438.
[7] Gnedenko, B. V. and J. N. Kovalenko, Introduction to Queueing Theory, Nauka, Moscow, 1966 (in Russian).
[8] Kameda, H., A Finite-Source Queue with Different Customers, J. ACM, 29 (1982), 478—491.
[9] Kleinrock, L., Queueing Systems, Vol. 2: Computer Applications, Wiley-Interscience, New York, 1976.
[10] Schatte, P., On the Finite Popoulation *GI/M/1* Queue and its Application to Multiprogrammed Computers, Journal of Information Processing and Cybernetics, 16 (1980), 433—441.
[11] Sztrik, J., Probability model for non-homogeneous multiprogramming computer system, Acta Cybernetica, 6 (1983), 93—101.
[12] Tomkó, J., On sojourn times for semi-Markov processes, 14-th European Meeting of Statisticians, Wroclaw, Poland, 1981.

(Received Oct. 19, 1983)

Characterization of clones acting bicentrally and containing a primitive group

László Szabó

1. Introduction

For a set F of operations on a set A the centralizer F^* of F is the set of operations on A commuting with every member of F. If $F=F^{**}$ then we say that F acts bicentrally. The sets of operations on A acting bicentrally form a complete lattice $\mathcal{L}_A$ with respect to $\subseteqq$.

The clones acting bicentrally were characterized in [7] and [12]. The lattice $\mathcal{L}_A$ was completely described for $|A|=3$ in [3] and [4]. Some further properties of $\mathcal{L}_A$ can be found in [5] and [13].

In this paper for finite sets the clones acting bicentrally and generated by permutations and constant operations are characterized (Theorem 1) and the clones acting bicentrally and containing primitive groups are described (Corollary 2 and Theorem 5). As a corollary a characterization for basic groups is obtained (Corollary 6).

2. Preliminaries

Let A be an at least two element finite set which will be fixed in the sequel. The set of all n-ary operations on A will be denoted by $0_A^{(n)}$ $(n\geqq 1)$. Furthermore, we set $0_A=\bigcup\limits_{n\geqq 1} 0_A^{(n)}$. A set $F\subseteqq 0_A$ is said to be a *clone* if it contains all projections and is closed with respect to superposition of operations. Denote by $[F]$ the clone generated by $F\subseteqq 0_A$. Let f and g be operations on A of arities n and m, respectively. We say that f and g *commute* if for any elements $a_{11}, \dots, a_{mn}\in A$ we have $f(g(a_{11}, \dots, a_{m1}), \dots, \dots, g(a_{1n}, \dots, a_{mn}))=g(f(a_{11}, \dots, a_{1n}), \dots, f(a_{m1}, \dots, a_{mn}))$. It can be easily seen that f and g commute if and only if f is a homomorphism from $\langle A; g\rangle^n$ into $\langle A; g\rangle$ (or g is a homomorphism from $\langle A; f\rangle^m$ into $\langle A; f\rangle$).

By the *centralizer* of a set $F\subseteqq 0_A$ we mean the set $F^*\subseteqq 0_A$ consisting of all operations that commute with every member of F. The set F^{**} is called the *bicentralizer* of F. If $F=F^{**}$ then we say that F *acts bicentrally.*

The set of all projections, the set of all permutations on A, and the set of all unary constant operations will be denoted by P_A, S_A and C_A, respectively.

An operation $f\in O_A$ *depends* on its i-th variable $(1\leqq i\leqq n)$ if there are elements $a_1, \ldots, a_n, a_i'\in A$ such that $f(a_1, \ldots, a_n)\neq f(a_1, \ldots, a_{i-1}, a_i', a_{i+1}, \ldots, a_n)$.

We adapt the terminology of [6] except that polynomials will be called *term functions*. Consequenltly, for an algebra $\mathfrak{A}=\langle A;\ F\rangle$ the set of its term functions and the set of its algebraic functions will be denoted by $T(\mathfrak{A})$ and $A(\mathfrak{A})$, respectively. The algebra $\mathfrak{A}$ is called *complete (functionally complete)* if $T(\mathfrak{A})=O_A$ $(A(\mathfrak{A})=O_A)$. $\mathfrak{A}$ is *trivial* if $T(\mathfrak{A})=P_A$.

Let A be a vector space over a field K. Then the algebra $\langle A; I\rangle$, where I is the set of all idempotent term functions of the vector space A, is said to be an *affine space* over K. By a *linear operation* over the vector space A we mean an operation of the form $\sum_{i=1}^{n} A_i x_i+a$ where $a\in A$ and the A_i are linear transformations of A. It is easy to check that I^* consists of all linear operations over A. If a clone $F\subseteqq O_A$ consists of linear operations over a vector space with base set A, then we say that F is a *linear clone.*

For a natural number n denote by $\underset{\sim}{n}$ the set $\{1, \ldots, n\}$. Let B be a nonempty set and let $m\geqq 2$. An n-ary wreath operation w on the set B^m is associated to transformations p_i of B $(i=1, \ldots, m)$ and maps $r\colon \underset{\sim}{m}\to\underset{\sim}{n}$, $s\colon \underset{\sim}{m}\to\underset{\sim}{m}$ as follows. For $x_i=(x_{i1}, \ldots, x_{im})\in B^m$, $i=1, \ldots, n$, let

$$w(x_1, \ldots, x_n) = \big(p_1(x_{r(1)s(1)}), \ldots, p_m(x_{r(m)s(m)})\big).$$

If $E\subseteqq O_B^{(1)}$ then $W_{E,m}$ denotes the set of all wreath operations on B^m with $p_i\in E$. Now a set of operations $F\subseteqq O_A$ is said to be a *wreath clone* if there is a set B, a natural number $m\geqq 2$ and a transformation monoid $E\subseteqq O_B^{(1)}$ containing id_B such that $\langle A;\ F\rangle\cong\langle B^m;\ W_{E,m}\rangle$.

For a permutation group G acting on A a subset $B\subseteqq A$ is called a *block* of G if for every $g\in G$, either $g(B)=B$ or $g(B)\cap B=\emptyset$. The one-element sets $\{a\}(a\in A)$ and A are called *trivial blocks.* A transitive permutation group G is said to be primitive if it has trivial blocks only.

Following Salomaa [10] a permutation group G on a finite set A is *basic* if the $\langle A;\ G\cup\{f\}\rangle$ is complete for every surjective operation $f\in O_A$ depending on at least two variables.

Two algebras (on a common base set) are *equivalent* if they have the same set of term functions.

We shall use the following result from [8].

Theorem A. A nontrivial finite algebra with primitive automorphism group is functionally complete unless it is equivalent to one of the following algebras:

(i) $\mathfrak{B}^m$, where $m\geqq 2$ and $\mathfrak{B}$ is a functionally complete algebra with primitive automorphism group of composite order,

(ii) an affine space over a finite field,

(iii) $\langle A;\ x+1\rangle$, where $\langle A;\ +\rangle$ is a cyclic group of prime order and $1\in A\setminus\{0\}$,

(iv) $\langle A;\ x-y+z+1\rangle$, where $\langle A;\ +\rangle$ is a cyclic group of prime order and $1\in A\setminus\{0\}$,

(v) $\langle A;\ xy+xz+yz\rangle$, where $\langle A;\ +,\ \cdot\rangle$ is the two element field.

We also need the following characterization of basic groups given in [9] (see also [1, 2]).

Theorem B. A permutation group G on A is basic if and only if G is primitive and G is a subset of no linear clone and no wreath clone.

Results

Let G be a permutation group acting on A, and denote by G_a the stabilizer of the element $a \in A$ (that is $G_a = \{g \in G | g(a) = a\}$). For any $a \in A$ let $\bar{a} = \{x \in A | G_a \subseteqq G_x\}$. The next theorem describes the subclones of $[S_A \cup C_A]$ acting bicentrally.

Theorem 1. Let $F \subseteqq [S_A \cup C_A]$ be a clone and put $G = F \cap S_A$, $C = F \cap C_A$. Then F acts bicentrally if and only if C contains every unary constant operation $A \to \{a\}$ with $\bar{a} = \{a\}$.

Proof. For any $a \in A$ let us denote by c_a the unary constant operation with value a.

Let $F \subseteqq [S_A \cup C_A]$ be a clone. First suppose that F acts bicentrally. Let $f \in F^*$ and choose an element $a \in A$ such that $\bar{a} = \{a\}$. If $g \in G_a$ then $f(a, \ldots, a) = f(g(a), \ldots, g(a)) = g(f(a, \ldots, a))$ showing that $f(a, \ldots, a) \in \bar{a}$ and $f(a, \ldots, a) = a$. It follows that $c_a \in F^{**} = F$.

Now suppose that $c_a \in C$ whenever $\bar{a} = \{a\}$. We have to show that $F = F^{**}$. In [11] it is proved that $[S_A \cup C_A]$ acts bicentrally. Thus we have $F^{**} \subseteqq [S_A \cup C_A]^{**} = [S_A \cup C_A]$. Therefore it is enough to show that $F^{**} \cap (S_A \cup C_A) = G \cup C$.

Let $A = \{a_1, \ldots, a_n\}$ and define an n-ary operation f as follows:

$$f(x_1, \ldots, x_n) = \begin{cases} x_1 & \text{if } (x_1, \ldots, x_n) = \big(g(a_1), \ldots, g(a_n)\big) \text{ for some } g \in G, \\ x_2 & \text{otherwise.} \end{cases}$$

Now $f \in (G \cup C)^* = F^*$ and if $g \in S_A \setminus G$ then $g \notin \{f\}^*$. It follows that $F^{**} \cap S_A = G$.

For any $a \in A$ with $c_a \notin C$ and for any $u \in \bar{a}$ define a unary operation $h_{a,u}$ as follows:

$$h_{a,u}(x) = \begin{cases} g(u) & \text{if } x = g(a) \text{ for some } g \in G, \\ x & \text{otherwise.} \end{cases}$$

Remark that h is well-defined. Indeed, if $g_1(a) = g_2(a)$ for some $g_1, g_2 \in G$ then $g_1^{-1} \circ g_2 \in G_a$ which implies $g_1^{-1} \circ g_2 \in G_u$ and $g_1(u) = g_2(u)$.

We show that $h_{a,u} \in F^* = (G \cup C)^*$. Let $t \in G$ and $x \in A$. If $x = g(a)$ for some $g \in G$ then $t(h_{a,u}(x)) = t(h_{a,u}(g(a))) = t(g(u)) = h_{a,u}(t(g(a))) = h_{a,u}(t(x))$. If $x \notin \{g(a) | g \in G\}$ then $t(x) \notin \{g(a) | g \in G\}$, and therefore $t(h_{a,u}(x)) = t(x) = h_{a,u}(t(x))$. Hence $h_{a,u}$ commutes with t. Now let $c_b \in C$. Then $b \notin \{g(a) | g \in G\}$ since $b = g(a)$ implies $c_a = g^{-1} \circ c_b \in F \cap C_A = C$. Therefore for any $x \in A$ we have $c_b(h_{a,u}(x)) = b = h_{a,u}(b) = h_{a,u}(c_b(x))$. Hence $h_{a,u} \in F^*$.

Finally let $c_a \in C_A \setminus C$. By assumption there is an element $u \in \bar{a}$ with $u \neq a$. Now $h_{a,u} \in F^*$ and $c_a(h_{a,u}(u)) = a \neq u = h_{a,u}(a) = h_{a,u}(c_a(u))$ showing that $h_{a,u}$ and c_a do not commute. Thus we have $F^{**} \cap C_A = C$ and $F^{**} \cap (S_A \cup C_A) = (F^{**} \cap S_A) \cup (F^* \cap C_A) = G \cup C$, which completes the proof. □

Corollary 2. Let $F\subseteqq[S_A\cup C_A]$ be a clone such that $G=F\cap S_A$ is a primitive permutation group. Then F acts bicentrally if and only if either $F\cap C_A=C_A$, or $F\cap C_A=\emptyset$ and G is a regular group of prime order.

Proof. Since G is transitive and F is a clone, we have either $F\cap C_A=C_A$ or $F\cap C_A=\emptyset$.

Suppose that $F\cap C_A$ contains every unary constant operation c_a with $\bar{a}=\{a\}$. If $F\cap C_A=\emptyset$ then $\bar{a}\neq\{a\}$ for any $a\in A$. Let $a\in A$ and $x\in\bar{a}$ with $x\neq a$. Then $G_a\subseteqq G_x$. Since the stabilizer subgroups of two distinct elements cannot coincide in a primitive group of composite order (see [14; Prop. 8.6]), it follows that G has prime order. Hence G is a regular group of prime order. In this case $\bar{a}=A$ for any $a\in A$. Finally apply Theorem 1. □

Next we prove two lemmas.

Lemma 3. If $\mathfrak{A}=\langle A;\ F\rangle$ is a functionally complete algebra then $F^*\subseteqq[S_A\cup C_A]$ and consequently $F^*=[\mathrm{End}\,\mathfrak{A}]$.

Proof. Using the functional completeness of $\mathfrak{A}$, it is not hard to show that $(*)$ if θ is a congruence on $\mathfrak{A}^n$ $(n\geqq 1)$ then there exist $i_1, \dots, i_k$ $(k\geqq 0,\ 1\geqq i_1<\dots<i_k\leqq n)$ such that

$$\theta = \{((a_1, \dots, a_n), (b_1, \dots, b_n))\in(A^n)^2\colon\ a_{i_1} = b_{i_1}, \dots, a_{i_k} = b_{i_k}\}.$$

Clearly, then θ has $|A|^k$ classes.

Now let $f\in F^*$ be an n-ary operation $(n\geqq 1)$. Then f is a homomorphism from $\mathfrak{A}^n$ to $\mathfrak{A}$ and therefore $\ker f$ is a congruence on $\mathfrak{A}^n$. If $\ker f=(A^n)^2$ then f is constant and therefore $f\in[C_A]\subseteqq[S_A\cup C_A]$. If $\ker f\neq(A^n)^2$ then (since $\ker f$ has at most $|A|$ classes), by $(*)$, there exists an i $(1\leqq i\leqq n)$ such that

$$\ker f=\{((a_1, \dots, a_n), (b_1, \dots, b_n))\in(A^n)^2\colon\ a_i=b_i\}.$$

From this it follows that f depends only on its i-th variable, and $f\in[S_A]\subseteqq[S_A\cup C_A]$. □

Lemma 4. Let $\mathfrak{A}=\langle A;\ F\rangle$ be a functionally complete algebra and let $m\geqq 2$. Then the centralizer of F on A^m coincides with $W_{E,m}$ where $E=\mathrm{End}\,\mathfrak{A}$.

Proof. Let f be an n-ary operation on A^m that is $f\colon (A^m)^n\to A$. Let $f_1, \dots, f_m$ be the $m\cdot n$-ary operations on A defined as follows: For any $(a_{i1}, \dots, a_{im})\in A^m$, $i=1, \dots, n$, let $f((a_{11}, \dots, a_{1m}), \dots, (a_{n1}, \dots, a_{nm}))=(f_1(a_{11}, \dots, a_{nm}), \dots, f_m(a_{11}, \dots, a_{nm}))$. Observe that f is a wreath operation if and only if each f_j depends on at most one variable, $j=1, \dots, m$.

Now $f\in F^*$ on A^m if and onyl if f is a homomorphism from $(\mathfrak{A}^m)^n$ into $\mathfrak{A}^m$ which is equivalent to that $f_1, \dots, f_m$ are homomorphisms from $(\mathfrak{A}^m)^n$ into $\mathfrak{A}$. The latter means exactly that $f_1, \dots, f_m\in F^*$ on A. Taking into consideration Lemma 3, this is equivalent to that $f_1, \dots, f_m\in[\mathrm{End}\,\mathfrak{A}]$. This completes the proof. □

The following theorem completely describes those clones acting bicentrally and containing a primitive permutation group which are not contained in $[S_A\cup C_A]$.

Theorem 5. Let $F\subseteqq O_A$ be aclone acting bicentrally and containing a primitive permutation group such that $F\nsubseteq[S_A\cup C_A]$. Then we have for F one of the following five possibilities:

(1) $\langle A;\ F\rangle \cong \langle B^m;\ W_{E,m}\rangle$ where B is a finite set, $m \geqq 2$, and $E = G \cup C_B$ with $G \subseteqq S_B$ a primitive group of composite order,

(2) F is the set of all linear operations over a vector space with base set A,

(3) $F = \{x+1\}^*$ where $\langle A;\ +\rangle$ is a cyclic group of prime order and $1 \in A \setminus \{0\}$,

(4) $F = [\{x-y+z+1\}]$ where $\langle A;\ +\rangle$ is a cyclic group of prime order and $1 \in A \setminus \{0\}$,

(5) $F = 0_A$.

Proof. Let F satisfy the hypothesis of the theorem. If $F^* = P_A$ then $F = F^{**} = P^* = 0_A$ that is we have case (5). Let us suppose that $F^* \neq P_A$. Then $\langle A;\ F^*\rangle$ is a nontrivial algebra with primitive automorphism group. If $\langle A;\ F^*\rangle$ is functionally complete then, according to Lemma 3, we have $F = F^{**} \subseteqq [S_A \cup C_A]$ contrary to our assumption on F. Hence $\langle A;\ F^*\rangle$ is not functionally complete. Therefore, taking into consideration Theorem A, we get for the algebra $\langle A;\ F^*\rangle$ one of the cases (i)—(v).

If $\langle A;\ F^*\rangle \cong \langle B^m;\ H\rangle$ where $m \geqq 2$ and $\mathfrak{B} = \langle B;\ H\rangle$ is a functionally complete algebra with primitive automorphism group of composite order, then Lemma 4 shows that $\langle A;\ F\rangle = \langle A;\ F^{**}\rangle \cong \langle B^m;\ W_{E,m}\rangle$ where $E = \operatorname{End} \mathfrak{B}$. In [8; Lemma 1] it was proved that a finite algebra with primitive automorphism group of composite order has idempotent term functions only. Therefore all unary constant operations on B belong to $\operatorname{End} \mathfrak{B}$. Hence $E = \operatorname{End} \mathfrak{B} = \operatorname{Aut} \mathfrak{B} \cup C_B$ and we have (1).

In case (ii) and (iii) F enjoys property (2) and (3), respectively. In case (iv) it is easy to show that $F = F^{**} = \{x-y+z+1\}^* = [\{x-y+z+1\}]$, that is we have (4).

Finally for $\langle A;\ F^*\rangle$ the case (v) cannot occur. Indeed, it can be shown easily that if $\langle A;\ +,\ \cdot\rangle$ is the two element field then $\{xy+xz+yz\}^* = [S_A \cup C_A]$ and $F = F^{**} = \{xy+xz+yz\}^* = [S_A \cup C_A]$ contradicts our assumption on F. □

Combining Theorem 5 with Theorem B we get the following characterization for basic groups.

Corollary 6. A permutation group G acting on A ($|A| \geqq 3$) is basic if and only if $(G \cup f)^{**} = 0_A$ for any operation $f \in 0_A \setminus [S_A \cup C_A]$.

Proof. If G is primitive then our statement follows immediately from Theorem 5 and Theorem B. If G is imprimitive then, by Theorem B, G is not basic. Let B be a nontrivial block of G and choose an element $a \in B$. Define a unary operation f and a binary operation g as follows:

$$f(x) = \begin{cases} a & \text{if } x \in B, \\ x & \text{otherwise,} \end{cases}$$

$$g(x, y) = \begin{cases} x & \text{if } (x, y) = (t(u),\ t(v)) \text{ for some } t \in G \text{ and } u, v \in B, \\ y & \text{otherwise.} \end{cases}$$

Then it is easy to check that $G \cup \{f\} \subseteqq \{g\}^*$. Furthermore, $\{g\}^* \neq 0_A$ since if $u, v \in B$, $u \neq v$, and $h \in 0_A^{(1)}$ is such that $h(u) \in B$ and $h(v) \in A \setminus B$ then $h \notin \{g\}^*$. Therefore $(G \cup \{f\})^{**} \subseteqq \{g\}^{***} = \{g\}^* \neq 0_A$, which completes the proof. □

References

[1] Baĭramov, R. A., Fundamentality criteria for permutation groups and transformation semigroups, Dokl. Akad. Nauk SSSR, 9 (1969), 455—457 (Russian).

[2] Baĭramov, R. A., Derivation of basicity criteria for permutation groups and transformation semigroups from a theorem of Rosenberg, Kibernetika (Kiev).

[3] Danil'cenko, A. F., On the parametrical expressibility of functions of three-valued logic, Algebra i logika, 16/4 (1977), 379—497 (Russian).

[4] Danil'cenko, A. F., Parametrically closed classes of functions of three-valued logic, Izv. Akad. Nauk Moldav. SSR, 1978, no. 2, 13—20 (Russian).

[5] Danil'cenko, A. F., On parametrical expressibility of the functions of k-valued logic, in: Finite Algebra and Multiple-valued Logic (Proc. Conf. Szeged, 1979), Colloq. Math. Soc. J. Bolyai, vol. 28, North-Holland (Amsterdam, 1981), 147—159.

[6] Grätzer, G., Universal algebra, 2nd ed., Birkhauser Verlag, 1979.

[7] Kuznecov, A. V., On detecting non-deducibility and non-expressibility, in: Logical deduction, Nauka, Moscow (1979) 5—33 (Russian).

[8] Pálfy, P. P., L. Szabó and Á. Szendrei, Automorphism groups and functional completeness, Algebra Universalis, 15 (1982), 395—400.

[9] Rosenberg, I. G., On closed classes, basic sets and groups, Preprint, CRMA Université de Montréal.

[10] Salomaa, A. A., On basic groups for the set of functions over a finite domain, Proc. Camb. Phil. Soc., 62 (1966), 597—611.

[11] Szabó, L., Endomorphism monoids and clones, Acta Math. Acad. Sci. Hung., 26 (1975), 279—280.

[12] Szabó, L., Concrete representation of related structures of universal algebras I., Acta Sci. Math., 40 (1978), 175—184.

[13] Szabó, L., On the lattice of clones acting bicentrally, Acta Cybernetica, v. 6 (1984), 381—387.

[14] Wielandt, H., Finite permutation groups, Academic Press (New York and London, 1964).

(Received Aug 30, 1983)

New books

G. Enderle, K. Kansy, G. Pfaff: Computer Graphics Programming, XVI+542 pages, Springer Verlag Berlin—Heidelberg (1984).

The subject of the book — the Graphical Kernel System (GKS) — is a consistent and complete description of a graphics standard system, consisting of a concise terminology for Computer Graphics (CG) and establishing the firm base of a methodology for CG too.

The book is divided into four main parts:

Part I gives an introduction to basic concepts of CG and to principles and concepts of GKS. Main areas of CG, different types of CG users, the interfaces of GKS and the underlying design concepts for the GKS development are explained. This part gives an informal introduction to the main concepts of GKS and their interrelation: output, attributes, coordinate systems, transformations, input, segments, metafile, state lists and error handling.

Part II describes the process of GKS design. Concepts and functionality, laid down in existing graphics systems and in literature were examined carefully to select those which would be included in the standard kernel system. This part gives very useful background information to understand why some features are included in GKS, why other useful concepts and features have been omitted, and why some features are included in a particular form.

Part III gives a detailed description of the functional capabilities of GKS. For every one of the different areas of the system, a detailed explanation of the concept is given. For all functions, both the language-independent definition from the GKS standard and the FORTRAN interface are given. Examples are used to clarify the GKS functions, they are presented both in Pascal and FORTRAN.

Part IV is devoted to the elaboration of the various interfaces of GKS within the CG environment.

The book is aimed, at one hand, at grapics users and experts who want to get an overview of the new standard and a better understanding of its concepts.

On the other hand, it adresses the graphics programmers who want to use GKS for realizing their graphical applications. It can serve as the base for teaching and studying functions, concepts and methods of GKS. It is a very useful book for a lot of people.

J. Csirik (Szeged)

Martin D. Davis and Elaine J. Weyuker: Computability Complexity, and Languages: Fundamentals of Theoretical Computer Science (Computer Science and Applied Mathematics), XIX+425 pages, Academic Press, 1983.

This well-written book provides an introduction to various, mostly classic topics of computation theory. Much of the material included goes well back in the history.

The book consists of five parts. Part 1 comprises an extensive study of various equivalent models of computation. Simple programming languages, Turing machines, semi-Thue processes, gramars, primitive recursive functions and minimalization are treated, and basic theorems of recursion theory are developed. Part 2 exhibits some aspects of formal languages and automata theory. Regular languages and finite automata, context-free languages and pushdown automata, context-sensitive languages and linear bounded automata are dealt with. Normal forms, closure properties, decidability results are presented in a way the reader can easily follow and attain. Propositional calculus, quantification theory, and especially, the resolution principle form the subject of Part 3, which culminates in Gödel's Incompleteness Theorem and the unsolvability of the first order logic. Three aspects of complexity theory give the material of Part 4. These are the infinite hierarchy of recursive functions computable by nested loop programs, axiomatic complexity theory, and polynomial time computability. The Blum axioms, Gap Theorem, Speedup Theorem, and Cook's Theorem are included. The book ends with Part 5, an insight into the more advanced recursion theory. The significance of oracle computations, various reducibility concepts and unsolvability degrees are explained.

The author's aim was to give a selfcontained, rigorous and unified introduction to selected topics of theoretical computer science. Certainly, they have done an excellent job. The material is well-arranged, the proofs are clear. A dependency graph and a subject index help guide the reader, who can be either a graduate or undergraduate student. Or even, one possessing with a basic knowledge of the material may find crucial points what the authors could show from a new point of view.

Z. Ésik (Szeged)

Data Base File Organization Ed. by S. P. Ghosh, Y. Kombayashi and W. Lipski, X+352 pages, Academic Press, 1983.

This book contains articles presented at an international conference organized by the Polish Academy of Sciences in Warsaw (1981). The topics are devoted to the so-called „consecutive retrieval property" of records in definite and finite query-answer data base systems.

Let be given a finite set X of records and a family $\mathfrak{M}$ of subsets of X. (Practically $\mathfrak{M}$ may be viewed as the collection of all possible answers for a definite query set Q.) We say that $\mathfrak{M}$ has the consecutive retrieval property, if there is a linear ordering on X by which every $M\in\mathfrak{M}$ is an interval. It is clear that this property is very useful with respect to the file organization in a time-independent data base system.

S. P. Ghosh and W. Lipski, Jr. give very informative surveys on the problems and results of the subject. Unfortunately most of the interesting algorithmic problems turned out to be NP-complete. Most of the papers deal with the topic from a mathematical point of view. The basic property, its generalizations and the similar notions can be formulated in different disciplines, i.e. in graph and set theory. Authors have results concerning the decomposition of query sets, equivalences of different formulations, combinatorial problems e.t.c. As to applications, we find articles on the connection of the mathematical properties of a query set and the structure of physical devices as well as on existing systems based on that theory.

W. Lipski, Jr. presents a large bibliography related to the subject.

Á. Makai (Szeged)

Robert R. Korfhage, Discrete Computational Structures (Computer Science and Applied Mathematics), XIII+360 pages, Academic Press, 1984.

This is the second edition of the book.

"The organization of this second edition has grown out of experience in using the first edition in classes. Some of the special topics in the first edition have been replaced by other more current topics, but the content is largely the same. There is an increased emphasis on numerical work, especially in the separate chapter on arrays and matrices and in the placement of the combinatorics chapter toward the front of the book. The close relationship between graph theory and combinatorics is indicated by the positioning of the combinatorics chapter after an introduction to graph theory and before more advanced graph topics are covered.

The other major change for this edition is the replacement of much of the algebraic work by a chapter focusing on finite automata as a basis for formal languages. In recent years these automata have begun to play an increasing role in practical computing because of their direct applicability to many string processing problems, their easy implementation in integrated circuit technology, and their use in robotics."

The book is written in a nice style, it is easy to read. Each section contains some exercises.

F. Gécseg (Szeged)

L. Uhr: Algorithm-Structured Computer Arrays and Networks, XXIII+413 pages, Academic Press, 1984

This book is useful as a textbook on computer architecture, parallel computers and networks as well as on designing and constracting parallel algorithms for a variety of different processor arrays. Author has a perspective, that the construction of the algorithm, program, language, operating system, and of the physical structure of the processors should attempt to maximize correspondence. He presents informative surveys and to a certain extent details too on:

built or designed arrays and networks of processors;

the problems of developing algorithms and programs using these architectures;

graph theoretical methods suggesting interconnection patterns for processor networks (trees, augmented trees, pyramids, lattices, n-cubes, e.t.c.);

modeling techniques for evaluating architectures, programs and the way programs are mapped onto hardware;

languages embedded in PASCAL for parallel processing;

future network architectures and technologies.

Discussion is more practical than theoretical, references are given to mathematical backgrounds. The very large up-to-date bibliography may be used as a basis of researches on special subfields.

Á. Makay (Szeged)

A SZERKESZTŐ BIZOTTSÁG CÍME:

6720 SZEGED
SOMOGYI U. 7.

EDITORIAL OFFICE:

6720 SZEGED
SOMOGYI U. 7.
HUNGARY

INDEX — TARTALOM

ISSN 0324—721.X

Felelős szerkesztő és kiadó: Gécseg Ferenc
A kézirat a nyomdába érkezett: 1984. április 26
Megjelenés: 1984. december
Terjedelem: 12,95 (A/5) ív
Készült monószedéssel, íves magasnyomással
az MSZ 5601 és az MSZ 5602—55 szabvány szerint
84-1980 — Szegedi Nyomda — F.v.: Dobó József igazgató