Tomus 7. Fasciculus 4.

ACTA CYBERNETICA

FORUM CENTRALE PUBLICATIONUM
CYBERNETICARUM HUNGARICUM

Szeged, 1986
Curat: Universitas Szegediensis de Attila József nominata

7. kötet 4. füzet

ACTA CYBERNETICA

A HAZAI KIBERNETIKAI KUTATÁSOK
KÖZPONTI PUBLIKÁCIÓS FÓRUMA

Szeged, 1986. augusztus, szeptember
A Szegedi József Attila Tudományegyetem gondozásában

Tomus 7.

ACTA CYBERNETICA

FORUM CENTRALE PUBLICATIONUM
CYBERNETICARUM HUNGARICUM

Szeged, 1986
Curat: Universitas Szegediensis de Attila József nominata

INDEX

Tomus 7.

86-286 — Szegedi Nyomda — F. v.: Surányi Tibor igazgató

The alternation number and a dot hierarchy of regular sets

STEPHEN L. BLOOM

In this note we introduce a property of languages we call the alternation number. This property is used to deduce several facts about regular languages in particular. One of these facts is related to what might be called the "*generalized dot height*" of a regular language. The complexity of regular expressions is usually determined by their "*star height*" [E], although other complexity measures have also been considered [EK]. In all of the papers we know of, a nontrivial argument is needed to establish that the complexity of '*regular expressions*' must grow in order to name all regular sets. (Our definition of '*regular expression*' allows an atomic name for each finite set, see below.) In the present note, we give a simple proof that the "*dot height*" (or depth of concatenation signs) of a regular expression also must grow. (The "*dot-depth*" considered in [BK] is unrelated to our dot height.) The dot height is related to the alternation number. We will show that if the alternation number of a regular language L is n, then any regular expression which denotes L contains (roughly) at least $\log n$ dots.

Define a function f from the set of nonnegative integers $\mathbf{N}$ to collections of regular subsets of Σ^* (for some fixed alphabet Σ) as follows:

$$f(0) = \text{all finite languages};$$

$$f(n+1) = f(n) \cup \{L\colon L = L_1 + L_2, L = L_1 \cdot L_2, \quad L = L_1^*, L_i \in f(n)\}.$$

Thus a language is regular iff it belongs to $f(n)$ for some n. We will show first that for each n, there is a language in $f(n+1) - f(n)$. The truth of this fact follows easily from the well-known star-height hierarchy theorem (see [DS] for one proof). The argument given here shows that the hieararchy of regular languages depends also on the operation of concatenation. As a corollary we will obtain the dot height hierarchy. Lastly, we mention an automaton characterization of the alternation number.

First we assume that Σ has at least two letters, say a and b. A language L *admits alternations of size* n if: for each $k > 0$ (or, equivalently, for infinitely many $k > 0$) there is a word w in L of the form

$$w(k) = u_0 x_0 \dots x_0 u_1 x_1 \dots x_1 u_2 \dots u_n x_n \dots x_n u_{n+1}$$

where

1. $u_0, u_1, \ldots, u_{n+1}$ are arbitrary words in Σ^*;
2. x_i are letters in Σ, and for each $i<n$, x_i and x_{i+1} are distinct;
3. there are k consecutive occurences of the letters x_i, $i=0, 1, \ldots, n$.

(We will say that a word of the form $w(k)$ has "n alternations of length at least k".) We say a language admits alternations of size 0 if it admits no alternations. For example, finite languages admit no alternations. $\{a\}^*$, $\{b\}^*$ admits alternations of size 1, as does $L\cdot\{a\}^*\cdot L'\cdot\{b\}^*\cdot L''$, for any nonempty languages L, L' and L''. Note that if L admits alternations of length $n+1$, then L also admits alternations of length n.

Definition. The alternation number of L, $a(L)$, is n if L admits alternations of size n but not of size $n+1$. Let $a(L)=\infty$ if for each n, L admits alternations of size n.

Proposition 1. Assume $a(L)=x$, and $a(K)=y$, where $x, y\in \mathbf{N}\cup\{\infty\}$. Then

$$a(L+K) = \max(x, y);\ x+y \leqq a(L\cdot K);\quad \text{if}\quad a(L^*) > 0, \tag{1.1}$$

$$\text{then}\quad a(L^*) = \infty.$$

$$a(L\cdot K) \leqq x+y+1; \tag{1.2}$$

(Of course, $n<\infty$ and $n+\infty=\infty$, for all $n\in\mathbf{N}$.)

Proof. We prove only the last part of 1.1. If $a(L^*)>0$, for each k, there is a word w in L^* which has 1 alternation of length at least k. But then ww has at least 3 alternations of length at least k, and www has at least 5 alternations of length at least k, etc. Thus $a(L^*)=\infty$.

The proof of 1.2 is longer. Assume that $L\cdot K$ admits alternations of size s. We will show $s\leqq a(K)+a(L)+1$. For each $k>0$ there is a word in $L\cdot K$ of the form

$$w(k) = u_0 x_0 \ldots x_0 u_1 x_1 \ldots x_1 u_2 \ldots u_s x_s \ldots x_s u_{s+1}$$

as described above. We may factor each word $w(k)$ as $l(k)\cdot v(k)$, with $l(k)\in L$ and $v(k)\in K$. Suppose that $i(k)$ is the greatest integer i, $-1\leqq i\leqq s$, such that

$$u_0 x_0^k \ldots u_i x_i^k$$

is an initial segment of $l(k)$ ($i(k)=-1$ if there is no such initial segment). Thus at least one of the integers between -1 and s is the value of $i(k)$ for infinitely many k; let n be the maximum of these integers, so that for infinitely many values of k, $i(k)=n$. (If $n=-1$ or $n=s$, then we may easily show that $s-1\leqq a(K)$ and $s\leqq a(L)$, respectively, so that from now on, we assume $0\leqq n<s$.)

For infinitely many values of k (say $k\in I$) we may write

$$l(k) = u_0 x_0^k \ldots u_n x_n^k l'(k),$$

$$v(k) = l''(k) u_{n+2} x_{n+2}^k \ldots x_s^k u_{s+1},$$

where $l'(k)l''(k)=u_{n+1}x_{n+1}^k$.

Case 1. For infinitely many values of k, say k in I', $l'(k)$ is an initial segment of u_{n+1}. Then, for k in I', we may write

$$v(k) = v'(k)\, x_{n+1}^k u_{n+2} x_{n+2}^k \dots x_s^k u_{s+1},$$

so that $s-(n+1) \leqq a(K)$; also, $n \leqq a(L)$, so that $s-i \leqq a(K)+a(L)$, as claimed.

Case 2. Otherwise. Then, for infinitely many k, u_{n+1} is an initial segment of $l'(k)$. Hence, for these values of k there is a number $h(k)$ for which

$$l(k) = u_0 x_0^k \dots x_n^k u_{n+1} x_{n+1}^{k-h(k)};$$

$$v(k) = x_{n+1}^{h(k)} u_{n+2} \dots u_s x_s^k u_{s+1}.$$

We now have two further subcases.

Case 2a. The numbers $h(k)$ are unbounded. Then, we may write

$$v(k) = x_{n+1}^{h(k)} u'_{n+2} x_{n+2}^{h(k)} \dots u'_s x_s^{h(k)} u'_{s+1},$$

for infinitely many k, which shows that $s-(n+1) \leqq a(K)$; clearly, $n \leqq a(L)$, so that again, $s-1 \leqq a(L)+a(K)$.

Case 2b. Otherwise. In this case, the numbers $k-h(k)$ are unbounded, so that for infinitely many k,

$$l(k) = u'_0 x_0^{k-h(k)} u'_1 \dots x_n^{k-h(k)} u'_{n+1} x_{n+1}^{k-h(k)}.$$

Since the numbers $k-h(k)$ are unbounded, $n+1 \leqq a(L)$; clearly, $s-(n+2) \leqq a(K)$, so that $s-1 \leqq a(L)+a(K)$, completing the proof.

Lemma 2. For each $n \in \mathbf{N}$ define

$$g(n) = \max \{a(L): L \in f(n) \quad \text{and} \quad a(L) < \infty\}.$$

Then $g(0)=g(1)=0$; $g(2)=1$ and for $n>0$, $g(n)=2^{(n-1)}-1$.

Proof. All languages in $f(0)$ are finite, so that $g(0)=0$; the sum and product of finite languages are finite, and $a(L^*)$ is either 0 or ∞, so that $g(1)=0$ also. Clearly $g(2)$ is at least 1 and by Proposition 1, $g(2)$ is at most 1. Assume $g(n)=2^{n-1}-1$. If $L \in f(n+1)$ and $a(L)<\infty$, then using the proposition, the largest $a(L)$ can be is $2g(n)+1=2[2^{n-1}-1]+1=2^n-1$. But it is easy to see that $g(n+1)$ is not less than $2g(n)+1$ also, completing the induction.

Theorem 3. For each positive $n \in \mathbf{N}$, there is a language in $f(n)-f(n-1)$.

Proof. Let L be a language in $f(n)$ with $a(L)=g(n)$. Then, if $n>1$, L is not in $f(n-1)$, since $g(n-1)<g(n)$. The statement is trivial for $n=1$.

What about the case that Σ is a singleton, say $\{a\}$, so that Σ^* may be identified with $\mathbf{N}$? In this case, if L is an ifinite regular set, there is a finite set F and a fixed integer n and numbers $k1, \dots, kt$ such that

$$L = F \cup \{a^{k1}\} \cdot \{a^n\}^* \cup \{a^{k2}\} \cdot \{a^n\}^* \cup \dots \cup \{a^{kt}\} \cdot \{a^n\}^* = F \cup \{a^{k1}, \dots, a^{kt}\} \cdot \{a^n\}^*.$$

Hence all regular subsets of $\mathbf{N}$ are in $f(3)$.

In order to avoid the trivial cases, we assume that the regular expressions (over Σ) are built from the atomic letters (i.e. a symbol for each *finite subset of* Σ^*) and the function symbols $+$, $\cdot$, and $*$ in the usual way. (Thus, a finite set of words may be denoted by a regular expression with none of the function signs $+$, $\cdot$, $*$.) Let $|\alpha|$ be the language denoted by the regular expression α. If α is a regular expression, let the '*dot height of* α', $\mathrm{dh}(\alpha)$, be 0, when α is an atomic letter; $\mathrm{dh}(\alpha+\beta)=\max(\mathrm{dh}(\alpha), \mathrm{dh}(\beta))$; $\mathrm{dh}(\alpha^*)=\mathrm{dh}(\alpha)$, and lastly,

$$\mathrm{dh}(\alpha\cdot\beta) = 1+\max(\mathrm{dh}(\alpha), \mathrm{dh}(\beta)).$$

Let $R(n)$ denote the family of regular expressions α with $\mathrm{dh}(\alpha)<n$.

Proposition 4. Suppose that $n>0$, that L is a language denoted by α, $\alpha\in R(n)$ and that $a(L)<\infty$. Then

$$a(L) \leqq g(n).$$

Proof. By induction on n. If $n=1$ and L is denoted by a regular expression α having no dot symbols, then either α is an atomic symbol or has the form

$$\beta+\sigma, \quad \text{or} \quad \beta^*$$

for some other regular expressions β, σ with dot height 0. By induction on the structure of α, one sees that either $a(L)=\infty$ or $a(L)=0=g(1)$.

Now assume that the proposition holds for n and that L is a language denoted by a regular expression in $R(n+1)-R(n)$ and $a(L)<\infty$. If α is of the form $\beta+\sigma$ or β^*, it is easily seen by induction on the structure of α that $a(L)\leqq g(n+1)$. If α is of the form $\beta\cdot\sigma$ then $\mathrm{dh}(\beta), \mathrm{dh}(\sigma)<n$. Thus, by the induction hypothesis, $a(|\beta|)$ and $a(|\sigma|)$ are at most $g(n)$, and by proposition 1, $a(L)$ is at most $2g(n)+1=g(n+1)$, completing the proof.

Corollary 5. (The '*dot height*' hierarchy). For each $n>0$, there is an infinite regular language L not denoted by a regular expression in $R(n)$.

Proof. Any regular language L with $g(n)<a(L)<\infty$ will do, by Proposition 2.

The alternation number of a regular language may be described by certain properties of a finite automaton which accepts it. Let $\mathbf{M}=(Q, i, F)$ be a finite Σ-automaton (with state set Q, initial state i and final states F; we denote the action of a word u in Σ^* on the state q by $q\cdot u$). A state q in Q is "*accessible*" if $q=i\cdot u$, for some word u in Σ^*. If x is a letter in Σ, we call a state q *x-stable* if $q\cdot u=q$, where u is some positive power of x (i.e. $u=x$ or xx or xxx, etc.); q is stable if q is x-stable for some letter x. The "*behavior of* q", $|q|$, is the set $\{u\in\Sigma^*: q\cdot u\in F\}$.

We now define by induction the notion of an "n-state", for $0\leqq n$.

Definition. a) The state q will be called "*a* 0-*state via the letter* x" if

1. $|q|$ is nonempty;
2. q is x-stable.

b) q is an "$n+1$ *state via* x" if

1. q is x-stable;
2. there is some word v such that $q\cdot v$ is an n-state via y, for some letter $y\neq x$;

A state is an "n-state" if it is an n-state via x, for some letter x.

The easy proof of the next fact is omitted.

Lemma 6. Let $\mathbf{M}=(Q, i, F)$ be an automaton which accepts the language L. Then, for $1 \leqq n$, L admits alternations of size n iff there is some accessible n-state in Q.

Corollary 7. Let $\mathbf{M}=(Q, i, F)$ be an automaton which accepts the language L. Then, for $n>0$, $a(L)=n$ iff Q contains an accessible n-state but no accessible k-state with $k>n$. If Q has m states, then $a(L)=\infty$ iff there is an accessible m-state in Q.

Proof. We need only prove that if the cardinality of Q is m, and Q has an accessible m-state, say q_0, then $a(L)=\infty$. But, there is a sequence of words $u_0, u_1, \ldots, u_m$ such that if $q_{i+1}=q_i \cdot u_i$, for $i=0, 1, \ldots, m$, then q_i is an $m-i$ state via x_i, with $x_i \neq x_{i+1}$; since the states q_i cannot all be distinct, let s and t, $t>0$, be least such that $q_s=q_{s+t}$. It is easy to see that q_s is also an $n-s+kt$ state, for all $k>0$; hence $a(L)=\infty$.

Corollary 8. There is an algorithm to determine, given a regular language L, what the alternation number of L is.

Proof. Suppose one is given an accessible finite automaton with n states which accepts L. First one finds all the 0-states, by considering only paths of length $\leqq n$, then 1-states, etc. until one knows all the n-states. Then one applies the previous Corollary.

Questions: Is there an algorithm to determine, given a regular language L, the least n such that $L \in f(n)$? Is there an algorithm to determine the dot height of a regular language?

Acknowledgements

It is a pleasure to thank Douglas Bauer and Klaus Sutner for several helpful conversations. The author especially thanks Zoltan Esik for carefully reading the manuscript, correcting several errors and suggesting several improvements.

DEPARTMENT OF COMPUTER SCIENCE
STEVENS INSTITUTE OF TECHNOLOGY
HOBOKEN, NJ 07030
U.S.A.

References

[BK] J. A. BRZOZOWSKI and R. KNAST, "The dot-depth hierarchy of star-free languages is infinite", J. Comp. and System Sci., 16 (1978) 37—55.

[DS] F. DEJEAN and M. P. SCHUTZENBERGER, "On a question of Eggan", Information and Control vol. 9 (1966) 23—25.

(E) L. C. EGGAN, "Transition graphs and the star-height of regular events", Michigan Math. J. 10 (1963) 385—397.

[EZ] A. EHRENFEUCHT and P. ZEIGER, "Complexity measures for regular expressions", J. Comp. System Sci., 12 (1976) 134—146.

(Received 25 May, 1985)

Minimal keys and antikeys

By V. D. Thi

§ 1. Introduction

The relational model, defined by E. F. Codd [3] is one of the most investigated data base models of the last years. Many papers have appeared concerning combinatorial characterization of functional dependencies, systems of minimal keys and antikeys. A set of minimal keys and a set of antikeys form Sperner-systems. Sperner-systems and sets of minimal keys are equivalent in the sense that for an arbitrary Sperner-system S a family of functional dependencies F can be constructed so that the minimal keys of F are exactly the elements of S (cf. [4]).

In the present paper we propose some combinational algorithms to determine antikeys and minimal keys. In the second part of the paper, we are going to study connections between minimal keys and antikeys for special Sperner-systems.

We start with some necessary definitions.

Definition 1.1. Let Ω be a finite set, and denote $P(\Omega)$ its power set. The mapping F: $P(\Omega)\to P(\Omega)$ is called a closure operation over Ω if, for every $A, B\subseteq\Omega$,

(1) $A\subseteq F(A)$ (extensivity),

(2) $A\subseteq B$ implies $F(A)\subseteq F(B)$ (monotonity),

(3) $F(A)=F(F(A))$ (idempotency).

In few cases Ω is represented by the set $\{1, \ldots, n\}$ or by the set of columns of an $m\times n$ matrix $\mathbf{M}$. If we use the second representation, a special closure operation $F_{\mathbf{M}}$ can be defined over the set of the columns of $\mathbf{M}$:

The i-th column of $\mathbf{M}$ belongs to $F_{\mathbf{M}}(A)$ if and only if for any two rows of $\mathbf{M}$ which are identical on A they are equal on the i-th column, too.

It is easy to see, that $F_{\mathbf{M}}(A)$ is a closure operation. It is known (see [1]) that any closure operation F over a finite set Ω can be represented by an appropriate matrix $\mathbf{M}$, that is we can choose $\mathbf{M}$ and represent Ω by the set of the columns of $\mathbf{M}$ so that F coincides with $F_{\mathbf{M}}$.

Definition 1.2. Let F be a closure operation over Ω, and $A\subseteq\Omega$. We say that

— A is a key of F, if $F(A)=\Omega$.

— A is a minimal key of F, if A is a key of F and for any $B\subseteq A$, $F(B)=\Omega$ implies $B=A$, i.e. no proper subset of A is a key of F.

Let us denote by K_F the set of all minimal keys of F. It is clear that K_F forms a Sperner-system.

If K is a Sperner-system over Ω, let us define $S(K)$ as $S(K)=\min\{m: K=K_{F_M}:$ M is an $m\times n$ matrix representation of $\Omega\}$. For a Sperner-system K, we can define the set of antikeys, denoted by K^{-1}, as follows:

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

It is easy to see that K^{-1} is the set of subsets of Ω, which does not contain the elements of K and which is maximal for this property. They are the maximal non-keys. Clearly, K^{-1} is also a Sperner-system.

In this paper we assume that Sperner-systems playing the role of the set of minimal keys (antikeys) are not empty (do not contain the full set Ω).

§ 2. Connection between minimal keys and antikeys

The following important result was proved in [1], [5]:

Remark 2.1. If K is an arbitrary Sperner-system, then there exists a closure operation F, for which $K=K_F$ and a closure operation F', for which $K=K_{F'}^{-1}$.

Let us given an arbitrary Sperner-system $K=\{B_1, \ldots, B_m\}$ over Ω. We are now going to construct the set of antikeys K^{-1}. Let us follow the algorithm described below:

Let $K_1=\{\Omega\setminus\{a\}: a\in B_1\}$. It is easy to see that $K_1=\{B_1\}^{-1}$.

Let us suppose that we have constructed $K_q=\{B_1, \ldots, B_q\}^{-1}$ for $q<m$. We assume that $X_1, \ldots, X_p$ are the elements of K_q containing B_{q+1}. So $K_q=F_q\cup\{X_1, \ldots, X_p\}$, where $F_q=\{A\in K_q: B_{q+1}\not\subseteq A\}$. For all i $(i=1, \ldots, p)$, we construct the antikeys of $\{B_{i+1}\}$ on X_i in the analogous way as K_1, which are the maximal subsets of X_i not containing B_{q+1}. We denote them by $A_1^i, \ldots, A_{\tau_i}^i$ $(i=1, \ldots, p)$.

Let

$$K_{q+1}=F_q\cup\{A_t^i: A\in F_q\Rightarrow A_t^i\not\subset A,\ 1\leqq i\leqq p,\ 1\leqq t\leqq\tau_i\}.$$

We have to prove, that $K_{q+1}=\{B_1, \ldots, B_{q+1}\}^{-1}$. For this using the inductive hypothesis $K_q=\{B_1, \ldots, B_q\}^{-1}$ we show that

a) if $A\in K_{q+1}$ then A is the subset of Ω not containing B_t $(t=1, \ldots, q+1)$ and being maximal for this property, i.e. $A\in\{B_1, \ldots, B_{q+1}\}^{-1}$,

b) every $A\subseteq\Omega$ not containing the elements B_t $(t=1, \ldots, q+1)$ and being maximal for this property is an element of K_{q+1}. First we prove the validity of (a). Let $A\in K_{q+1}$. If $A\in F_q$ then A does not contain the elements B_t $(t=1, \ldots, q)$ and A is maximal for this property and at the same time $B_{q+1}\not\subseteq A$. Consequently, A is a maximal subset of Ω not containing B_t $(t=1, \ldots, q+1)$.

Let $A\in K_{q+1}\setminus F_q$. It is clear that there is an A_t^i $(1\leqq i\leqq p$ and $1\leqq t\leqq\tau_i)$ such that $A=A_t^i$. Our construction shows that $B_l\not\subseteq A_t^i$ for all l $(l=1, \ldots, q+1)$. Because A_t^i is an antikey of $\{B_{q+1}\}$ for X_i we obtain $A_t^i=X_i\setminus\{b\}$ for some $b\in B_{q+1}$. It is obvious that $B_{q+1}\subseteq A_t^i\cup\{b\}$. If $a\in\Omega\setminus X_i$ then, by the inductive hypothesis, for $A_t^i\cup\{a, b\}=X_i\cup\{a\}$ there exists B_s $(s=1, \ldots, q)$ such that $B_s\subseteq A_t^i\cup\{a, b\}$. X_i does not contain $B_1, \ldots, B_q$ by $X_i\in K_q$. Hence $a\in B_s$. If $B_s\setminus\{a\}\subseteq A_t^i$ then $B_s\subseteq A_t^i\cup\{a\}$. For every B_s $(1\leqq s\leqq q)$ with $B_s\subseteq X_i\cup\{a\}$ and $B_s\not\subseteq A_t^i$ we have $b\in B_s$. Hence $B_s\setminus\{a, b\}\subseteq A_t^i$. Consequently, there exists an $A_1\in F_q$ such that

$A_t^i \subset A_1$. This contradicts $A \in K_{q+1} \setminus F_q$. So there is a B_s $(1 \leqq s \leqq q)$ such that $B_s \subseteq A_t^i \cup \{a\}$.

Next we turn to the proof of (b). Suppose that A is the maximal subset of Ω not containing B_t $(1 \leqq t \leqq q+1)$. By the inductive hypothesis, there is a $Y \in K_q$ such that $A \subseteq Y$.

The first case: If $B_{q+1} \nsubseteq Y$ then Y does not contain $B_1, \ldots, B_{q+1}$. Because A is the maximal subset of Ω not containing B_t $(1 \leqq t \leqq q+1)$ we obtain $A = Y$. $B_{q+1} \nsubseteq Y$ implies $A \in F_q$. Consequently, we have $A \in K_{q+1}$.

The second case: If $B_{q+1} \subseteq Y$ then $Y = X_i$ holds for some i in $\{1, \ldots, p\}$ and $A \subseteq A_t^i$ holds for some t in $\{1, \ldots, \tau_i\}$. If there exists an $A_1 \in F_q$ such that $A_t^i \subset A_1$, then we also have $A \subset A_1$. By the definition of F_q it is clear that A_1 does not contain $B_1, \ldots, B_{q+1}$. This contradicts the definition of A. Hence $A_t^i \in K_{q+1}$. It is easy to see that A_t^i does not contain $B_1, \ldots, B_{q+1}$. By the definition of A we obtain $A = A_t^i$, i.e. $K_{q+1} = \{B_1, \ldots, B_{q+1}\}^{-1}$.

By the above proof it is clear that $K_m = \{B_1, \ldots, B_m\}^{-1}$. Thus we have

Theorem 2.2. $K_m = K^{-1}$.

Because K and K^{-1} are uniquely determined by each other, the determination of K^{-1} based on our algorithm does not depend on the order of $B_1, \ldots, B_m$.

Now we assume that the elementary step being counted is the comparison of two attribute names. Consequently, if we assume that subsets of Ω are represented as sorted lists of attribute names, then a Boolean operation on two subsets of Ω requires at most $|\Omega|$ elementary steps.

Let $K_o = \{\Omega\}$. According to the construction of our algorithm we have $K_q = F_q \cup \{X_1, \ldots, X_{t_q}\}$, where $1 \leqq q \leqq m-1$. Denote l_q the number of elements of K_q. It is clear that for constructing K_{q+1} the worst-case time of algorithm is $O(n^2(l_q - t_q)t_q)$ if $t_q < l_q$ and $O(n^2 t_q)$ if $l_q = t_q$. Consequently, the total time spent by the algorithm in the worst cases is

$$O\Big(n^2 \sum_{q=1}^{m-1} t_q u_q\Big), \quad \text{where} \quad |\Omega| = n,$$

$$u_q = \begin{cases} l_q - t_q & \text{if} \quad l_q > t_q, \\ 1 & \text{if} \quad l_q = t_q. \end{cases}$$

It is obvious that, if $F_q = \emptyset$, then $l_q = t_q$.

It can be seen that when there are only a few minimal keys (that is m is small) our algorithm is very effective, it does not requires exponential time in $|\Omega|$. In cases for which $l_q \leqq l_m$ $(\forall q: 1 \leqq q \leqq m-1)$ it is obvious that our algorithm requires a number of elementary operations which is not greater than $O(n^2|K||K^{-1}|^2)$. Thus, in these cases our algorithm finds K^{-1} in polynomial time in $|\Omega|$, $|K|$ and $|K^{-1}|$.

After Theorem 2.12 we shall give an example to show that our algorithm requires exponential time in $|\Omega|$. On the other hand K_q in each step of our algorithm is obviously a Sperner-system. It is known ([4]) that the size of arbitrary Sperner-system over Ω can not be greater than $\binom{n}{[n/2]}$, where $n = |\Omega|$. $\binom{n}{[n/2]}$ is asymptotically equal to

$\frac{2^{n+1/2}}{(\pi \cdot n)^{1/2}}$. Consequently, the worst-case time of our algorithm can not be more than exponential in the number of attributes.

Let $K^{-1}=\{A_1, \ldots, A_t\}$ be a set of antikeys. Let $R=\{h_0, h_1, \ldots, h_t\}$ be a relation over Ω given as follows: for all $a\in\Omega$, $h(a)=0$

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

If we consider R as a matrix, then R represents K (see [5]). Thus, based on our algorithm, for an arbitrarily given Sperner-system K, we can construct a matrix which represents K.

Example 2.3. Let $\Omega=\{1, 2, 3, 4, 5, 6\}$ and $K=\{(2, 3, 4), (1, 4)\}$. According to the above algorithm we have $K_1=\{(1, 3, 4, 5, 6), (1, 2, 4, 5, 6)\}\cup F_1$, where $F_1=\{(1, 2, 3, 5, 6)\}$, and $K_2=\{(3, 4, 5, 6), (2, 4, 5, 6), (1, 2, 3, 5, 6)\}$. It is obvious that $K^{-1}=K_2$.

We consider the following matrix:

The attributes:

$$\mathbf{M} = \begin{array}{c} \begin{matrix} 1 & 2 & 3 & 4 & 5 & 6 \end{matrix} \\ \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 \\ 2 & 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 3 & 0 & 0 \end{pmatrix} \end{array}$$

It is clear that $\mathbf{M}$ represents K.

Now we describe the "reverse" algorithm: for given Sperner-system considered as the set of antikeys we construct its origin. The following definitions are necessary for us.

Let F be a closure operation over Ω. Set

$$Z(F) = \{A \subseteq \Omega: F(A) = A\}$$

$$\text{and} \quad T(F) = \{A \subset \Omega: A\in Z(F) \quad \text{and} \quad A\subset B \Rightarrow F(B) = \Omega.$$

The elements of $Z(F)$ are called closed sets. It is clear that $T(F)$ is the family of maximal closed sets (except Ω). Now we prove the following lemma.

Lemma 2.4. Let F be a closure operation over Ω, and K_F the set of minimal keys of F. Then $K_F^{-1}=T(F)$.

Proof. Let A be an arbitrary antikey and suppose that $A\subset F(A)$. Hence $F(F(A))=F(A)=\Omega$. Consequently, A is a key. This contradicts $\forall B\in K_F$: $B\not\subseteq A$. If there is an A' such that $A\subset A'$ and $A'\in Z(F)\setminus\{\Omega\}$, then A' is a key. This contradicts $A'\subset\Omega$.

On the other hand, if A is a maximal closed set and there is a B $(B\in K_F)$ such that $B\subseteq A$, then $F(A)=\Omega$, which conflicts with the fact that $A\subset\Omega$. If $A\subset D(D\subseteq\subseteq\Omega)$, then it can be seen that $F(D)=\Omega$ (because A is the maximal closed set). Consequently, A is an antikey. The lemma is proved.

Now we construct an algorithm for finding a minimal key.

Let H be a Sperner-system and $\Omega \notin H$. We take a B $(B \in H)$ and an $a \in \Omega \setminus B$. We suppose that $B=\{b_1, \ldots, b_m\}$. Let $G=\{B_t \in H: a \notin B_t\}$ and $T_0=B \cup \{a\}$. define

$$T_{q+1}=\begin{cases} T_q \setminus \{b_{q+1}\} & \text{if } \forall B_i \in H \setminus G: T_q \setminus \{b_{q+1}\} \not\subseteq B_i, \\ T_q & \text{otherwise.} \end{cases}$$

Theorem 2.5. If H is a set of antikeys, then $\{T_0, T_1, \ldots, T_m\}$ are the keys and T_m is a minimal key.

Proof. By Remark 2.1 there exists a closure F such that $H=K_F^{-1}$. We prove the theorem by the induction. It is clear that T_0 is a key. If T_q and $T_{q+1}=T_q$, then it is obvious that T_{q+1} is a key. If $T_{q+1}=T_q \setminus \{b_{q+1}\}$ and $F(T_{q+1}) \neq \Omega$ then, by Lemma 2.4, there is a $B_t \in H$ such that $F(T_{q+1}) \subseteq B_t$. Hence $T_{q+1} \subseteq B_t$, which conflicts with the fact $\forall B_t \in H$: $T_{q+1} \not\subseteq B_t$. Consequently, T_{q+1} is a key.

Now suppose that A is a proper subset of T_m. If $a \notin A$, then, clearly, $F(A) \neq \Omega$. If $a \in A$, then there exists a $b_q \in B$ such that $b_q \in T_m \setminus A$ $(1 \leqq q)$. By the given algorithm there exists a $B_t \in H \setminus G$ such that $T_{q-1} \setminus \{b_q\} \subseteq B_t$. We obtain $A \subseteq T_m \setminus \{b_q\} \subseteq T_{q-1} \setminus \{b_q\} \subseteq B_t$ by $T_m \subseteq T_q$ $(0 \leqq q \leqq m-1)$. Hence $F(A) \neq \Omega$. Consequently, T_m is a minimal key. The theorem is proved.

Remark 2.6. Theorem 2.5 is also true if $T_0=\{b_1, \ldots, b_m\}$ is an arbitrary key. At this time define

$$T_{q+1}=\begin{cases} T_q \setminus \{b_{q+1}\} & \text{if } \forall B_i \in H: T_q \setminus \{b_{q+1}\} \not\subseteq B_i, \\ T_q & \text{otherwise.} \end{cases}$$

— It is clear that the worst-case time of the algorithm is $O(n^2 \cdot |H|)$, where $n=|\Omega|$, $|H|$ is the number of elements of H.

— It is best to choose B such that $|B|$ is minimal.

— If there is a B such that $\forall B_t \in H \setminus \{B\}$: $B_t \cap B=\emptyset$ and $a \in \bigcup_{B_t \in H \setminus \{B\}} B_t$ then $a \cup b$ is a minimal key $(\forall b \in B)$.

— If $(\Omega \setminus \bigcup_{B_t \in H} B_t) \neq \emptyset$, then $a \in \Omega \setminus \bigcup_{B_t \in H} B_t$ is a minimal key.

— Let $Y=\bigcup_{B_t \in H} B_t$ $(B_t \neq B)$. If $B \setminus Y \neq \emptyset$, then it is best to choose $T_0=(B \cap Y) \cup \{a\} \cup \{b\}$, where $b \in B \setminus Y$.

Remark 2.7. Let H be a Sperner-system $(\Omega \notin H)$ and $A \subset \Omega$. We can give an algorithm (which is analogous to the above one) to decide whether A is a key or not. If A is a key, then this algorithm finds an A' such that $A' \subseteq A$ and A' is a minimal key.

Remark 2.8. In the paper [5] the equality sets of the relation are defined as follows: Let $\mathbf{R}=\{h_1, \ldots, h_m\}$ be a relation over Ω. For $i \neq j$, we denote by E_{ij} the set $\{a \in \Omega: h_i(a)=h_j(a)\}$, where $1 \leqq i \leqq m$, $1 \leqq j \leqq m$. Now we define $\mathbf{M}=\{E_{ij}: \nexists E_{pq}$ such that $E_{ij} \subset E_{pq}\}$. Practically, it is possible that there are some E_{ij} which are equal to each other. We choose one E_{ij} from $\mathbf{M}$. According to Lemma 2.4 it can be seen that $\mathbf{M}$ is the set of antikeys of $K_{F_\mathbf{R}}$ (we consider $\mathbf{R}$ as a matrix).

Example 2.9. Let $\Omega=\{1, 2, 3, 4, 5, 6\}$ and **R** be the following relation:

$$\begin{pmatrix} 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 \\ 2 & 0 & 0 & 1 & 2 & 2 \\ 0 & 1 & 2 & 2 & 0 & 3 \\ 3 & 2 & 1 & 0 & 3 & 0 \end{pmatrix}$$

It can be seen that $\mathbf{M}=\{(1, 2), (3, 4, 5), (4, 6)\}$, where $E_{14}=\{1, 2\}$, $E_{15}=\{4, 6\}$ and $E_{25}=\{3, 4, 5\}$. By Theorem 2.5 and Remark 2.6, it is clear that $\{1, 3\}$, $\{1, 4\}$, $\{1, 5\}$, $\{1, 6\}$, $\{2, 3\}$, $\{2, 4\}$, $\{2, 5\}$, $\{2, 6\}$ are the minimal keys. We use the algorithm (Theorem 2.5) with $T_0=\{3, 4, 6\}$ and $T_0=\{4, 5, 6\}$, then it can be seen that $\{3, 6\}$ and $\{5, 6\}$ are minimal keys. Thus, based on this algorithm for an arbitrarily given relation **R** we can find a minimal key of **R**.

Let K be an arbitrary Sperner-system. The following theorem has been proved in [2].

Theorem 2.10.

$$\binom{S(K)}{2} \geqq |K^{-1}| \geqq S(K)-1.$$

Denote by $\binom{\Omega}{k}$ the family of all k-element subsets of Ω. Let $F_k(n)=\max\left\{S(K): K\subseteqq\binom{\Omega}{k},\ |\Omega|=n\right\}$.

Theorem 2.11. ([6])

$$F_k(n) \geqq \sqrt{2}\binom{2k-2}{k-1}^{1/2\cdot[n/(2k-2)]}.$$

We define the function f_{2k-1}: $N\to N$ for $2k-1\leqq n$ by

$$f_{2k-1}(n)=\begin{cases} \binom{2k-1}{k-1}^{n/(2k-1)} & \text{if } n\equiv 0 \pmod{(2k-1)}, \\ \binom{2k-1}{k-1}^{[n/(2k-1)]-1}\times\binom{2k-1+p}{k-1} & \text{if } n\equiv p \pmod{(2k-1)} \\ & \text{and } 1\leqq p\leqq k-1, \\ \binom{2k-1}{k-1}^{[n/(2k-1)]}\times\binom{p}{k-1} & \text{if } n\equiv p \pmod{(2k-1)} \\ & \text{and } k\leqq p\leqq 2k-2, \end{cases}$$

and the function f_{2k-2} for $2k-2\leqq n$ by

$$f_{2k-2}(n)=\begin{cases} \binom{2k-2}{k-1}^{n/(2k-2)} & \text{if } n\equiv 0 \pmod{(2k-2)}, \\ \binom{2k-2}{k-1}^{[n/(2k-2)]-1}\times\binom{2k-2+p}{k-1} & \text{if } n\equiv p \pmod{(2k-2)} \\ & \text{and } 1\leqq p\leqq k-1, \\ \binom{2k-2}{k-1}^{[n/(2k-2)]}\times\binom{p}{k-1} & \text{if } n\equiv p \pmod{(2k-2)} \\ & \text{and } k\leqq p\leqq 2k-3, \end{cases}$$

where N denotes the set of natural numbers. Let us take a partition $\Omega = X_1 \cup \ldots \cup X_m \cup W$, where $m = \left[\frac{n}{2k-1}\right]$ and $|X_i| = 2k-1$ $(1 \leqq i \leqq m)$.

Let

$$K = \{B\colon |B| = k, B \subseteq X_i, \forall i\} \quad \text{if} \quad |W| = 0.$$

$$K = \{B\colon |B| = k, B \subseteq X_i \ (1 \leqq i \leqq m-1) \text{ and } B \subseteq X_m \cup W\} \quad \text{if} \quad 1 \leqq |W| \leqq k-1.$$

$$K = \{B\colon |B| = k, B \subseteq X_i \ (1 \leqq i \leqq m) \text{ and } B \subseteq W\} \quad \text{if} \quad k \leqq |W| \leqq 2k-2.$$

It is clear that

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

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

$$\text{if} \quad 1 \leqq |W| \leqq k-1.$$

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

$$\text{if} \quad k \leqq |W| \leqq 2k-2.$$

It can be seen that $f_{2k-1}(n) = |K^{-1}|$. If we take a partition: $\Omega = X_1 \cup \ldots \cup X_m \cup W$; where $m = \left[\frac{n}{2k-2}\right]$ and $|X_i| = 2k-2$, in an analogous way we

$$K = \{B\colon |B| = k, B \subseteq X_i, \forall i\} \quad \text{if} \quad |W| = 0.$$

$$K = \{B\colon |B| = k, B \subseteq X_i \ (1 \leqq i \leqq m-1) \text{ and } B \subseteq X_m \cup W\} \quad \text{if} \quad 1 \leqq |W| \leqq k-1.$$

$$K = \{B\colon |B| = k, B \subseteq X_i \ (1 \leqq i \leqq m) \text{ and } B \subseteq W\} \quad \text{if} \quad k \leqq |W| \leqq 2k-3.$$

It is clear that

$$f_{2k-2}(n) = |K^{-1}| \quad \text{and} \quad f_{2k-2}(n) \geqq \binom{2k-2}{k-1}^{[n/(2k-2)]}.$$

Theorem 2.12. Let $\Omega = \{1, \ldots, n\}$.
If $n \equiv 0 \pmod{(2k-2)(2k-1)}$, then $f_{2k-1}(n) > f_{2k-2}(n)$. For a fixed k,

$$\lim_{n \to \infty} \frac{f_{2k-1}^{(n)}}{f_{2k-2}^{(n)}} = \infty.$$

Proof. If $k=2$, then it is easy to prove that $\forall n\colon f_3(n) \geqq f_2(n)$. If $n=6$ or $n \geqq 8$, then $f_3(n) > f_2(n)$. Let

$$F = \frac{\binom{2k-1}{k-1}^{n/(2k-1)}}{\binom{2k-2}{k-1}^{n/(2k-2)}} = \frac{\binom{2k-1}{k}^{n/(2k-1)}}{\binom{2k-2}{k-1}^{n/(2k-2)(2k-1)}}.$$

It is known that $n!=\sqrt{2\pi n}\left(\frac{n}{e}\right)^n \times e^{\theta n/(12n)}$, where $0<\theta_n<1$. So

$$F \geqq \frac{\left(\frac{2k-1}{k}\right)^{n/(2k-1)}}{2^{n/(2k-1)} \cdot \left(\frac{e^{\theta_n/(12(2k-2))}}{\sqrt{\pi(k-1)}}\right)^{n/(2k-2)(2k-1)}} \geqq \frac{\left(1-\frac{1}{2k}\right)^{n/(2k-1)}}{\left(\frac{e^{1/(24(k-1))}}{\sqrt{\pi(k-2)}}\right)^{n/(2k-2)(2k-1)}} = E.$$

For this E we obtain, that

$$T = \ln E = \frac{n}{2k-1}\left(\ln\left(1-\frac{1}{2k}\right)+\frac{1}{2k-2}\left(\frac{1}{2}\ln(\pi(k-1))-\frac{1}{24(k-1)}\right)\right)$$

and by

$$\left|\ln\left(1-\frac{1}{2k}\right)\right| \leqq \frac{1}{2k-1}$$

we have

$$T \geqq \frac{n}{2k-1}\left(\frac{1}{2k-2}\left(\frac{1}{2}\ln(\pi(k-1))-\frac{1}{24(k-1)}\right)-\frac{1}{2k-1}\right).$$

It can be seen that if $k=3$, then

$$\frac{1}{2k-2}\left(\frac{1}{2}\ln(\pi(k-1))-\frac{1}{24(k-1)}\right)-\frac{1}{2k-1} > 0$$

and, for every $k \geqq 4$,

$$\frac{1}{2}\ln(\pi(k-1))-\frac{1}{24(k-1)} > 1.$$

Hence

$$\frac{1}{2k-2}\left(\frac{1}{2}\ln(\pi(k-1))-\frac{1}{24(k-1)}\right)-\frac{1}{2k-1} > 0.$$

Consequently, if $n \equiv 0 \pmod{(2k-2)(2k-1)}$, then $f_{2k-1}(n) > f_{2k-2}(n)$. Now let n be an arbitrary natural number. It can be seen that, for a fixed k, there exists a number $M>0$ such that

$$\frac{\binom{2k-1+p}{k-1}}{\binom{2k-1}{k-1}^{1+(p/(2k-1))}} < M, \quad \frac{\binom{p}{k-1}}{\binom{2k-1}{k-1}^{p/(2k-1)}} < M,$$

$$\frac{\binom{2k-2+p}{k-1}}{\binom{2k-2}{k-1}^{1+(p/(2k-2))}} < M \quad \text{and} \quad \frac{\binom{p}{k-1}}{\binom{2k-2}{k-1}^{p/(2k-2)}} < M.$$

Hence $\ln E \underset{n\to\infty}{\to}\infty$. Consequently, $F \underset{n\to\infty}{\to}\infty$. Thus,

$$\frac{f_{2k-1}(n)}{f_{2k-2}(n)} \underset{n\to\infty}{\to}\infty.$$

(It is easy to see that $k=2$ is also true.) The theorem is proved.

As a consequence of Theorem 2.12 and Theorem 2.10 we have

Corollary 2.13

$$F_k(n) \geqq \sqrt{2f_{2k-1}(n)}.$$

Example 2.14. In Theorem 2.12 let $k=2$. Then we have $n-1 \leqq |K| \leqq n+2$ and $3^{(n/4)} < f_3(n)$, where $n=|\Omega|$., i.e. $3^{(n/4)} < |K^{-1}|$. Thus, we always can construct an example, in which the number of K (minimal keys) is not greater than $n+2$, but the size of K^{-1} (antikeys) is exponential in the number of attributes.

§ 3. Some special Sperner-systems

In this section we investigate connections between the minimal keys and antikeys for some special Sperner-systems.

The notion of saturated Sperner-system is defined in [7], as follows:

A Sperner-system K over Ω is saturated if for any $A \subseteqq \Omega$, $K \cup \{A\}$ is not a Sperner-system.

An important result in [7] has been proved; if K is a saturated Sperner-system then $K=K_F$ uniquely determines F, where F is a closure operation.

Now we investigate some special Sperner-systems which are strictly connected with saturated Sperner-systems.

We consider the following example.

Example 3.1. Let $\Omega=\{1, 2, 3, 4, 5, 6\}$ and $N=\{(1, 2), (3, 4), (5, 6)\}$ be a Sperner-system. It can be seen that $N^{-1}=\{(1, 3, 5), (1, 3, 6), (1, 4, 5), (1, 4, 6), (2, 3, 5), (2, 3, 6), (2, 4, 5), (2, 4, 6)\}$. Let $K=N \cup N^{-1}$. It is clear that K is saturated. We use the algorithm which finds a set of antikeys. Then $K^{-1}=\{(1, 3), (1, 4), (1, 5), (1, 6), (2, 3), (2, 4), (2, 5), (2, 6), (3, 5), (3, 6), (4, 5), (4, 6)\}$.

By the fact that $K^{-1} \cup \{1, 2\}$ is a Sperner-system it is obvious that K^{-1} is not saturated. Thus, we have

Corollary 3.2. There is a K so that K is saturated and K^{-1} is not saturated.

Now we define the following notion.

Definition 3.3. Let K be a Sperner-system over Ω. We say that K is embedded, if for every $A \in K$ there is a $B \in H$ such that $A \subset B$, where $H^{-1}=K$. We have

Theorem 3.4. Let K be a Sperner-system over Ω. K is saturated if and only if K^{-1} is embedded.

Proof. Let K be a saturated Sperner-system. According to the definition of K^{-1} it is clear that K^{-1} is embedded. Assume that K^{-1} is an embedded Sperner-system, but K is not saturated. Consequently, for K there exists an $A \subset \Omega$ such that $K \cup \{A\}$ is a Sperner-system. It can be seen that, for every $C \in K$, we have $C \subset \Omega$ (because of $\Omega \notin K$). Hence we can construct B such that $A \subseteqq B$, $K \cup \{B\}$ is a Sperner-system and, for every $B'(B \subset B')$, there is a $C \in K$ with $C \subseteqq B'$. It can be seen that $B \in K^{-1}$. This contradicts the fact that K^{-1} is embedded. The proof is complete.

Now we define an inclusive Sperner-system.

Definition 3.5. Let K be a Sperner-system over Ω. We say that K is inclusive, if for every $A \in K$, there exists a $B \in K^{-1}$ such that $B \subset A$. We have

Theorem 3.6. K is an inclusive Sperner-system if and only if K^{-1} is a saturated one.

Proof. Now, assume that K is an inclusive Sperner-system but K^{-1} is not saturated. By the definition of K^{-1}, there is a $B \in (K^{-1})^{-1}$ such that $K^{-1} \cup \{B\}$ is a Sperner-system. By Remark 2.1, for K there is a closure operation F such that $K = K_F$. If $F(B) \subset \Omega$, then by Lemma 2.4 there exists an $A \in K^{-1}$ with $F(B) \subseteqq A$ (the set of antikeys is family of the maximal closed sets), which conflicts with the fact that $K^{-1} \cup \{B\}$ is a Sperner-system. Consequently, B is a key. If we use the algorithm which finds a minimal key in Theorem 2.5, then it can be seen that there exists a $B'(B' \subseteqq B)$ such that $B' \in K$, and it is clear that $K^{-1} \cup \{B'\}$ is a Sperner-system. This contradicts the definition of K. Thus, K^{-1} is saturated.

On the other hand by the definition of K^{-1} and by the assumption that K^{-1} is saturated it is clear that K is an inclusive Sperner-system. The theorem is proved. Now, we have the following corollary by Theorem 3.4 and Theorem 3.6.

Corollary 3.7. Let K be a Sperner-system over Ω. Denote H a Sperner-system, for which $H^{-1} = K$. The following facts are equivalent:

(1) K is saturated,
(2) K^{-1} is embedded,
(3) H is inclusive.

Proposition 3.8. There exists a Sperner-system K such that

(1) K is saturated, but K^{-1} is not saturated.
(2) K is saturated, but H is not saturated.
(3) K is embedded, but K^{-1} is not embedded.
(4) K is embedded, but H is not embedded.
(5) K is inclusive, but K^{-1} is not inclusive.
(6) K is inclusive, but H is not inclusive,

where H denotes a Sperner-system for which $H^{-1} = K$.

Proof. From Example 3.1 we have (1). By Theorem 3.4, $(K^{-1})^{-1}$ is not embedded in this example. Hence we have (3). By Theorem 3.6, in Example 3.1 H is inclusive, where $H^{-1} = K$. Now, we suppose that, if K is inclusive, then the set of antikeys of K is also inclusive. Consequently, in Example 3.1, H is inclusive, and K is an inclusive Sperner-system. From Theorem 3.6, K^{-1} is saturated. This constradicts the fact that K^{-1} in Example 3.1 is not a saturated Sperner-system. Hence we have (5). (2) can be proved as follows: Let K be a Sperner-system. Let $K^1 = K$ and, for $n \geqq 2$, define

K^n by the equality $(K^n)^{-1}=K^{n-1}$. We know that the number of the Sperner-systems over Ω is finite (at most $2^{2^{|\Omega|}}$). On the other hand, K and K^{-1} are determined uniquely by each other. Consequently, there exists a number m $(2\leqq m\leqq 2^{2^{|\Omega|}})$ such that $K^m=K$ and $K^{m-1}=K^{-1}$. If we suppose that K is saturated, then H is also saturated, where $H^{-1}=K$. This means that for every p with $2\leqq p<m$, K^p is also saturated. This contradicts Corollary 3.2. Thus, there is a Sperner-system K such that K is saturated, but H is not saturated. By similar arguments we have also (4) and (6). The proposition is proved.

Acknowledgement

The author would like to take this opportunity to express deep gratitude to Professor J. Demetrovics and L. Hannák for their help, valuable comments and suggestions.

COMPUTER AND AUTOMATION INSTITUTE
HUNGARIAN ACADEMY OF SCIENCES
KENDE U. 13—17.
BUDAPEST, HUNGARY
H—1502

References

[1] ARMSTRONG, W. W., Dependency Structures of Data Base Relationships, Information Processing 74, North-Holland Publ. Co. (1974) 580—583.
[2] BÉKÉSSY, A., DEMETROVICS J., HANNÁK L., KATONA G. O. H., FRANKL P., On the number of maximal dependencies in data relation of fixed order. Discrete Math., 30 (1980) 83—88.
[3] CODD, E. F., Relational model of data for large shared data banks. Communications of the ACM, 13, (1970) 377—384.
[4] DEMETROVICS, J., On the equivalence of candidate keys with Sperner systems. Acta Cybernetica 4 (1979) 247—252.
[5] DEMETROVICS, J., Relációs adatmodell logikai és strukturális vizsgálata. MTA SZTAKI Tanulmányok, Budapest, 114 (1980) 1—97.
[6] DEMETROVICS J., FÜREDI Z., KATONA G. O. H., Minimum matrix representation of closure operations. Preprint of the mathematical institute of the Hungarian academy of sciences, Budapest, 12 (1983) 1—22.
[7] DEMETROVICS J., FÜREDI Z., KATONA G., A függőségek és az individumok száma közötti kapcsolat összetett adatrendszerek esetén. Alkalmazott Matematikai Lapok 9 (1983) 13—21.

(Received April 2, 1985.)

On the index of concavity of neighbourhood templates

J. PECHT

Abstract

In automatic image analysis with parallel algorithms or parallel processors successive Minkowski-operations (like erosions and dilatations) with a given neighbourhood template (also referred to as structuring element), T, play an important role. It can be shown that, after a certain number of such steps, the neighbourhood template E, which contains only the extreme points of T, can be used instead of T. This number of steps is called the index of concavity of T. In bit-plane oriented parallel processors this fact can be used to speed-up pattern recognition algorithms. The speed-up is only assymptotical and its practical performance depends upon whether the index of concavity is low or high. In this paper it is shown that for the practical cases of convex or small templates the index is very small, namely at most 2 or 3 resp. which ensures speed-up for this type of templates. As against to this result it is, however, also shown that, theoretically, arbitrary high indices of concavity can be achieved for appropriately chosen (exotic) templates.

1. Introduction

Minkowski-operations play an important role in automatic image analysis, particulary in optical material control. Herein, after thresholding the video image (from camera) appropriatelly a binary image, b, (usually 256×256 or 512×512 pixels) is produced. b is also called a bit-plane. The bit-plane b is eroded repeatedly and after each step of erosion a measurement of area, boundary length and/or number of particles is done. Assembling these numbers in one (or 3) feature vector(s), convenient statistical classification procedures can be applied to get final decision of certain material properties. Depending on the material properties to be jugded upon various neighbourhood templates must be chosen (1, 2).

In bit-plane oriented parallel array processors (so called: bitplane processors) (3, 4, 5, 6) a straight forward implementation of this operations needs Ct elementary parallel bitwise logical operations where t is the number of elements in T (1, 2). There it is also shown that in case of convex, symmetric templates $Cu/2$ operations are sufficient where u is the number of boundary points of T. In (7) this result was improved be showing that, for any template T, assymptotically already Ce operations are also sufficient where e is the number of extreme points of T. This result relies on the fact that, after a certain number of steps, the Minkowski-operation with T can be replaced by the same operation using only the template E which contains just the extreme

points of T. This number depends on T and is called the *index of concavity* of T. It is denoted $\mu(T)$.It is clear that the assymptotic speed gain is only achieved if $\mu(T)$ is low. It is the intent of this paper to show that for the practically important cases of small (i.e. 3×3) templates or (possibly big) convex templates the index of concavity does not exceed 3 or 2 (resp.). On the other hand, exotic templates (with some few and, however, wide spread points) can yield arbitrarily high indices of concavity.

After presenting some necessary mathematical definitions and facts in chapter 2 we derive our claim as cited above in chapter 3.

2. Basic definitions and facts

Definition 1. Let $\mathbf{Z}$ denote the set of integers. Any finite subset T of $\mathbf{Z}^2$ is called a *neighbourhood template*. Between two templates T and U the sum $T\oplus U$ is defined as $\{t+u/t\in T \text{ and } u\in U\}$ ($+$ is here the usual componentwise vector sum). For any template T the sequence $(kT)_{k\in\mathbf{N}}$ ($\mathbf{N}=\{0, 1, 2, \ldots\}$) is recursively defined by

$$0T = \{\mathbf{0}\}, \tag{1}$$

$$(k+1)\,T = kT\oplus T \quad (k \geqq 0). \tag{2}$$

Here, $\mathbf{0}=(0, 0)$ is the 2-dimensional origin in $\mathbf{Z}^2$. $x\in T$ is called an *extreme point* of T, if any representation $x=\sum_{t\in T} a_t t$ with $a_t\geqq 0$, $a_t\in\mathbf{R}$ and $\sum_{t\in T} a_t=1$ implies $a_x=1$, and $a_t=0$ for $t\neq x$. The set of extreme points of T is denoted E or $E(T)$.

Proposition 1. (7) For any template T there is a $k_0\in\mathbf{N}$ such that

$$kT\oplus T = kT\oplus E \quad (k \geqq k_0). \tag{3}$$

Definition 2. For any template T let $\mu(T)$ denote the minimal k_0 such that Proposition 1 holds. $\mu(T)$ is called the *index of concavity* of T.

Definition 3. For any template T let $\overline{T}$ denote the *convex hull* of T (in $\mathbf{R}^2$), formally:

$$\overline{T} := \Big\{\sum_{t\in T} a_t t/a_t \geqq 0,\ a_t\in\mathbf{R},\ \sum_{t\in T} a_t = 1\Big\} \tag{4}$$

and $\dot{T}:=\overline{T}\cap\mathbf{Z}^2$. A template T is called *convex* if $T=\dot{T}$. The norm $\|T\|$ of T is the maximal absolute value of all occuring coordinates of all elements of T. T is called *small*, if $\|T\|\leqq 1$.

Equipped with these preliminaries we proceed to prove our claims.

3. The index of concavity of certain classes of templates

Theorem 1. For any $k\in\mathbf{N}$, there is a neighbourhood template T with $\|T\|\leqq \leqq\sqrt{(k/3)}+5$ such that $\mu(T)\geqq k$.

Proof. Let $k\in\mathbf{N}$, and consider the template $T=\{x_1, x_2, x_3, x_4\}$ with $x_1=(n, 0)$, $x_2=(0, n-1)$, $x_3=(-(n-2), -(n-2))$, and $x_4=(0, 0)$ where n is the greatest odd

natural number less than or equal to $\sqrt{(k/3)}+5$. Note that, in all cases, n is odd and not smaller than 5. Let $hT\oplus E=hT\oplus T$. We show that $h\geqq k$. Because

$$\mathbf{0} = (0, 0)\in hT\oplus T, \quad \mathbf{0}\in hT\oplus E.$$

Thus $\mathbf{0}=k_1x_1+k_2x_2+k_3x_3$, where $k_1, k_2, k_3\geqq 0$ and $h+1\geqq k_1+k_2+k_3>0$ ($k_1, k_2, k_3\in\mathbb{N}$). So at least one k_i is greater than 0 and $k_1n=k_3(n-2)=k_3(n-2)=k_2(n-1)$. Because $n, n-1$ and $n-2$ have no common divisor except unity we conclude that $k_1\geqq(n-1)(n-2)$, $k_2\geqq(n-2)n$, and $k_3\geqq n(n-1)$. Thus $h+1\geqq 3(n-2)^2\geqq 3(\sqrt{(k/3)}+5-2)^2\geqq k+1$.

Q.E.D.

Theorem 2. For any small template we have $\mu(T)\leqq 3$.

Proof. The validity of this claim was checked by an appropriate computer program: For all small templates, T, their sets of extreme points, E, were computer and the first k were searched for which $kT\oplus E=kT\oplus T$. One proves easily that these k equal $\mu(T)$.

Q.E.D.

Theorem 3. For any convex template T we have $\mu(T)\leqq 2$.

Proof. A proof can be obtained by combining some partial results of (8) and (9). In (8) it is shown that $\overset{\cdot}{(d+1)T}=\overset{\cdot}{dT}\oplus E$ for any (d-dimensional) template T which yields, for our case $d=2$, the claim $\overset{\cdot}{3T}=\overset{\cdot}{2T}\oplus E$. In (9) it is shown that $\overset{\cdot}{kT}=kT$ for all 2-dimensional convex templates and any $k\geqq 0$. Thus, we get

$$3T = \overset{\cdot}{3T} = \overset{\cdot}{2T}\oplus E = 2T\oplus E. \tag{5}$$

This proves our theorem.

Q.E.D.

In case of rectangular convex templates we get even lower indices:

Theorem 4. For the rectangular template $T=\{n, n+1, \ldots, n+i\}\times\{m, m+1, \ldots, m+j\}$, we have $\mu(T)=1$.

Proof. Let $x=(x_1, x_2)\in 2T$. Then $2n\leqq x_1\leqq 2n+2i$ and, consequently, $n\leqq (x_1-n)\leqq n+2i$. If $x_1-n>n+i$ then $n-i\leqq x_1-(n+i)\leqq n+i$ and $n\leqq x_1-(n+i)\leqq n+i$. A similar argument shows that either $m\leqq x_2-(m+j)\leqq m+j$ or $m\leqq x_2-m\leqq m+j$. This proves our theorem because $E(T)=\{n, n+i\}\times\{m, m+j\}$.

Q.E.D.

4. Summary

In a former paper (7) the author had proved that, for any neighbourhood template T, there is a number, $\mu(T)$, such that $kT\oplus T=kT\oplus E$ $(k\geqq\mu(T))$ where E is the template containing only the extreme points of T. $\mu(T)$ is called the index of concavity of T. In image analysis with bit-oriented parallel computers this fact can be used to speed-up pattern classification algorithms which make excessive use of

Minkowski-operations like erosion, dilation, opening and closing by appropriately chosen neighbourhood templates. This speed-up is only achieved if $\mu(T)$ is low. In this paper, it is shown that this is, in fact, true for all practically important templates, i.e., for (arbitrary) convex ones and small ones. Nevertheless, exotic templates can be derived having arbitrarily high indices of concavity.

Acknowledgement

The author thanks Ms. I. Mayer for writing the program for the proof of Theorem 2 and Dr. D. Vollath for some useful editorial hints.

INSTITUT FÜR MATERIAL- UND
FESTKÖRPERFORSCHUNG III
KERNFORSCHUNGZENTRUM KARLSRUHE
POSTFACH 3640
D—75 KARLSRUHE 1
WST-GERMANY

Literature

[1] SERRA, J., Introduction a la morphologie mathematique, Le cahier du Centre de Morphologie Mathematique de Fontainbleau, Fascicule 3, 1969.
[2] VOLLATH, D., The image analysing system PACOS, Praktische Metallographie *19*, 7—23 and 94—103, (1979).
[3] BATCHER, K. E., Bit-serial parallel processing systems, IEEE Transactions on Computers, *C-31*, 5, May 1982, 377—384.
[4] PARKINSON, D., An introduction to array processors, Reprint from Systems International, Nov 1977.
[5] BABENHAUSERHEIDE, M., PECHT, J., PPSTAR. A FORTRAN IV software package to support the development of portable image processing software, Proceedings of the 6th International Conference on Pattern Recognition, Munich, October 1982, p. 1204.
[6] PECHT, J., VOLLATH, D. and GRUBER, P., A fast bit-plane processor for quantitative image processing in a minicomputer environment, hardware and software architecture, in: Schüßler, H. V., (editor): Signal Processing II: Theory and Applications, EUSIPCO 1983, pp. 809—812.
[7] PECHT, J., Speeding-up successive Minkowski-operations with bit-plane computers. Pattern Recognition Letters 3 (1985), pp. 113—117.
[8] SZWERINSKI, H., Zellularautomaten mit symmetrischer lokaler Transformation, Dissertation, TU Braunschweig (FRG), 1982 pp. 72—77.
[9] PECHT, J., Ein weiterer Ansatz zur Mustertransformation und -erkennung in zellularen Räumen, Dissertation, TU Braunschweig (FRG), 1980, pp. 206—213.

(Received July 31, 1985.)

Optimization of multi valued logical functions based on evaluation graphs

A. VARGA

Dedicated to Professor K. Tandori on his 60th birthday

Abstract

In this paper we discuss simplifications of multi valued logical functions. The simplification is carried out in the following way. We associate tree graphs with the disjunctive or conjunctive normal forms of the functions. Under certain conditions some vertices of these trees can be omitted. This cancellation will correspond to reduction of terms or variables in the original function.

After all possible simplifications a normal form, which is equivalent to the function in question, is obtained.

1. Definitions, notations

Let $k(>2)$ be a natural number and ε_k the set $\{0, 1, 2, \ldots, k-1\}$. Any function $f\colon \varepsilon_k^n \to \varepsilon_k$ is called a k-valued logical function of n-variables where ε_k^n denotes the Cartesian product of n copies of ε_k. These functions are often given by their truth-tables and they will also be denoted by $f(\mathbf{X}^n)$ or $f(\mathbf{X}^n)=f(X_1, X_2, \ldots, X_n)$. The set of k-valued logical function will be denoted by P_k. Several properties valid in the theory of ordinary two-valued logic remain true in the theory of k-valued logic as well. But in the case $k\geqq 3$ certain characteristics are essentially different from those in ordinary logic.

A major problem is the definition of negation, since it can be defined in several ways.

Definition 1. Let $A_i \in \{0, 1, \ldots, k-1\}$, $i=1, 2, \ldots, n$; $n\geqq 2$. Then the operators defined by

$$A_1 \wedge A_2 \wedge \ldots \wedge A_n = \min(A_1, A_2, \ldots, A_n)$$

and

$$A_1 \vee A_2 \vee \ldots \vee A_n = \max(A_1, A_2, \ldots, A_n)$$

are called the conjunction and disjunction of the variables $A_1, A_2, \ldots, A_n$, respectively.

The following identities can easily be proved:

I. $A \wedge B = B \wedge A$,

$A \vee B = B \vee A$, for every A, B.

II. $A \wedge (B \wedge C) = (A \wedge B) \wedge C$,

$A \vee (B \vee C) = (A \vee B) \vee C$, for every A, B, C.

III. $A \wedge (B \vee C) = (A \wedge B) \vee (A \wedge C)$,

$A \vee (B \wedge C) = (A \vee B) \wedge (A \vee C)$, for every A, B, C.

IV. $A \vee A = A$,

$A \wedge A = A$, for every A.

V. $A \wedge (k-1) = A$,

$A \vee \emptyset = A$, for every A.

Below we give two types of negation: one for logical constants and one for logical variables.

Definition 2. Let $A \in \varepsilon_k$. Then

$$\bar{A} = (k-1) - A.$$

Definition 3. If X is a variable then $\bar{X}$ denotes that function the actual value of which is the negation (in the sense of Definition 2) of the actual value of X. Let us introduce the following unary operator

$${}^aX^b = \begin{cases} k-1, & \text{if } a \leqq X \leqq b, \\ 0 & \text{elsewhere,} \end{cases}$$

where $a, b, X \in \varepsilon_k$ and $a \leqq b$ are fixed. It should be noticed that ${}^aX^b$ is two-valued. By Definition 3, the negation of ${}^aX^b$ is

$$\overline{{}^aX^b} = \begin{cases} 0, & \text{if } a \leqq X \leqq b, \\ k-1 & \text{elsewhere,} \end{cases}$$

where $a, b, X \in \varepsilon_k$ and $a \leqq b$ are fixed. The formulae in the theory of k-valued logic, similarly to those of two valued logic, will be given by recursive definition.

Definition 4.

(0) The elements of ε_k are k-valued logical formulae;
(1) $X_1, X_2, \ldots, X_n, {}^{a_1}X^{b_1}, {}^{a_2}X^{b_2}, \ldots, {}^{a_n}X^{b_n}$ are k-valued logical formulae;
(2) If F is a k-valued logical formula, then $\bar{F}$ is a k-valued formula;
(3) If F and G are k-valued logical formulae, then $F \vee G$, $F \wedge G$ are k-valued logical formulae;

(4) Every k-valued logical formula can be obtained by a repeated application of (0)—(3).

In what follows the letters $f, g, \ldots$ will denote functions and the capital letters $F, G, \ldots$ will denote formulae. By a function of n-variables we mean a k-valued logical function of n-variables ($k \geqq 3$).

Value assignement. The ordered n-tuple $(X_1, X_2, \ldots, X_i|A_i, \ldots, X_n)$ is called a value assignement of the i-th variable. If every variable has value simultaneously, then the ordered n-tuple $(\mathbf{X}^n|\mathbf{A}^n)=(X_1|A_1, X_2|A_2, \ldots, X_n|A_n)$ is simply called a value assignement.

Let $f(\mathbf{X}^n)$ be a function. Then

$$\mathrm{f}(\mathbf{X}^n|\mathbf{A}^n) = f(X_1|A_1, X_2|A_2, \ldots, X_n|A_n)$$

denotes the fact that X_i is replaced by A_i, where $A_i \in \varepsilon_k$ $i=1, 2, \ldots, n$. The value $f(X_1|A_1, X_2|A_2, \ldots, X_n|A_n)$ is called the value of $f(\mathbf{X}^n)$ under the value assignement $(X_1|A_1, X_2|A_2, \ldots, X_n|A_n)$. Below the value assignement $(X_1|A_1, X_2|A_2, \ldots, X_n|A_n)$ and the value $f(X_1|A_1, X_2|A_2, \ldots, X_n|A_n)$ will be denoted simply by $(A_1, A_2, \ldots, A_n)$ and $f(A_1, A_2, \ldots, A_n)$, respectively. One can define value assignements for formulae as well.

Definition 5. Let $f, g \in P_k$. If the value of g does not exceed that of f (in any position of the truth-table), then we say that g implies f and write $g \to f$.

Definition 6. Formulae F and G are said to be equivalent if the corresponding functions f and g are equal. In this case we write $F=G$.

An easy computation gives

Lemma 1. Let $f(\mathbf{X}^n)=f(X_1, X_2, \ldots, X_n)$, $n \geqq 2$. Then for every $i=1, 2, \ldots,$

$$f(X_1, X_2, \ldots, X_i, \ldots, X_n) = \bigvee_{j=0}^{k-1} [f(X_1, \ldots, X_{i-1}, X_i|j, X_{i+1}, \ldots, X_n) \wedge {}^jX_i^j].$$

Remark. Below the conjunction will be denoted by $\cdot$ (sometimes it will be omitted) or, in the case of several variables, by Π, and the disjunction will be denoted by $+$ or Σ. The following lemma can easily be verified.

Lemma 2. Let $f(\mathbf{X}^n)=f(X_1, X_2, \ldots, X_n)$, $n \geqq 2$. Then the relation

$$f(X_1, X_2, \ldots, X_n) = \sum_{(a_1, a_2, \ldots, a_n)} {}^{a_1}X_1^{a_1} {}^{a_2}X_2^{a_2} \ldots {}^{a_n}X_n^{a_n} f(a_1, a_2, \ldots, a_n)$$

holds, where Σ is taken over all the possible ordered n-tuples, and $a_i \in \varepsilon_k$, $i=1, 2, \ldots, n$.

Definition 7. By a superposition of the k-valued logical functions $f(X_1, X_2, \ldots, X_i, \ldots, X_n)$ and $g(X_1, X_2, \ldots, X_n)$ we mean the function $f(X_1, X_2, \ldots, g(X_1, X_2, \ldots, X_n), \ldots, X_n)$ which is obtained by substituting the function g for the i-th argument X_i of f.

Definition 8. The set of functions $\{f_1, f_2, \ldots, f_l\}$ is called a basis-set for P_k if every elements of P_k can be expressed by X_i $(i=1, 2, \ldots, n)$ and the functions $f_1, f_2, \ldots, f_n$ applying superpositions finitely many times. It is customary to say that the elements of a basis set form a functionally complete function system.

By virtue of Lemma 2 we get that the system $\{0, 1, \ldots, k-1, {}^0X^0, {}^1X^1, \ldots, {}^{k-1}X^{k-1}, \min(X_1, X_2), \max(X_1, X_2)\}$ is complete in P_k.

Definition 9. The expression

$$\sum_{(a_1, a_2, \ldots, a_n)} {}^{a_1}X_1^{a_1}\, {}^{a_2}X_2^{a_2} \ldots {}^{a_n}X_n^{a_n} f(a_1, a_2, \ldots, a_n)$$

is called the full disjunctive normal form $F_v(X_1, X_2, \ldots, X_n)$ of the function f.

Since $a_i \in \varepsilon_k$, $i=1, 2, \ldots, n$, the number of all different n-tuples $(a_1, a_2, \ldots, a_n)$, is k^n. Denoting the value $f(a_1^{(j)}, a_2^{(j)}, \ldots, a_n^{(j)})$, concerning the j-th n-tuple (in a fixed ordering) $(a_1^{(j)}, a_2^{(j)}, \ldots, a_n^{(j)})$ by α_j and the corresponding conjunction

$${}^{a_1(j)}X_1^{b_1(j)\, a_2(j)}X_2^{a_2(j)} \ldots {}^{a_n(j)}X_n^{a_n(j)}$$

by E_j^n the full disjunctive normal form belonging to $f(\mathbf{X}^n)$ can be written in the form

$$F_V(\mathbf{X}^n) = \sum_{j=0}^{k^n-1} \alpha_j E_j^n.$$

$E_j^{(n)}$ is called a min term of n-variables. We will require some further formulae which can easily be verified.

$$ {}^aX^c + {}^dX^b = \begin{cases} {}^aX^b & \text{if,} \quad a \leqq d \leqq c \leqq b, \\ {}^dX^b & \text{if,} \quad d \leqq a \leqq c \leqq b, \\ \overline{{}^{c+1}X^{d-1}} & \text{if,} \quad a \leqq c < d \leqq b,\ a, b, c, d, X \in \varepsilon_k. \end{cases} \tag{1}$$

$$ {}^aX^c \cdot {}^dX^b = \begin{cases} 0 & \text{if,} \quad a \leqq c < d \leqq b, \\ {}^dX^c & \text{if,} \quad a \leqq d \leqq c \leqq b, \\ {}^dX^b & \text{if,} \quad a \leqq d \leqq b \leqq c,\ a, b, c, d, X \in \varepsilon_k. \end{cases} \tag{2}$$

$$ {}^a\overline{X^b} = {}^0X^{a-1} + {}^{b+1}X^{k-1}, \tag{3}$$

where

$${}^0X^{a-1} = 0 \quad \text{if} \quad a = 0, \quad {}^{b+1}X^{k-1} = 0 \quad \text{if} \quad b = k-1,\ a, b, X \in \varepsilon_k$$

$$ {}^0X^{k-1} = k-1. \tag{4}$$

$$ {}^aX^b + \overline{{}^aX^b} = k-1,\ a, b, X \in \varepsilon_k. \tag{5}$$

$$ \overline{\alpha\, {}^{a_1}X_1^{b_1}\, {}^{a_2}X_2^{b_2} \ldots {}^{a_n}X_n^{b_n}} = \bar{\alpha} + \overline{{}^{a_1}X_1^{b_1}} + \overline{{}^{a_2}X_2^{b_2}} + \ldots + \overline{{}^{a_n}X_n^{b_n}}. \tag{6.a}$$

$$ \overline{\alpha + {}^{a_1}X_1^{b_1} + {}^{a_2}X_2^{b_2} + \ldots + {}^{a_n}X_n^{b_n}} = \bar{\alpha}\, \overline{{}^{a_1}X_1^{b_1}}\, \overline{{}^{a_2}X_2^{b_2}} \ldots \overline{{}^{a_n}X_n^{b_n}}, \tag{6.b}$$

where

$$X_i, a_i, b_i \in \varepsilon_k, \quad i = 1, 2, \ldots, n.$$

Formulae (6a) and (6b) are the de Morgan's identities in the theory of multi-valued logic.

The full conjunctive normal form can be defined in a similar manner.

Definition 10. By the full conjunctive normal form of a k-valued function $f(\mathbf{X}^n)$ we mean the formula

$$F_\wedge(\mathbf{X}^n) = \prod_{j=0}^{k^n-1} (\alpha_j + \bar{E}_j^n),$$

where $\bar{E}_j^n$ can be obtained from E_j^n by the de Morgan's identities and denotes the so called max terms. Using the usual rules of the theory of two-valued logic the full normal forms can immediately be found from the truth-table. Every conjunction term and disjunction term of the full conjunctive and disjunctive normal forms contains the expression ${}^{a_1}X_1^{b_1}, {}^{a_2}X_2^{b_2}, \ldots, {}^{a_n}X_n^{b_n}$ of the variables $X_1, X_2, \ldots, X_n$.

The full disjunctive and conjunctive normal forms can be written in the following ways

$$F_\vee(\mathbf{X}^n) = F_1 + F_2 + \ldots + F_{k-1} = \sum_{i=1}^{k-1} F_i,$$

and

$$F_\wedge(\mathbf{X}^n) = F_1' F_2' \ldots F_{k-1}' = \prod_{j=1}^{k-1} F_j',$$

where F_i (F_j') is the sub-formula consisting only of min terms (max terms) which determine the i-th (j-th) value of the function.

Definition 11. Let F be a disjunctive normal form of $f \in P_k$, and let G be a conjunction term of F. We say that G is an implicant of f if $G \to f$. G is called prime implicant if, for every G' obtained by omitting any variable of G, $G' \not\to f$ holds.

Remark. The above defined min and max operations are mutually distributive (see identity III). Using this fact and the duality of the two operations we can treat the disjunction terms in a conjunctive normal form in the same way as we treat the conjunction terms in a disjunctive normal form. A normal form is called irredundant if the following properties hold:

(1) each of its terms is a primimplicant, and
(2) no expression obtained by omitting any term in the normal form implies the original function.

A normal form is called redundant if it is not irredundant.

2. Representation of formulae of functions.

The tree-construction procedure

We will work with a fixed order of our variables, which will be denoted by S. We agree that if we write $f(\mathbf{X}^n) = f(X_1, X_2, \ldots, X_n)$ then $S \equiv (X_1, X_2, \ldots, X_n)$. The simplification procedure we are going to discuss depends on S, therefore to some of the objects in the procedure we will affix S. By successive evaluation we mean successive evalution determined by S(i.e. we change first the first variable for logical values then the second one etc.)

Let $f(\mathbf{X}^n)$ and $S \equiv (X_1, X_2, \ldots, X_n)$ be given, and let

$$
\begin{aligned}
f(\mathbf{X}^n) = f(X_1, X_2, \ldots, X_n) &= f_{0,1} \\
f(0, X_2, \ldots X_n) &= f_{1,1} \\
f(1, X_2, \ldots X_n) &= f_{1,2} \\
\vdots \qquad\qquad & \vdots \\
f(i, X_2, \ldots, X_n) &= f_{1,i+1} \\
\vdots \qquad\qquad & \vdots \\
f(k-1, X_2, \ldots, X_n) &= f_{1,k} \\
f(0, 0, X_3, \ldots, X_n) &= f_{2,1} \\
\vdots \qquad\qquad & \vdots \\
f(0, k-1, X_3, \ldots, X_n) &= f_{2,k} \\
f(1, 0, X_3, \ldots, X_n) &= f_{2,k+1} \\
\vdots \qquad\qquad & \vdots \\
f(k-1, k-1, X_3, \ldots, X_n) &= f_{2,k^2} \\
\vdots \qquad\qquad & \vdots \\
f(k-1, k-1, \ldots, k-1, X_n) &= f_{n-1,k^{n-2}} \\
f(k-1, k-1, \ldots, k-1, 0) &= f_{n,1} \\
\vdots \qquad\qquad & \vdots \\
f(k-1, k-1, \ldots, k-1, k-1) &= f_{n,k^n}
\end{aligned}
$$

Using the results of Lemma 1, the following arrangement can be given (Fig. 1).

The functions $f_{0,1}, f_{1,1}, \ldots, f_{n,k^n}$ are called level-functions. Every function $f_{m,j}$ $(0 \leqq m < n,\ 1 \leqq j \leqq k^m)$ determines k new functions on the $(m+1)$-th level in the following way:

$$f_{m+1,jk-(i-(k-1))}(\ldots, X_j, \ldots) = f_{m,j}(\ldots, i, \ldots).$$

So there are k^{m+1} level-functions on the $(m+1)$-th level. The ${}^iX_j^i s$ $(i=0, 1, \ldots, k-1,$ $j=1, 2, \ldots, n)$ appearing at the edges of the tree above indicate that the variable X_j is replaced by the constant i. The functions $f_{n,1}$, being on the n-th level, are logical values.

This way we can associate a k-ary tree with every function $f(\mathbf{X}^n)$.

The tree which has just been obtained will be denoted by Φ_S (notice that the construction depends on the fixed order S of the variables). Since Φ_S contains all the possible level functions, Φ_s will be called complete.

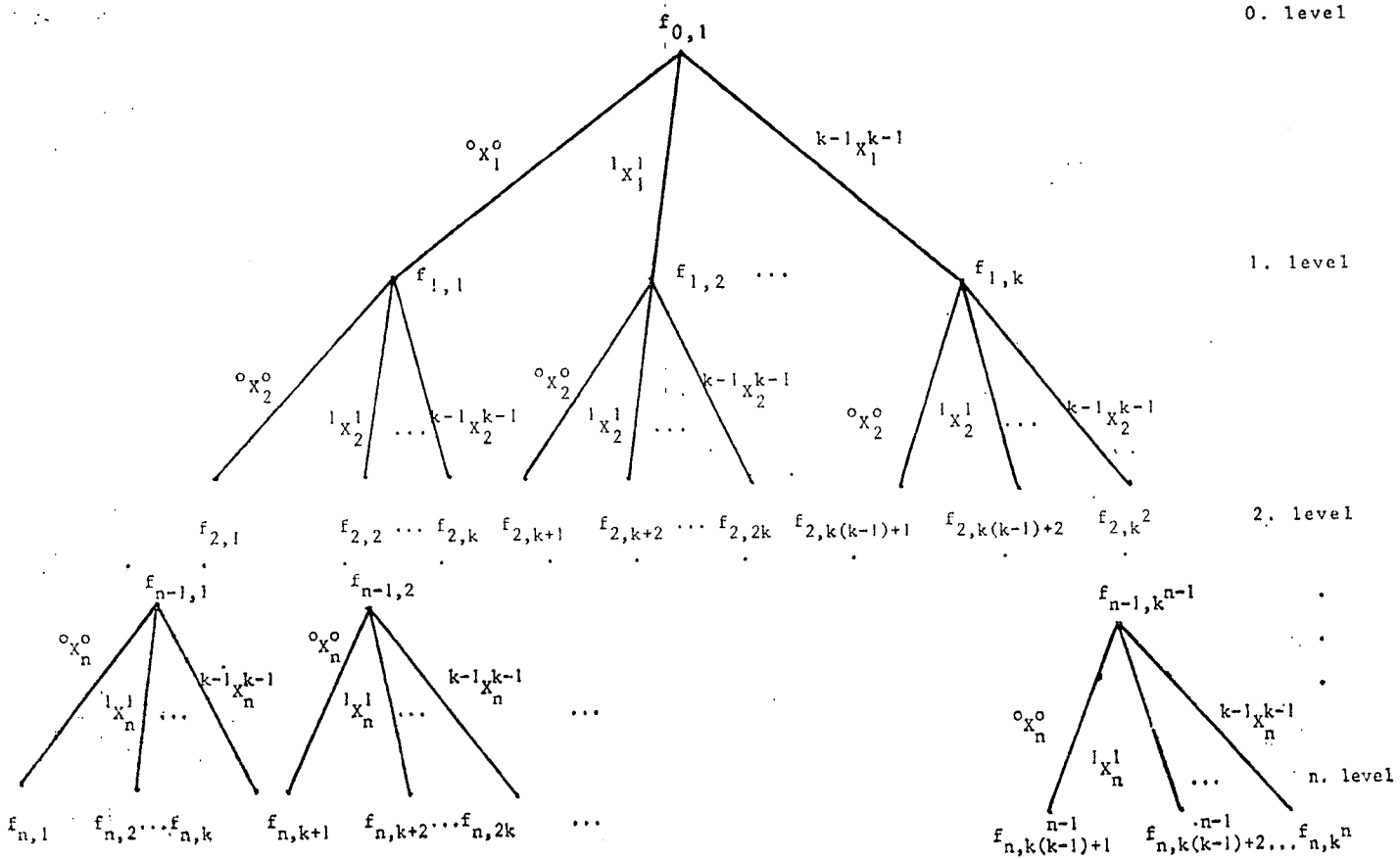

0. level
1. level
2. level
n. level
$f_{0,1}$
$^{o}x_1^{o}$
$^{1}x_1^{1}$
$^{k-1}x_1^{k-1}$
$f_{1,1}$
$f_{1,2}$ ...
$f_{1,k}$
$^{o}x_2^{o}$
$^{1}x_2^{1}$
$^{k-1}x_2^{k-1}$
$f_{2,1}$
$f_{2,2}$ ... $f_{2,k}$
$f_{2,k+1}$
$f_{2,k+2}$ ... $f_{2,2k}$
$f_{2,k(k-1)+1}$
$f_{2,k(k-1)+2}$
f_{2,k^2}
$f_{n-1,1}$
$f_{n-1,2}$
$f_{n-1,k^{n-1}}$
$^{o}x_n^{o}$
$^{1}x_n^{1}$
$^{k-1}x_n^{k-1}$
$f_{n,1}$
$f_{n,2}$ ... $f_{n,k}$
$f_{n,k+1}$
$f_{n,k+2}$ ... $f_{n,2k}$
$f_{n,k(k-1)+1}$
$f_{n,k(k-1)+2}$ ... f_{n,k^n}

Fig. 1

Notion of endpoint and path

By endpoints we mean the "leaves" of the tree (the vertices on the lowest, n-th level). Any sequence of edges joining the root with some endpoint will be called a path. Some more notations:

Let Φ_S be a tree belonging to $f(\mathbf{X}^n)$, and let $S \equiv (X_1, X_2, \ldots, X_n)$ be fixed. Suppose that the edge denoted by ${}^aX_{i+1}^a$ connects the level functions $f_{i,j}$ and $f_{i+1,1}$.

Below $f_{i,j}$ and $f_{i+1,j}$ will be called the start-point and the endpoint of the edge ${}^aX_{i+1}^a$, respectively. Obviously there is a one-to one correspondence between the evaluations of a function $f(\mathbf{X}^n)$ and the paths of the corresponding tree Φ_S. If we know the tree Φ_s corresponding to a function $f(\mathbf{X}^n)$ then it is easy to determine the $F_\vee(\mathbf{X}^n)$ full disjunctive and $F_\wedge(\mathbf{X}^n)$ full conjunctive normal forms of $f(\mathbf{X}^n)$. To obtain $F_\vee(\mathbf{X}^n)$ we have to take the conjunction of the variables ${}^jX_i^j$ along paths together with the logical value of the endpoint of the path and take the disjunction of all these expressions for every possible paths. If we interchange here "disjunction" and "conjunction" and "variable" for "negation of variable" we obtain $F(\mathbf{X}^n)$.

This method shows that the tree Φ_S is a representation of the formulae $F_\vee$ and $F_\wedge$. It can also be seen that Φ_S is equivalent to the truth-table of the function, the difference between them is that Φ_S can be obtained by successive evaluation while the truth-table is given by simultaneous evaluation.

Theorem 1. Let $f(\mathbf{X}^n) \in P_k$, $S \equiv (X_1, X_2, \ldots, X_n)$. Then the tree-construction procedure associates a uniquely determined k-ary tree to f.

Proof. The level function $f_{m+1,jk-(i-(k-1))}$ has fewer variables than $f_{m,j}$. Since $f_{0,1}$ contains a finite number of variables, the procedure must necessarily stop after the construction of a finite number of levels, which gives the existence of the tree. The unicity can be obtained from the equivalence of simultaneous and successive evaluations.

Definition 12. Any function f with domain $D(f) \subset \varepsilon_k^n$ is called a partially defined function. Those places where f is not defined will be marked by $(*)$ in the truth-table and at the "leafs" of the tree.

In the process of simplification we can assign any value to these places, which, in certain cases, yields a simpler representation.

3. The simplification procedure

Let $f(\mathbf{X}^n) \in P_k$ and let S be fixed. In order to construct an irredundant equivalent of $f(\mathbf{X}^n)$ first we construct the tree Φ_S and choose that subtrees Φ_s^t $(t=0, 1, \ldots, k-1)$ of Φ_S which consists of those paths of Φ_S that have t at their end.

Definition 13. Those points of Φ_s^t $(t=0, 1, \ldots, k-1)$ from which exactly k edges start will be called complete branching points, and the k edges starting from such a point will be called a complete edge-system. A complete branching point of a subtree is called m-multiple complete branching point if the subtree has altogether m total branching points with the same complete edge-system as the given point (more precisely the variables attached to the complete edge systems must be the same).

Let p be an arbitrary path of Φ_S^t $(t=0, 1, \ldots, k-1)$ and let n be the number of its edges.

Let

$$*X_{n_j}^* = \begin{cases} {}^0X_{n_j}^0 & \text{if } {}^0X_{n_j}^0 \text{ belongs to } p, \\ {}^1X_{n_j}^1 & \text{if } {}^1X_{n_j}^1 \text{ belongs to } p, \\ \vdots & \vdots \quad \vdots \quad \vdots \quad \vdots \\ {}^{k-1}X_{n_j}^{k-1} & \text{if } {}^{k-1}X_{n_j}^{k-1} \text{ belongs to } p, \end{cases} \quad (1 \leqq n_{j-1} < n_j \leqq n,\ 2 \leqq j \leqq m)$$

and

$$\{*X_{n_1}^*, *X_{n_2}^*, \ldots, *X_{n_m}^*\} \equiv \bar{X} = \bar{X}\{p, n_j, m\}$$

some edges of p.

Those edges of p (if there is any) which do not belong to $\bar{X}$ will be called connecting sequences of $\bar{X}$ (relative to p) and will be denoted $\varkappa = \{\varkappa_1, \varkappa_2, \ldots, \varkappa_S\}$.

Let Φ_S^t be given, and let p be a path of Φ_S^t. A subtree $\Phi_S'^t$ of Φ_S^t will be called maximally simplifiable subtree of order m (below briefly MSST) if

(1) $\Phi_S'^t$ contains p,

(2) there exists such an edge set $\bar{X}=\bar{X}\{p, n_j, m\}$ $(1 \leqq n_{j-1} < n_j \leqq n,\ 2 \leqq j \leqq m)$ of p taken in the fixed order determined by S that the edges marked by $*X_{n_{i+1}}^*$ $(i=0, 1, \ldots, m-1)$ belong to k^i-multiple total edge systems of $\Phi_S'^t$, and if p' is any other path of $\Phi_S'^t$ then the connecting sequences of $\bar{X}=\bar{X}\{p, n_j, m\}$ and $\bar{X}'=\bar{X}'\{p', n_j, m\}$ relative to p and p' are the same (more precisely, are marked in order with the same variables ${}^uX_v^u$).

(3) There exist no subtree $\Phi_S''^t$ of $\Phi_S'^t$ that has properties (1) and (2) and which has more than m total branching points.

The structure of an MSST of order m is shown on Fig. 2.

Remark. $\varkappa_i$ is the sequence of edges between $*X_{n-1}^*$ and $*X_{n_i}^*$ in the order determined by S. If $n_i=n_{i-1}+1$ then $\varkappa_i$ is empty. If $m=0$ then $\Phi_S'^t=p$. It is obvious that if a tree Φ_S^t and its path p are given then there exists at least one MSST containing p.

Theorem 2. Let $f(\mathbf{X}^n) \in P_k$, Φ_S^t $(t=0, 1, \ldots, k-1)$ a tree belonging to a fixed S, p a path of Φ_S^t and $\mathfrak{M}$ an MSST of p. Let the n-term conjunction of variables along the paths of $\mathfrak{M}$ be: $p_1, p_2, \ldots, p_{k^l}$ $(1 \leqq l \leqq n)$, and the (variables at the) connecting sequence $\varkappa_1, \varkappa_2, \ldots, \varkappa_m$. Then

$$\sum_{j=1}^{k^l} p_j = \prod_{i=1} \varkappa_i$$

holds.

Proof. $\mathfrak{M}$ contains k^l paths, so there are $k^0, k^1, \ldots, k^{m-1}$ total branchings on the different levels. In other words the formula F_t corresponding to Φ_S^t does not depend on the variables appearing in the total branchings because it takes the value t independently of these variables, so they can be omitted.

This theorem shows that every $\mathfrak{M}$ yields one term. The term which is obtained by the method above is called the simplified formula of $\mathfrak{M}$. The disjunction of such simplifications of MSST's is the simplified formula of the function.

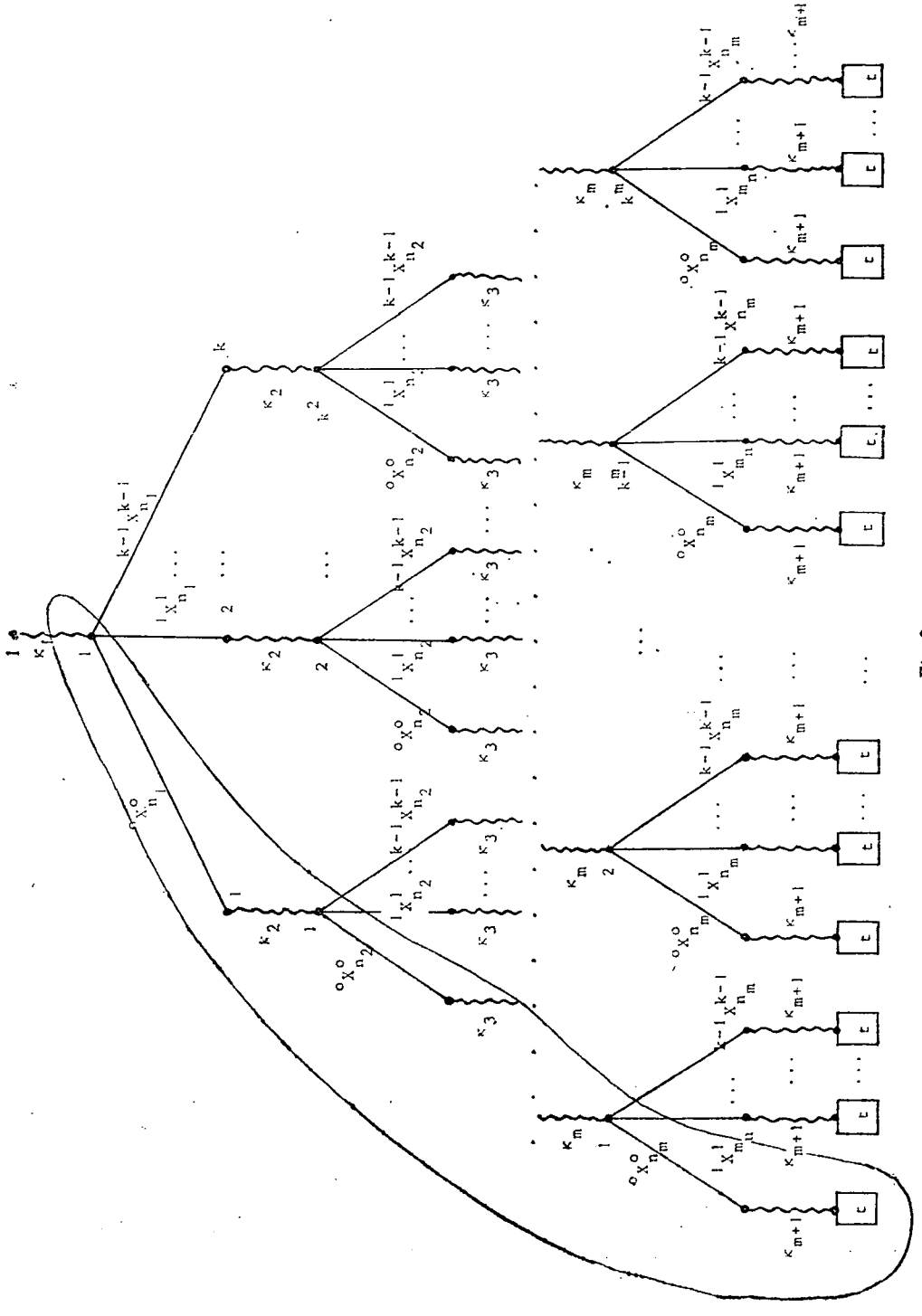


Fig. 2

4. Irredundant coverings

Definition 14. A set of MSST-s of a tree Φ_S^t $(t=0, 1, \ldots, k-1)$ is called a covering if each path of the tree belongs to at least one of the MSST-s of the set.

A covering is called irredundant if any MSST in it contains at least one path belonging only to this MSST.

Theorem 3. Let Φ_S^t represent the disjunctive normal form of an $f(\mathbf{X}^n) \in P_s$, $(t=0, 1, \ldots, k-1)$, together with one of its irredundant coverings. Let $\tilde{F}_V$ denote the disjunction of simplified formulae obtained from the elements of the set of MSST-s giving the irredundant covering in question. Then $\tilde{F}_V$ is irredundant.

Proof. Suppose that $\tilde{F}_V$ is redundant. Then there exist two cases.

(1) some disjunction term of $\tilde{F}_V$ can be omitted;

(2) at least one variable can be omitted from some conjunction term of $\tilde{F}_V$.

First suppose that a term $F^{(1)}$ of $\tilde{F}_V$ can be omitted. Since every MSST gives only one conjunction term, omitting this is equivalent to omitting the MSST from the covering, but taking into account the irredundancy, this is impossible.

Secondly we note that, if an $F^{(i)}$ can be replaced by an $F'^{(i)}$ obtained from $F^{(i)}$ by omitting some variables, then the MSST giving $F'^{(i)}$ contains the MSST which gave $F^{(i)}$, but this contradict the definition of MSST.

Remark. Theorem 3 is formulated for full disjunctive normal forms, but because of the principle of duality it is true for full conjunctive normal forms as well.

5. Simplifiable paths, simplification algorithm

Definition 15. Let Φ_S^t $(t=0, 1, \ldots, k-1)$ be given, and take a path p of Φ_S^t.

— p is called singular if the MSST coincides with p.

— p is called simply covered if p is covered by one and only one MSST.

— p is multiply covered if it is covered by at least two MSST-s.

Theorem 4. Let $f(\mathbf{X}^n) \in P_k$ be given by either its disjunctive or conjunctive full normal forms. If f is given by its full disjunctive normal form F_V and some max term E_j^n is simultaneously represented by formulae $F_{l-m}, F_{l-m+1}, \ldots, F_{l-m+i}$, then

$$\min(F_{l-m}, F_{l-m+1}, \ldots, F_{l-m+i}) = F_{l-m}. \tag{1}$$

If f is given by the full conjunctive normal form F_Λ and some max term $\bar{E}_j^n$ is simultaneously represented by formulae $F_{l-m}, F_{l-m+1}, \ldots, F_{l-m+i}$, then

$$\max(F_{l-m}, F_{l-m+1}, \ldots, F_{l-m+i}) = F_{l-m+i}. \tag{2}$$

The statement can easily be proved taking into account the definitions of the min and max operators.

Formula (1) means that the simplification procedure of a function f (or tree Φ_S which is representing the function and is written from the disjunctive normal form) value (for example in case $\varepsilon_k = \{0, 1, \ldots, k-1\}$ with Φ_S^{k-1}). After the first step of the

simplification the endpoints marked with $(k-1)$ can be considered of $(*)$-value, that is undefined, in the tree Φ_S. Let us introduce the following notations:

(i) $$\Phi_S^{k-2,*} = \Phi_S^{k-1} \cup \Phi_S^{k-2}$$

and

(ii) $$\Phi_S^{k-i,*} = \Phi_S^{k-(i-1),*} \cup \Phi_S^{k-i} \quad (i > 2)$$

where $l=k-j$ and $(*)$ is written on the places $j<i$. By virtue of Theorem 4, there are subtrees which may give more favourable conditions for simplifiction.

On the other hand relation (2) shows in case of tree of functions given by full conjunctive normal form that simplification has to be started with the simplification of that subtree determined by the path with smallest logical value and we have to apply the method above. Below the procedure will be shown only for functions given by their full disjunctive normal forms. The case of full conjunctive normal forms can be treated in a similar way.

Now we can give the simplification procedure.

6. Simplification algorithm for representations of irredundant formulae

(1) Let $i=1$. Mark the paths with endpoint $t=k-1$ in the tree Φ_S (that is we start from the subtree Φ_S^{k-1}). If in the tree Φ_S originally there are endpoints marked with $(*)$, then we begin with $\Phi_S^{k-1} \cup \Phi_S^*$.

We choose a path and an MSST containing it. We take a record of the simplified formulae corresponding to this MSST and mark the paths in it.

(2) We choose an unmarked path and determine an MSST covering it, preferably with unmarked endpoints (this will speed up the algorithm). This way such an MSST is chosen which is necessary for an irredundant covering. The simplified formula belonging to the MSST we have just obtained will be taken record of and the so far unmarked paths of the MSST will be marked.

Repeat step 3 until we can find unmarked paths in Φ_S^{k-1}.

If there is no unmarked path, then let $i=i+1$. If $i<k$, then consider the subtree $\Phi_S^{k-1,*}$ and carry out the above steps (1), (2), (3). If $i=k$ the algorithm is over.

Finally the simplified formula of the function $f(X^n)$ can be determined as follows:

Let

$$F_{s,1}^{k-1}, F_{s,2}^{k-1}, \ldots, F_{s,i_1}^{k-1}$$

$$F_{s,1}^{k-2}, F_{s,2}^{k-2}, \ldots, F_{s,i_2}^{k-2}$$

$$\vdots \qquad \vdots \qquad \vdots$$

$$F_{s,1}^{1}, F_{s,2}^{1}, \ldots, F_{s,i_{k-1}}^{1},$$

denote the simplified formulae obtained from the subtrees $\Phi_S^{k-1}, \Phi_S^{k-2,*}, \ldots, \Phi_S^{1,*}$;

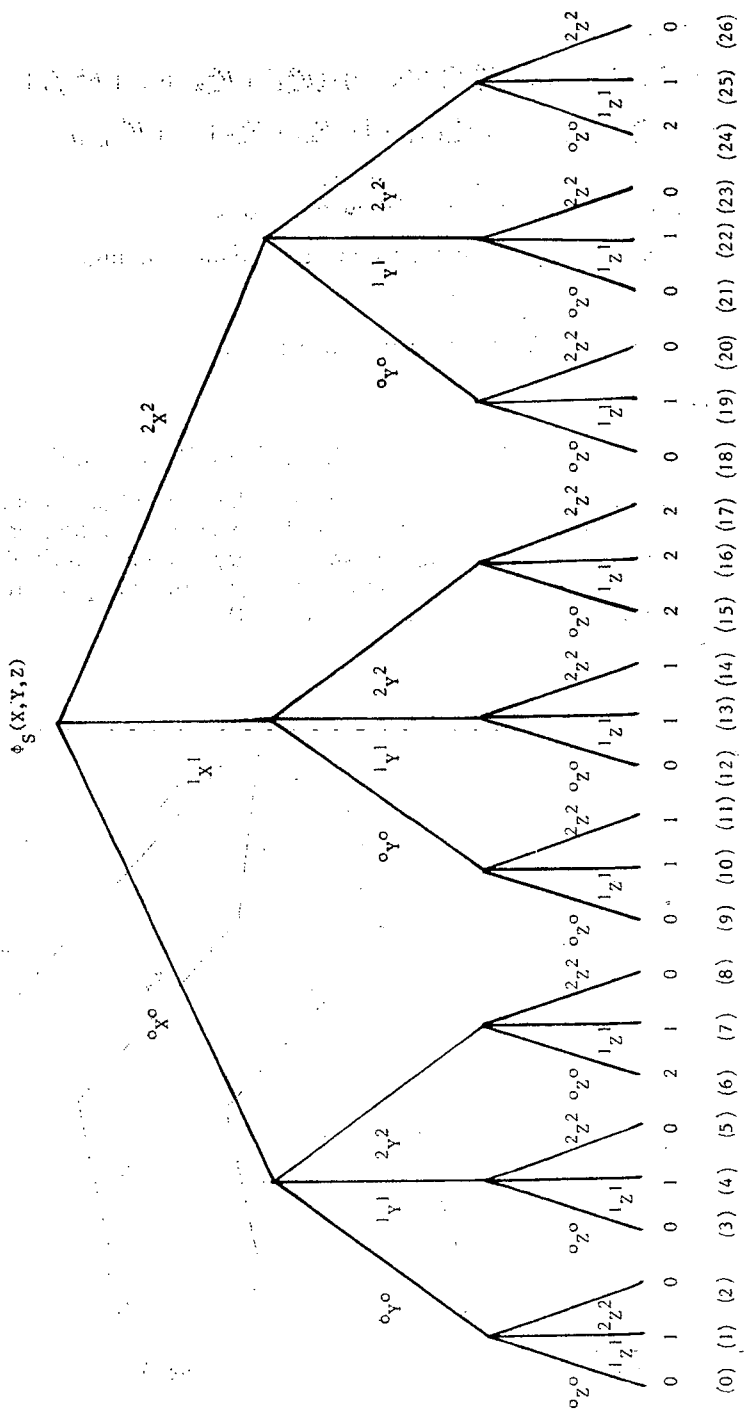

$\Phi_S(X,Y,Z)$
$2_X{}^2$
$1_X{}^1$
$0_X{}^0$
$2_Y{}^2$
$1_Y{}^1$
$0_Y{}^0$
$2_Z{}^2$
$1_Z{}^1$
$0_Z{}^0$
(0) (1) (2) (3) (4) (5) (6) (7) (8) (9) (10) (11) (12) (13) (14) (15) (16) (17) (18) (19) (20) (21) (22) (23) (24) (25) (26)

Fig. 3

respectively. Then the formula:

$$(k-1)\cdot(F_{s,1}^{k-1}+F_{s,2}^{k-1}+\ldots+F_{s,i_1}^{k-1})+(k-2)\cdot(F_{s,1}^{k-2}+F_{s,2}^{k-2}+\ldots+F_{s,i_2}^{k-2})+\ldots$$
$$+2\cdot(F_{s,1}^{2}+F_{s,2}^{2}+\ldots+F_{s,k-2}^{2})+1\cdot(F_{s,1}^{1}+F_{s,2}^{1}+\ldots+F_{s,i_{k-1}}^{1})$$

corresponds to an irredundant covering of Φ_S.

All these can be summarised in the following theorem.

Theorem 5. Every tree Φ_S has at least one irredundant covering.

7. Some demonstrative examples

I. Consider the function

$$f^3(X, Y, Z) = 1\Sigma(1, 4, 7, 10, 11, 13, 14, 19, 22, 25)+2\Sigma(6, 15, 16, 17, 24)$$

given by its full disjuntive normal form (here we use the conventional notation of binary logic; only the numbers in brackets should be considered as numbers in the number system with base k instead of 2). We will simplify the function $f^3(X, Y, Z)$. Let $S\equiv(X, Y, Z)$ be the order of evaluation. Fig. 3 gives the complete tree of f^3 With $k=2$ pick the tree Φ_S^2 and let us analyse it (Fig. 4).

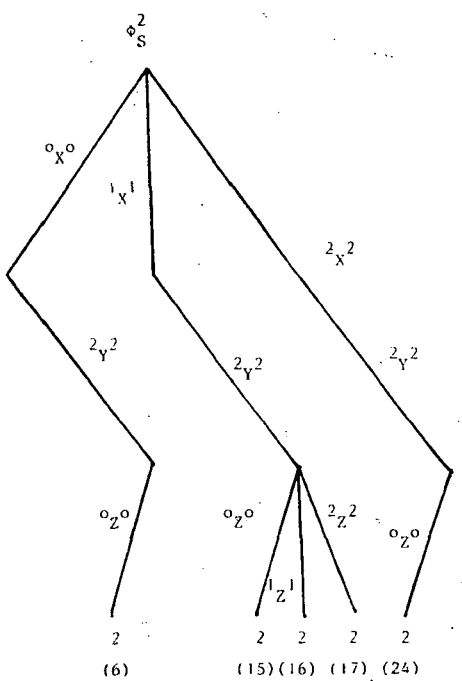


Fig. 4

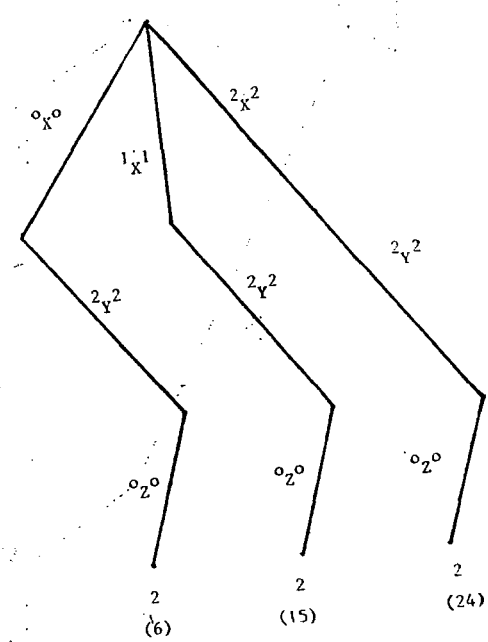


Fig. 5

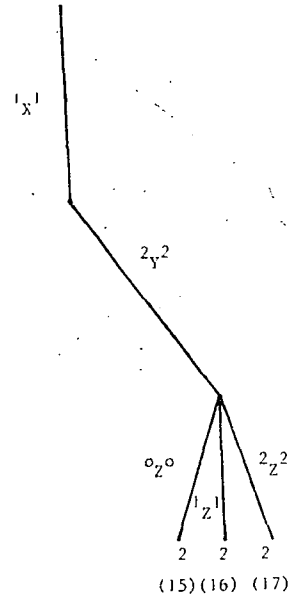


Fig. 6

Let us investigate the paths of this tree moving from the left to the right

1. There is no singular path.

2. Simply covered paths: (6), (16), (17). The MSST belonging to (6) is (6—15—24) (Fig. 5). The next path is (16) and the corresponding MSST is (15—16—17) (Fig. 6).

The simplified formulae

$$^{2}Y^{2}.{}^{0}Z^{0}$$

$$^{1}X^{1}\,{}^{2}Y^{2}$$

3. There is no more unmarked path.

We write $(*)$ instead of 2 and consider $\Phi_S^{1,*}$ with $k=1$. (Fig. 7)

1. There is no singular path.

2. Simply covered paths are:

(1) and the corresponding MSST is (1—4—7—10—13—16—19—22—25) (Fig. 8), (11) and the MSST is (11—14—17) (Fig. 9).

The simplified formulae are:

$$^{1}Z^{1}$$

$$^{1}X^{1}.{}^{2}Z^{2}$$

The simplified irredundant formula is:

$$2(^{1}X^{1}\,{}^{2}Y^{2}+{}^{2}Y^{2}\,{}^{0}Z^{0})+1(^{1}Z^{1}+{}^{1}X^{1}\,{}^{2}Z^{2}).$$

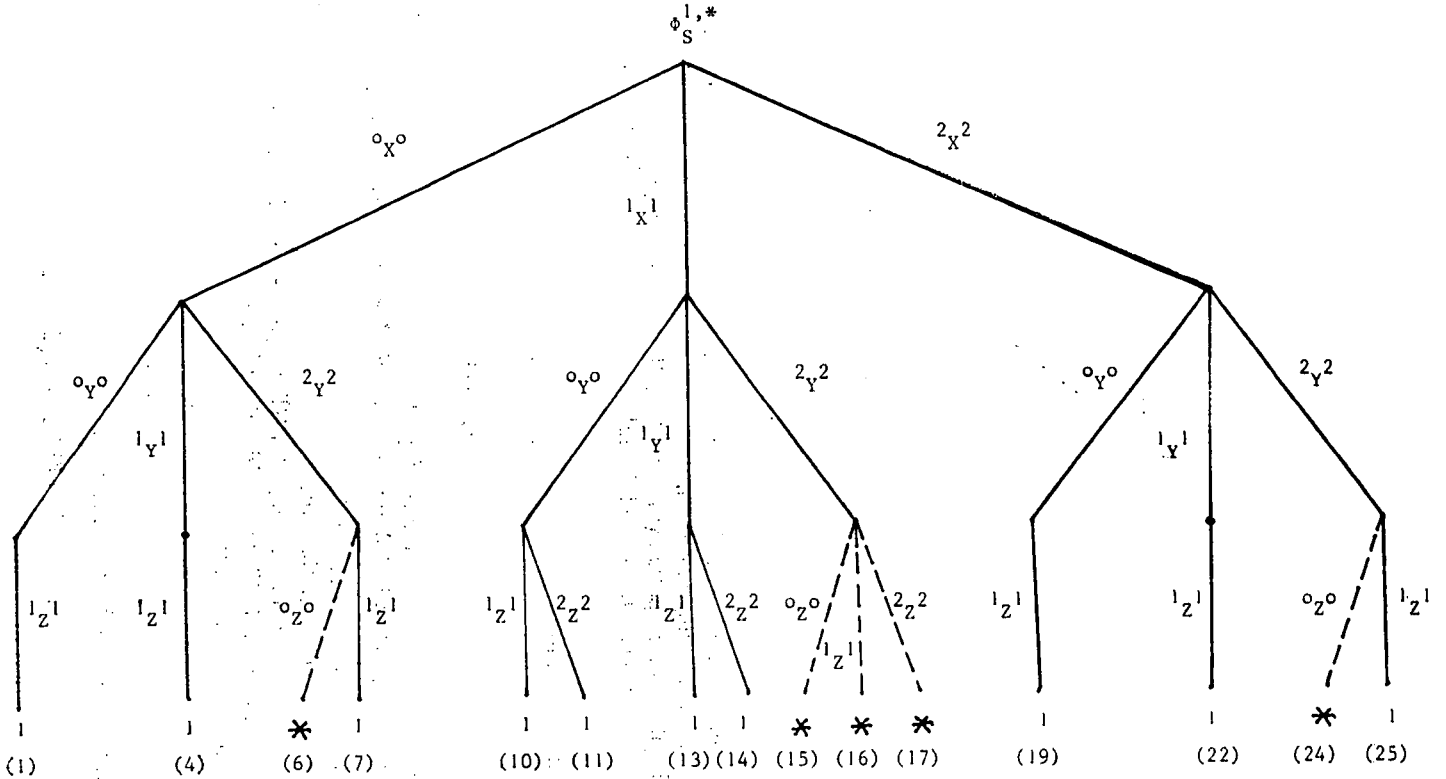

$\phi_S^{1,*}$
$^{o}X^{o}$
$^{1}X^{1}$
$^{2}X^{2}$
$^{o}Y^{o}$
$^{1}Y^{1}$
$^{2}Y^{2}$
$^{1}Z^{1}$
$^{2}Z^{2}$
$^{o}Z^{o}$
(1) (4) (6) (7) (10) (11) (13) (14) (15) (16) (17) (19) (22) (24) (25)

Fig. 7

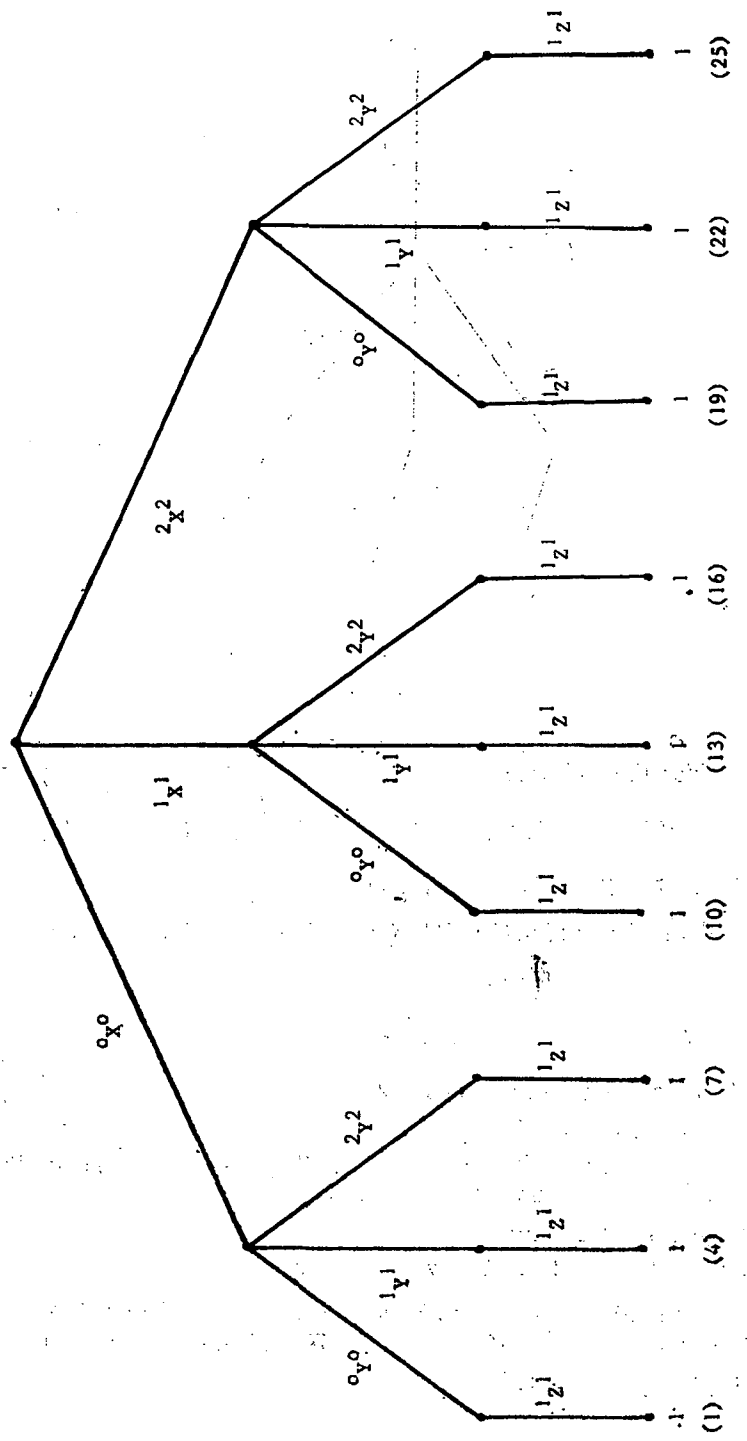


Fig. 8

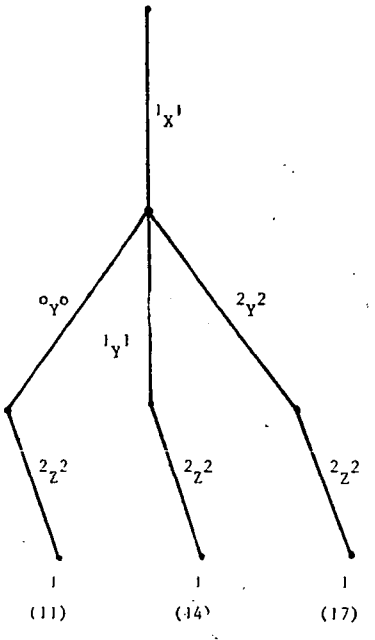


Fig. 9

II. Consider the following function

$$f^4(X, y) = 1\Sigma(3, 4, 6, 7)+2\Sigma(1, 5, 13)+3\Sigma(9, 10, 11, 14)$$

and let $S\equiv(X, Y)$. Simplify this function.

1. Singular paths are: (9), (10), (11), (14) and the formulae belonging to these are:

$$^{2}X^{2}\,^{1}Y^{1},\ ^{2}X^{2}\,^{2}Y^{2},\ ^{2}X^{2}\,^{3}Y^{3},\ ^{3}X^{3}\,^{2}Y^{2}.$$

2. There is no more unmarked path.

We write $(*)$ instead of 3 and let $k=2$.

1. There is no singular path.
2. Simply covered path is: (1) and the MSST is (1—5—9—13) (Fig. 13)
3. There is no more unmarked path.

The simplified formula is $^{1}Y^{1}$.

We write instead of 2 and 3 now $(*)$ and let $k=1$.

1. Singular path is: (3) and the corresponding formula is: $^{0}X^{0}\,^{3}Y^{3}$
2. Simply covered path is (4) and the MSST is (4—5—6—7) (Fig. 15).

The simplified formula is: $^{1}X^{1}$

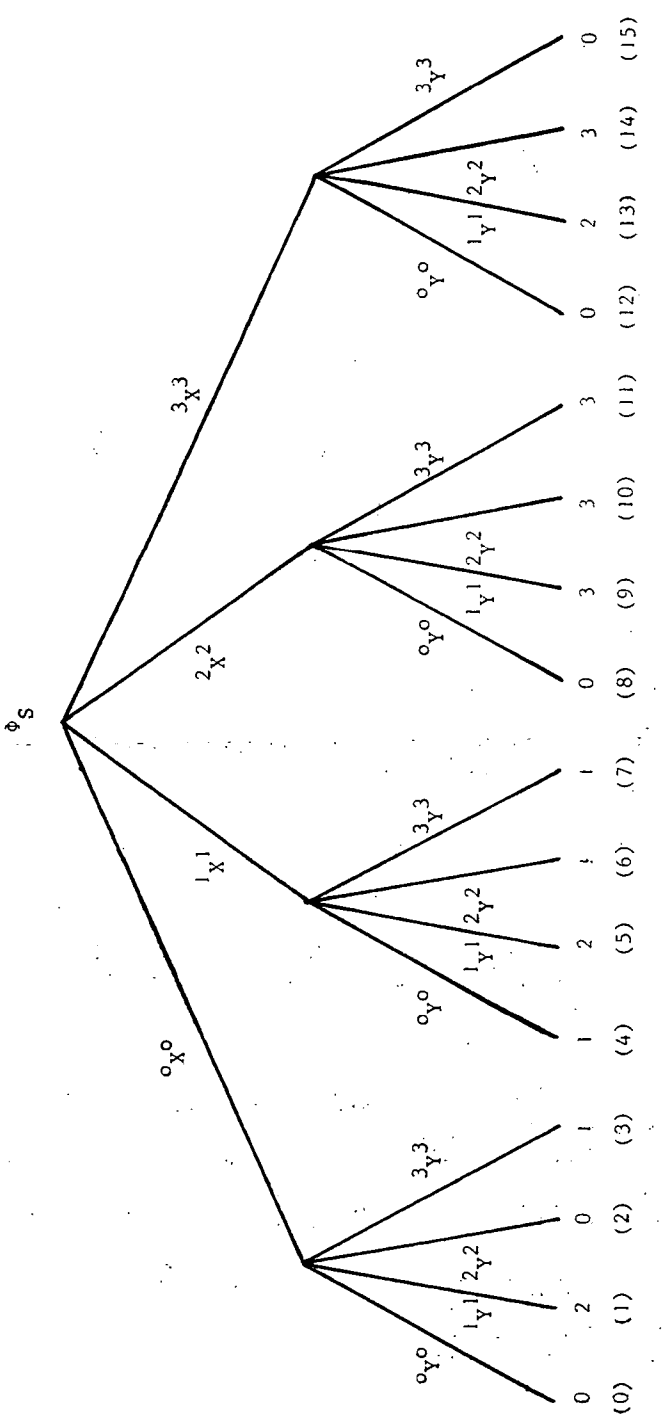

ϕ_S
0_X^0
1_X^1
2_X^2
3_X^3
0_Y^0
1_Y^1
2_Y^2
3_Y^3
0 2 0 1 1 2 1 0 0 3 3 0 0 2 3 0
(0) (1) (2) (3) (4) (5) (6) (7) (8) (9) (10) (11) (12) (13) (14) (15)

Fig. 10

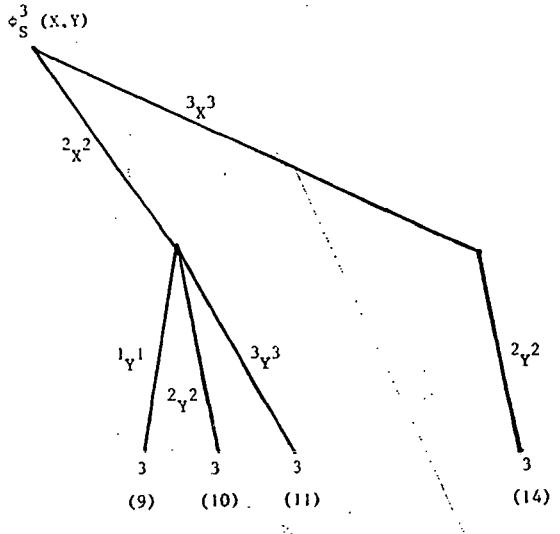


Fig. 11

3. The simplified function is:

$$3({}^2X^2\,{}^1Y^1+{}^2X^2\,{}^2Y^2+{}^2X^2\,{}^3Y^3+{}^3X^3\,{}^2Y^2)+2^1Y^1+1({}^0X^0\,{}^3Y^3+{}^1X^1).$$

Remark. The irredundant formula we have just obtained can be transformed by virtue of indentities treated above.

For example:

$$3({}^2X^2({}^1Y^1+{}^3Y^3)+{}^2Y^2({}^2X^2+{}^3X^3))+2^1Y^1+1({}^0X^0\,{}^3Y^3+{}^1X^1)=$$

$$=3({}^2X^2\,{}^1Y^3+{}^2X^3\,{}^2Y^2)+2^1Y^1+1({}^0X^0\,{}^3Y^3+{}^1X^1).$$

III. Let $f^3(X, Y, Z)$ be given by its truth-table (Fig. 16). Simplify this function Let $S\equiv(X, Y, Z)\cdot\Phi_S(X, Y, Z)$ is sketched in Fig. 17. For the endpoints marked with $k=2$ and $*$ we have:

1. There is no singular path.
2. Simply covered paths are:

(i) (13) and the corresponding MSST is (4—13—22) (marked with +) (Fig. 17). The simplified formula is: ${}^1Y^1\,{}^1Z^1$;

(ii) (21), the MSST is (21—22—23) (marked with o) and the simplified formula is ${}^2X^2\,{}^1Y^1$.

(iii) (24), the MSST is (18—21—24) (marked with "=") and the simplified formula is ${}^2X^2\,{}^0Z^0$.

3. There is no more unmarked path with endpoint 2. Consider now the subtree with endpoints $k=1$, $2\equiv *$ and $*$

1. There is no singular path.
2. Simply covered paths are:

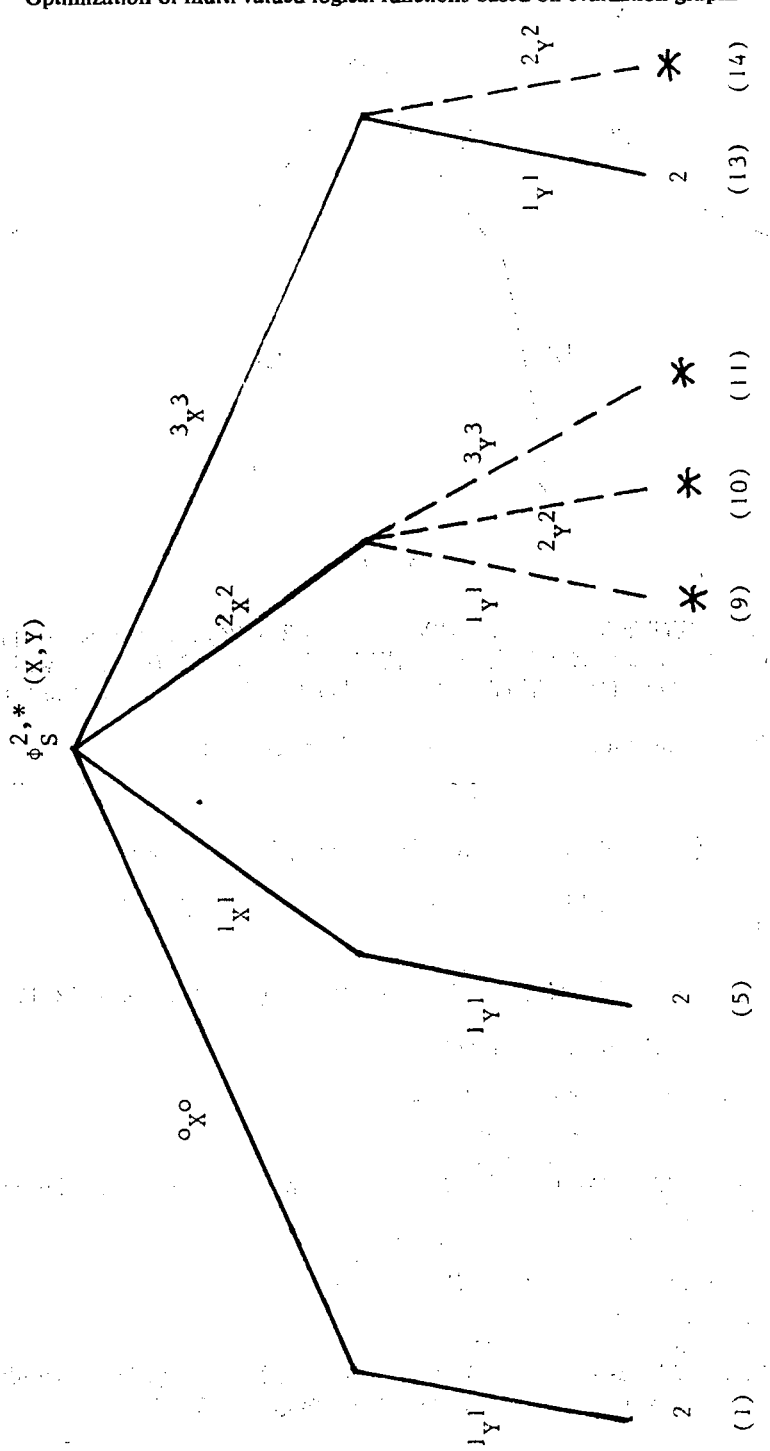

$\phi_S^{2,*}(X,Y)$
3_X^3
2_X^2
1_X^1
0_X^0
3_Y^3
2_Y^2
1_Y^1
2_Y^2
1_Y^1
1_Y^1
1_Y^1
2
2
2
(1)
(5)
(9)
(10)
(11)
(13)
(14)

Fig. 12

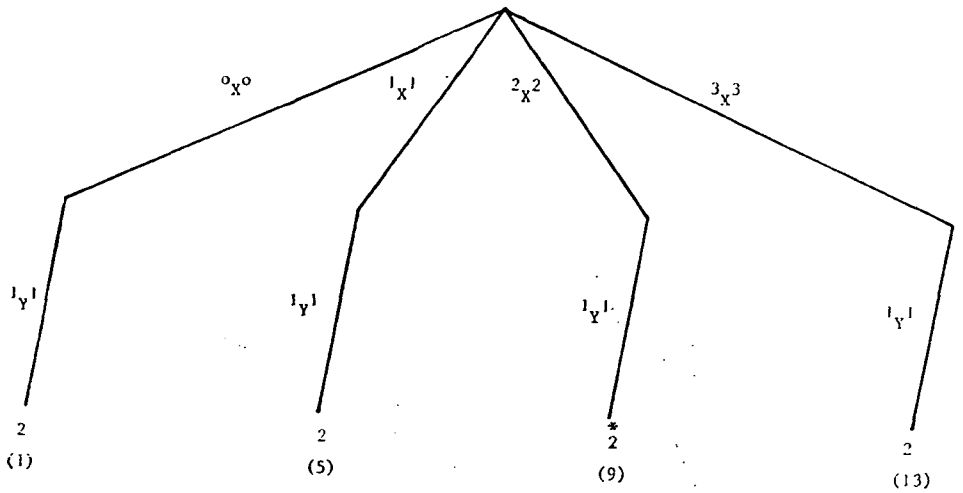


Fig. 13

(0), the MSST is (0—3—6—9—12—15—18—21—24) (marked with ⊡) and the simplified formula is: ${}^{0}Z^{0}$; (10), the MSST is (9—10—11) (marked with •), the formula is ${}^{1}X^{1}\,{}^{0}Y^{0}$; (16) MSST: (15—16—17) (marked with X), the formula is: ${}^{1}X^{1}\,{}^{2}Y^{2}$.

3. There is no more unmarked path with endpoint 1.

The simplified formula of the function is:

$$2({}^{1}Y^{1}\,{}^{1}Z^{1}+{}^{2}X^{2}\,{}^{1}Y^{1}+{}^{2}X^{2}\,{}^{0}Z^{0})+1({}^{0}Z^{0}+{}^{1}X^{1}\,{}^{0}Y^{0}+{}^{1}X^{1}\,{}^{2}Y^{2})=$$

$$=2({}^{1}Y^{1}\,{}^{1}Z^{1}+{}^{2}X^{2}\,{}^{1}Y^{1}+{}^{2}X^{2}\,{}^{0}Z^{0})+1({}^{0}Z^{0}+{}^{1}X^{1}\,{}^{1}Y^{2}).$$

IV. Let

$$f^{4}(X, Y) = 1\Sigma(5, 8, 9, 11)+2\Sigma(2, 6, 10)+3\Sigma(13, 14)+*\Sigma(1, 12, 15)$$

and $S\equiv(X, Y)$. Simplify this function

For the paths with endpoints $k=3$ and $*$:

1. There is no singular path,
2. Simply covered paths are:
   (13) MSST: (12—13—14—15) (marked with ∅) (Fig. 18) the simplified formula: ${}^{3}X^{3}$.
3. There is no more path with endpoint $k=3$.

For the paths with endpoints $k=2$, $3\equiv *$ and $*$:

1. There is no singular path.
2. Simply covered paths are:
   (2) MSST: (2—6—10—14) (marked with +) the simplified formula is ${}^{2}Y^{2}$.
3. There is no more unmarked path with endpoint $k=2$.

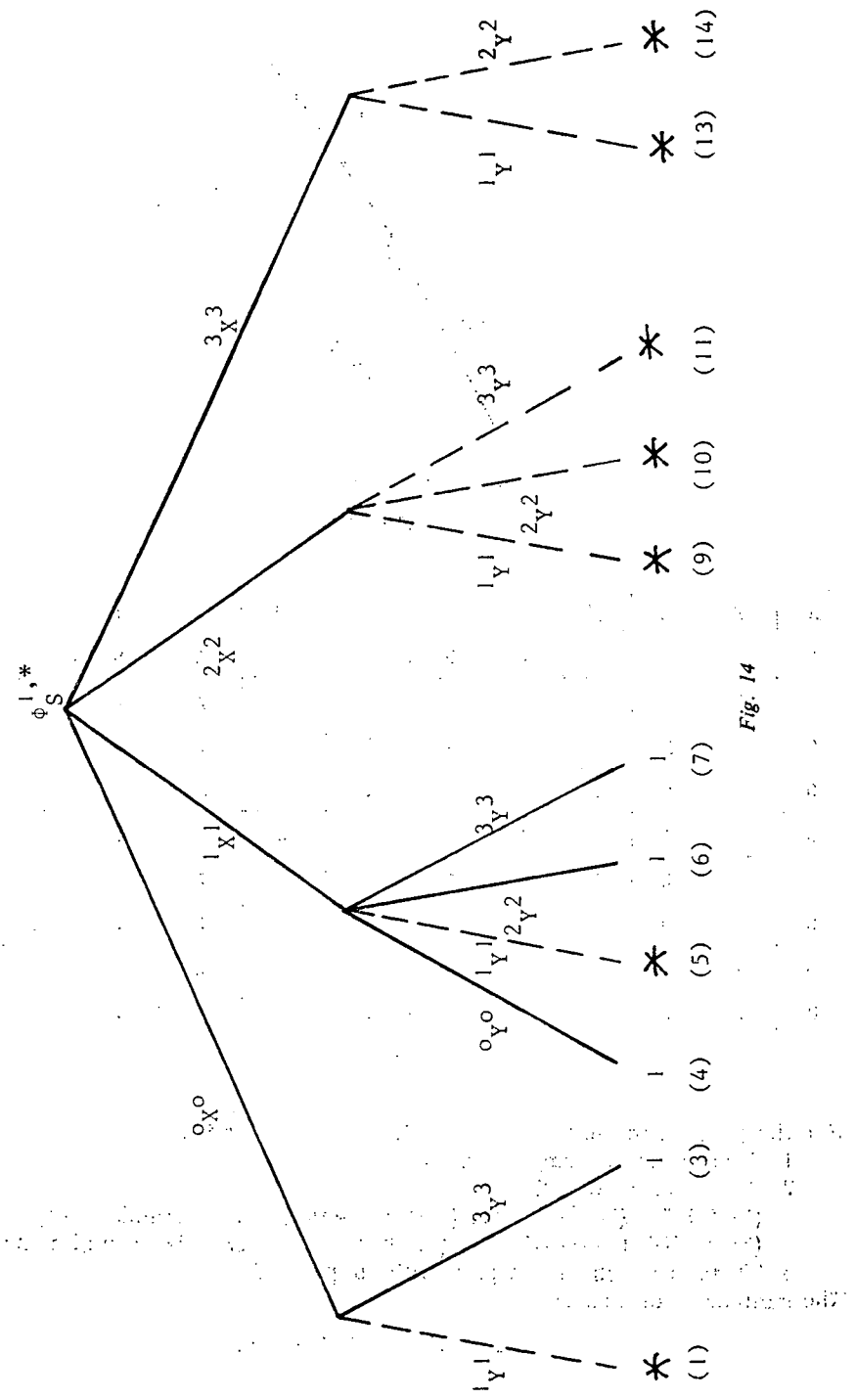

$\phi_S^{1,*}$
3_X^3
2_X^2
1_X^1
0_X^0
2_Y^2
1_Y^1
3_Y^3
2_Y^2
1_Y^1
3_Y^3
2_Y^2
1_Y^1
0_Y^0
3_Y^3
1_Y^1
1
1
1
1
(14)
(13)
(11)
(10)
(9)
(7)
(6)
(5)
(4)
(3)
(1)

Fig. 14

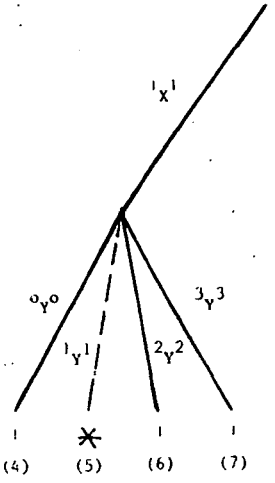


Fig. 15

X	Y	Z	f^3	X	Y	Z	f^3	X	Y	Z	f^3
0	0	0	1	1	0	0	1	2	0	0	*
0	0	1	0	1	0	1	1	2	0	1	0
0	0	2	*	1	0	2	*	2	0	2	0
0	1	0	1	1	1	0	1	2	1	0	2
0	1	1	*	1	1	1	2	2	1	1	2
0	1	2	0	1	1	2	0	2	1	2	2
0	2	0	1	1	2	0	1	2	2	0	2
0	2	1	0	1	2	1	1	2	2	1	0
0	2	2	*	1	2	2	1	2	2	2	0

Fig. 16

For the paths with endpoints $k=1$, $2\equiv *$, $3\equiv *$, and $*$.

1. There exists no singular path.
2. Simply covered paths:
   (5) MSST: (1—5—9—13) (marked with □) the formula: $^{1}Y^{1}$;
   (8) MSST: (8—9—10—11) (marked with X) and the formula: $^{2}X^{2}$.
3. There is no unmarked path with endpoint $k=1$.

The simplified formula is:

$$3^{3}X^{3}+2^{2}Y^{2}+1(^{2}X^{2}+^{1}Y^{1}).$$

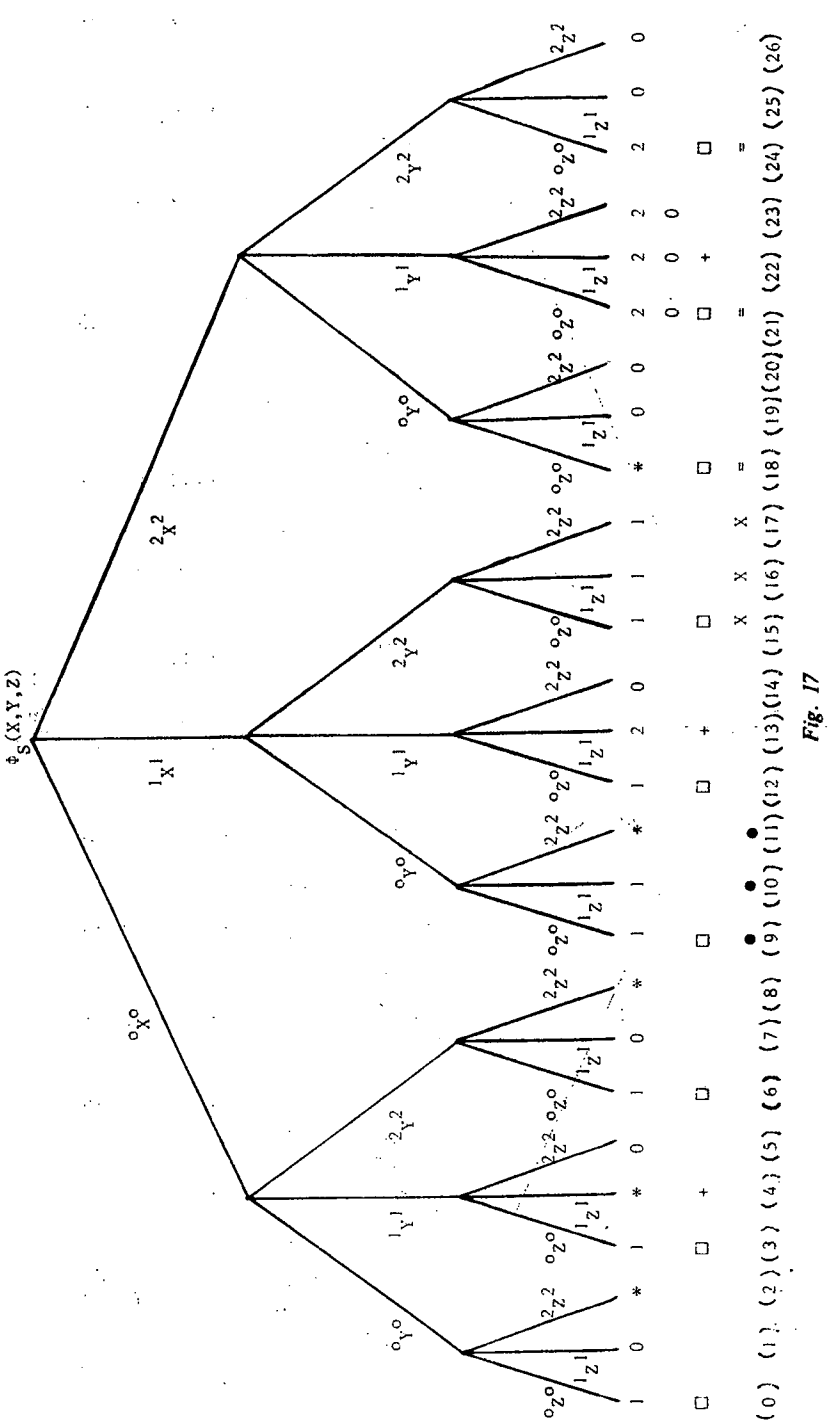

$\Phi_S(X,Y,Z)$
$_0X^0$
$_1X^1$
$_2X^2$
$_0Y^0$
$_1Y^1$
$_2Y^2$
$_0Z^0$
$_1Z^1$
$_2Z^2$
(0) (1) (2) (3) (4) (5) (6) (7) (8) (9) (10) (11) (12) (13) (14) (15) (16) (17) (18) (19) (20) (21) (22) (23) (24) (25) (26)

Fig. 17

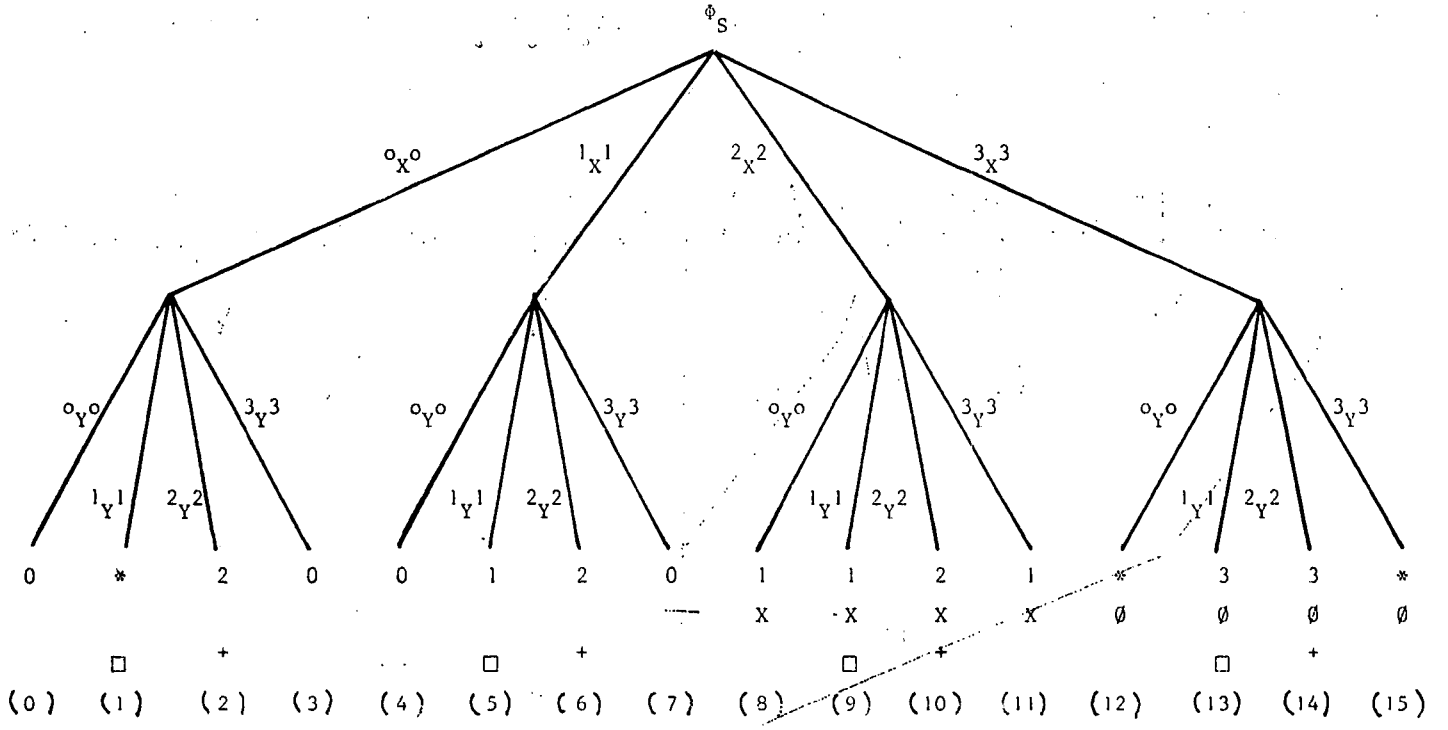

ϕ_S
$^{o}X^{o}$
$^{1}X^{1}$
$^{2}X^{2}$
$^{3}X^{3}$
$^{o}Y^{o}$
$^{1}Y^{1}$
$^{2}Y^{2}$
$^{3}Y^{3}$
0 * 2 0
0 1 2 0
1 1 2 1
* 3 3 *
— X X X Ø Ø Ø Ø
□ + □ + □ + □ +
(0) (1) (2) (3) (4) (5) (6) (7) (8) (9) (10) (11) (12) (13) (14) (15)

Fig. 18

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

References

[1] Computer science and multiple-valued logic (theory and applications) ed. David C. Rine (Nort-Holland, Amsterdam 1977.) p. 262—282.
[2] S. Y. H. Su and A. A. Sarris, "The Existence of Multivalued Algebra with Circuit Implementations IEEE. Comp. G. Repository R-70-114.
[3] S. Y. H. Su and A. A. Sarris, Relationship betwen Multivalued Switching Algebra and Boolean Algebra under Definitions of Complement. IEEE. Trans. Comp. vol C-21. May 1972.
[4] P. Varga Katalin, Módszerek Boole-függvények minimális vagy nem redundáns {∨, ∧, ⌐} vagy {NOR} vagy {NAND} bázisbeli, zárójeles vagy zárójel nélküli formuláinak előállítására. MTA SZTAKI Tanulmányok, 1/1973.
[5] P. Varga K., On Some Minimizing Algorithms of Boolean functions. Acta Techn. Budapest 77. 419.
[6] Varga A., Hatékony eljárás Boole-függvények irredundáns normálformáinak előállítására (doktori értekezés) Szeged, 1979, p. 101.

(Received Sept. 25, 1985.)

An Erdős—Ko—Rado type theorem II

By K. Engel and H.-D. O. F. Gronau

1. Introduction and results

Let R denote the interval $[1, r]$ of the first r positive integers. Let k be an integer with $0 \leqq k \leqq r$. The set of all k-element subsets of R will be denoted by $\binom{R}{k}$. The aim of this paper is to present the

Theorem 1. *Let $\mu \geqq 4$ and $\nu \geqq 4$ be integers. If $F \subseteq \binom{R}{k}$,*

$$\frac{r-1}{\nu}+1 \leqq k \leqq \frac{\mu-1}{\mu}(r-1), \tag{1}$$

and F satisfies

$$X_1 \cap X_2 \cap \ldots \cap X_\mu \neq \emptyset \quad \text{for all} \quad X_1, X_2, \ldots, X_\mu \in F, \tag{2}$$

as well as

$$X_1 \cup X_2 \cup \ldots \cup X_\nu \neq R \quad \text{for all} \quad X_1, X_2, \ldots, X_\nu \in F, \tag{3}$$

then

$$|F| \leqq \binom{r-2}{k-1}.$$

This is best possible. The families $F_{x,y} = \left\{X \in \binom{R}{k} : x \in X, y \notin X\right\}$, where x and y are different fixed elements of R, are maximal.

This theorem was proved, for $\mu \geqq 6$ and $\nu \geqq 6$ and for some partial cases of k if $\mu = 4,5$ or $\nu = 4,5$, in *Gronau* [2]. Our proof here uses the same method but in a refined version.

Condition (1) is natural. For all other k's one of the conditions (2) or (3) is satisfied automatically, and the problem reduces to the generalized *Erdős—Ko—Rado* theorem by *Frankl* [1]. For another simple proof, see *Gronau* [3].

Theorem 2. *(generalized Erdős—Ko—Rado theorem)*
Let $\mu \geqq 2$ *be an integer. If* $F \subseteq \binom{R}{k}$, $0 \leqq k \leqq \frac{\mu-1}{\mu} r$, *and* F *satisfies* (2), *then*

$$|F| \leqq \binom{r-1}{k-1}.$$

Turning to the complements we obtain a dual version.

Theorem 2'. *Let* $\nu \geqq 2$ *be an integer. If* $F \subseteq \binom{R}{k}$, $\frac{r}{\nu} \leqq k \leqq r$, *and* F *satisfies* (3), *then*

$$|F| \leqq \binom{r-1}{k}.$$

2. Some reductions

Let $\mu, \nu \geqq 4$, k and $F \subseteq \binom{R}{k}$ be given such that (1), (2) and (3) hold. If

$$\bigcap_{X \in F} X \neq \emptyset \quad \text{or} \quad \bigcup_{X \in F} X \neq R \quad \text{then} \quad |F| \leqq \binom{r-2}{k-1}$$

follows by Theorem 2 or 2' immediately. Since the described families $F_{x,y}$ have cardinality $\binom{r-2}{k-1}$ and satisfy (2) as well as (3), the proof of Theorem 1 will be completed by proving

Theorem 3. *Let* $\mu \geqq 4$ *and* $\nu \geqq 4$ *be integers. If* $F \subseteq \binom{R}{k}$ *and* F *satisfies* (2) *and* (3) *as well as* $\bigcap_{X \in F} X = \emptyset$ *and* $\bigcup_{X \in F} X = R$, *then*

$$|F| < \binom{r-2}{k-1}.$$

Observe that here is no restriction on k. Therefore, it is sufficient to prove Theorem 3 only for $\mu = \nu = 4$. Furthermore, we may restrict ourselves to $k \leqq \frac{r}{2}$ in the proof since $k > \frac{r}{2}$ follows by duality. We make use of some results from [2].

Proposition 1. ([2, Lemma 1]).

$$|X_1 \cap X_2| \geqq 3,$$

$$|X_1 \cap X_2 \cap X_3| \geqq 2 \quad \textit{for all} \quad X_1, X_2, X_3 \in F.$$

Proposition 2. *We may suppose that for all $X\in F$ it holds: If $i\notin X$, $j\in X$ and $i<j$, then $(X-\{j\})\cup\{i\}\in F$.*

The last proposition is a consequence of Lemma 4 in [2], by the *Erdős—Ko—Rado* exchange operation.

Finally we prove Theorem 3 for small k, similarly to [2], by a short argument.

Lemma 1. *Theorem 3 is true for* $k\leqq\frac{r}{4}+\frac{3}{2}$.

Proof. By Theorem 6 in [2], $|F|\leqq\binom{r}{k-3}$. Hence,

$$\frac{|F|}{\binom{r-2}{k-1}}\leqq\frac{\binom{r}{k-3}}{\binom{r-2}{k-1}}=\frac{r(r-1)(k-1)(k-2)}{(r-k+3)(r-k+2)(r-k+1)(r-k)}\leqq$$

$$\leqq\frac{r(r-1)\left(\frac{r}{4}+\frac{1}{2}\right)\left(\frac{r}{4}-\frac{1}{2}\right)}{\left(\frac{3}{4}r+\frac{3}{2}\right)\left(\frac{3}{4}r+\frac{1}{2}\right)\left(\frac{3}{4}r-\frac{1}{2}\right)\left(\frac{3}{4}r-\frac{3}{2}\right)}=\frac{16}{27}\frac{r}{r+\frac{2}{3}}\frac{r-1}{r-\frac{2}{3}}\frac{r+2}{r+2}\frac{r-2}{r-2}<1.\ \square$$

3. An upper bound for |F|

Suppose that F satisfies the suppositions of Theorem 3, and $\frac{r}{4}+\frac{3}{2}<k\leqq\frac{r}{2}$. We decompose F into F_1, F_2, and F_3 according to

$$F_1=\{X\in F:\ \{1,2\}\subseteq X\},$$
$$F_2=\{X\in F:\ 1\in X,\ 2\notin X\},$$
$$F_3=\{X\in F:\ 1\notin X\}.$$

i) Let $F_1'=\{X:\ X\cup\{1,2\}\in F_1,\ \{1,2\}\cap X=\emptyset\}$. Then F_1' is a family of $(k-2)$-element subsets of the $(r-2)$-element set $\{3,4,\ldots,r\}$ satisfying (3) for $v=4$. Since $k-2>\left(\frac{r}{4}+\frac{3}{2}\right)-2=\frac{r-2}{4}$, we may apply Theorem 2′ and obtain

$$|F_1|=|F_1'|\leqq\binom{r-3}{k-2}.\tag{4}$$

In order to estimate $|F_2|$ and $|F_3|$ we use the description of the families by walks in the plane. We associate with every $X\in\binom{R}{k}$ a certain walk. We start from $(0,0)$. If we are after i moves at point (a,b) then we turn to $(a,b+1)$ or $(a+1,b)$ depending on whether $i+1\in X$ or $i+1\notin X$. So every set of $\binom{R}{k}$ is associated with a walk from $(0,0)$ to $(r-k,k)$ and vice versa.

Let F_2' and F_3' denote the set of walks associated with F_2 and F_3, respectively. By the definition of F_2 and F_3, every walk of F_2' starts with (0, 0)—(0, 1)—(1, 1) whereas every walk of F_3' starts with (0, 0)—(1, 0).

ii) Every walk of F_2' meets the line $y=2x+2$, since otherwise, by Proposition 2, F_2 would contain the set $X_1=\{1, 3, 4, 6, 7, 9, 10, \ldots\}$. For the same reason, F would contain $X_2=\{1, 2, 4, 5, 7, 8, 10, \ldots\}$ and $X_3=\{1, 2, 3, 5, 6, 8, 9, \ldots\}$. But $|X_1 \cap X_2 \cap X_3|=|\{1\}|=1$, contradicting Proposition 1.

If a walk meets the line $y=2x+2$ the first time at $(i, 2i+2)$, $i \geqq 1$, then this walk passes through $(i, 2i-1)$, too. Hence the number of these walks is not greater than

$$\binom{3i-3}{i-1}\binom{r-3i-2}{k-2i-2}$$

since $\binom{3i-3}{i-1}$ is the total number of walks from (1, 1) to $(i, 2i-1)$, whereas $\binom{r-3i-2}{k-2i-2}$ is the total number of walks from $(i, 2i+2)$ to $(r-k, k)$. Consequently, using $\binom{0}{0}=1$, we obtain

$$|F_2| = |F_2'| \leqq \sum_{i=1}^{\left\lfloor \frac{k-2}{2} \right\rfloor} \binom{3i-3}{i-1}\binom{r-3i-2}{k-2i-2} \tag{5}$$

iii) Every walk of F_3' meets the line $y=3x+1$. This follows by the same arguments as in the preceding case recalling (2). Thus,

$$|F_3| = |F_3'| \leqq \sum_{i=1}^{\left\lfloor \frac{k-1}{3} \right\rfloor} \binom{4i-4}{i-1}\binom{r-4i-1}{k-3i-1} \tag{6}$$

By (4), (5), and (6) we obtain

$$|F| \leqq \binom{r-3}{k-2} + \sum_{i=1}^{\left\lfloor \frac{k-2}{2} \right\rfloor} \binom{3i-3}{i-1}\binom{r-3i-2}{k-2i-2} + \sum_{i=1}^{\left\lfloor \frac{k-1}{3} \right\rfloor} \binom{4i-4}{i-1}\binom{r-4i-1}{k-3i-1}. \tag{7}$$

4. Some lemmas

In order to estimate (7) we need the following lemmas.

Lemma 2. *For any natural numbers n and i with $n \geqq 2$,*

$$\frac{\binom{n(i+1)}{i+1}}{\binom{ni}{i}} \leqq \frac{n^n}{(n-1)^{n-1}}.$$

Proof

$$\frac{\binom{n(i+1)}{i+1}}{\binom{ni}{i}} = \frac{(n(i+1))!\,i!\,((n-1)i)!}{(i+1)!((n-1)(i+1))!\,(ni)!} = \frac{n(i+1)}{i+1}\prod_{j=1}^{n-1}\frac{ni+j}{(n-1)i+j} \leqq$$

$$\leqq n\prod_{j=1}^{n-1}\frac{n}{n-1} = \frac{n^n}{(n-1)^{n-1}}. \quad \square$$

Lemma 3. *For integers r, k, i satisfying $k \leqq \frac{r}{2}$ and $i \geqq 1$ we have*

a) $$\frac{\binom{r-3i-5}{k-2i-4}}{\binom{r-3i-2}{k-2i-2}} \leqq \frac{1}{8} \quad \text{if} \quad i \leqq \frac{k-4}{2}.$$

b) $$\frac{\binom{r-4i-5}{k-3i-4}}{\binom{r-4i-1}{k-3i-1}} \leqq \frac{1}{16} \quad \text{if} \quad i \leqq \frac{k-4}{3}.$$

Proof. Since, for positive α and β, $\alpha\cdot\beta \leqq \left(\frac{\alpha+\beta}{2}\right)^2$, and $k \leqq \frac{r}{2}$, we have

a) $$\frac{\binom{r-3i-5}{k-2i-4}}{\binom{r-3i-2}{k-2i-2}} = \frac{(r-3i-5)!(k-2i-2)!(r-k-i)!}{(k-2i-4)!\,(r-k-i-1)!\,(r-3i-2)!} =$$

$$= \frac{(k-2i-2)(k-2i-3)(r-k-i)}{(r-3i-2)(r-3i-3)(r-3i-4)} \leqq \frac{\frac{r}{2}-2i-2}{r-3i-4}\frac{\left(\frac{r-3i-3}{2}\right)^2}{(r-3i-3)^2} \leqq$$

$$\leqq \frac{1}{8}\frac{r-4i-4}{r-3i-4} \leqq \frac{1}{8}.$$

b) $$\frac{\binom{r-4i-5}{k-3i-4}}{\binom{r-4i-1}{k-3i-1}} = \frac{(r-4i-5)!(k-3i-1)!(r-k-i)!}{(k-3i-4)!\,(r-k-i-1)!\,(r-4i-1)!} =$$

$$= \frac{(k-3i-1)(k-3i-3)(k-3i-2)(r-k-i)}{(r-4i-3)(r-4i-4)(r-4i-1)(r-4i-2)} \leqq$$

$$\leqq \frac{\frac{r}{2}-3i-1}{r-4i-3}\,\frac{\frac{r}{2}-3i-3}{r-4i-4}\,\frac{\left(\frac{r-4i-2}{2}\right)^2}{(r-4i-2)^2} \leqq$$

$$\leqq \frac{1}{16}\frac{r-6i-2}{r-4i-3}\frac{r-6i-6}{r-4i-4} \leqq \frac{1}{16} \text{ for } i \geqq 1. \quad \square$$

Immediate induction consequences of our lemmas are

$$\binom{ni}{i} \leqq \binom{n\gamma}{\gamma}\left[\frac{n^n}{(n-1)^{n-1}}\right]^{i-\gamma} \quad \text{if} \quad i \geqq \gamma,$$

$$\binom{r-3i-2}{k-2i-2} \leqq \left(\frac{1}{8}\right)^{i-1}\binom{r-5}{k-4} \quad \text{if} \quad 1 \leqq i \leqq \left\lfloor\frac{k-2}{2}\right\rfloor, \tag{9}$$

and

$$\binom{r-4i-1}{k-3i-1} \leqq \left(\frac{1}{16}\right)^{i-1}\binom{r-5}{k-4} \quad \text{if} \quad 1 \leqq i \leqq \left\lfloor\frac{k-1}{3}\right\rfloor. \tag{10}$$

Finally, by (8), with $n=3$ and 4 we have

$$\sum_{i=1}^{\infty}\binom{3i-3}{i-1}\left(\frac{1}{8}\right)^{i-1} \leqq 1+\frac{\binom{3}{1}}{8}+\frac{\binom{6}{2}}{8^2}+\frac{\binom{9}{3}}{8^3}+\frac{\binom{12}{4}}{8^4}+$$

$$+\frac{\binom{15}{5}}{8^5}\left[\sum_{i=6}^{\infty}\left(\frac{3^3}{2^2}\right)^{i-6}\left(\frac{1}{8}\right)^{i-6}\right] =$$

$$= 1+\frac{3}{8}+\frac{15}{64}+\frac{84}{512}+\frac{495}{4096}+\frac{3003}{32768}\,\frac{1}{1-\frac{27}{32}} < 2.481, \tag{11}$$

and

$$\sum_{i=1}^{\infty}\binom{4i-4}{i-1}\left(\frac{1}{16}\right)^{i-1} \leqq 1+\frac{\binom{4}{1}}{16}+\frac{\binom{8}{2}}{16^2}+\frac{\binom{12}{3}}{16^3}+ \tag{12}$$

$$+\frac{\binom{16}{4}}{16^4}\left[\sum_{i=5}^{\infty}\left(\frac{4^4}{3^3}\right)^{i-5}\left(\frac{1}{16}\right)^{i-5}\right] = 1+\frac{4}{16}+\frac{28}{256}+\frac{220}{4096}+\frac{1820}{65536}\,\frac{1}{1-\frac{16}{27}} < 1.482.$$

5. Proof of Theorem 3

Now we are able to prove the Theorem 3. Starting with (7) and using (9), (10), (11), and (12) we get

$$|F| \leqq \binom{r-3}{k-2}+\left\{\sum_{i=1}^{\infty}\binom{3i-3}{i-1}\left(\frac{1}{8}\right)^{i-1}+\sum_{i=1}^{\infty}\binom{4i-4}{i-1}\left(\frac{1}{16}\right)^{i-1}\right\}\binom{r-5}{k-4} <$$

$$< \binom{r-3}{k-2}+\{2.481+1.482\}\binom{r-5}{k-4} < \binom{r-3}{k-2}+4\binom{r-5}{k-4}.$$

Furthermore, recalling $k \leqq \frac{r}{2}$,

$$\frac{|F|}{\binom{r-2}{k-1}} < \frac{\binom{r-3}{k-2}}{\binom{r-2}{k-1}} + 4\frac{\binom{r-5}{k-4}}{\binom{r-2}{k-1}} = \frac{k-1}{r-2} + 4\frac{(k-1)(k-2)(k-3)}{(r-2)(r-3)(r-4)} \leqq$$

$$\leqq \frac{\frac{r}{2}-1}{r-2} + 4\frac{\left(\frac{r}{2}-1\right)\left(\frac{r}{2}-2\right)\left(\frac{r}{2}-3\right)}{(r-2)(r-3)(r-4)} = \frac{1}{2} + 4\frac{1}{8}\frac{(r-4)(r-6)}{(r-3)(r-4)} < 1.$$

This completes the proof. □

ERNST-MORITZ-ARNDT-UNIVERSITÄT
SEKTION MATHEMATIK
DDR-2200 GREIFSWALD
FRIEDRICH-LUDWIG-JAHN-STR. 15 A

WILHELM-PIECK-UNIVERSITÄT
SEKTION MATHEMATIK
DDR-2500 ROSTOCK
UNIVERSITÄTSPLATZ 1

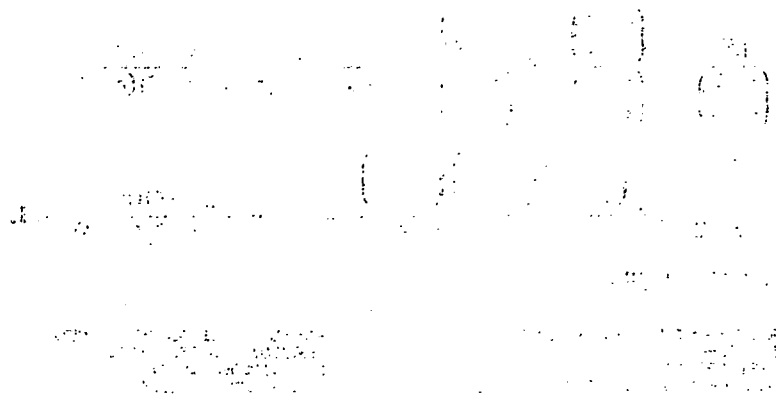
References

[1] FRANKL, P., On Sperner families satisfying an additional condition. *J. Combin. Theory Ser. A* v. 20, 1976, pp. 1—11.

[2] GRONAU, H.-D. O. F., An Erdős—Ko—Rado type theorem, Finite and Infinite Sets (Proc. 6th Hung. Colloq. on Combinatorics, Eger 1981), pp. 333—342, *Colloq. Math. Soc. J. Bolyai, 37*, North-Holland, Amsterdam, 1985.

[3] GRONAU, H.-D. O. F., On Sperner families in which no k sets have an empty intersection III, *Combinatorica*, v. 2, 1982, 25—36.

(Received March 25, 1985.)



Giving mathematical semantics of nondeterministic and parallel programming structures by means of attribute grammars

By R. ALVAREZ GIL

1. Introduction

A formal definition of the semantics of the programming languages is a prerequisite for the verification of specific implementations. The definition of the semantics of a programming language can be formulated in different ways. Knuth [8] has introduced attribute grammars for this purpose, Scott and Strachey [10] has developed a mathematical method.

Many papers has been published about the relation between attribute grammars and mathematical semantics. Mayoh [9] has shown that for any attribute grammar it is possible to find an equivalent mathematical semantics. The reverse affirmation is true only with several restrictions [3].

In this paper after the introduction of the used notations and the concept of attribute grammar we give an example to show how it is possible to describe the mathematical semantics of programming languages with the help of attribute grammars in section 3.

In section 4 we describe the nondeterministic structures introduced by Dijkstra [2] and give their mathematical semantics by means of attribute grammars. For this purpose it is sufficient to employ the notion of the possible states used in the description of the semantics of sequential programs, but many-valued functions are necessary to give the semantics of statements and programs. In section 5 we have to extend the notion of the possible states, too, to give the mathematical semantics of a parallel programming language allowing communication of sequential processes through Hoare's monitors.

2. Attribute grammars

In this section we will follow in general the notations used in [7].
An attribute grammar is defined by a 4-tupel

$$AG = (CFG, A, SR, SC)$$

where $CFG=(N, T, P, S)$ is a reduced context-free grammar (N is the set of non-terminal symbols, T is the set of terminal symbols, P is the set of productions or syntactical rules and S is the start symbol), A is a finite set of attributes, SR is the set of semantic rules and SC is the set of semantic conditions.

A production $p\in P$ is denoted by $p: X_0 ::= X_1 X_2 \dots X_{n_p}$, where $n_p \geqq 0$, $X_0 \in N$ and $X_i \in N \cup T$ for all i $(1 \leqq i \leqq n_p)$.

For each $X \in N$ there is a subset $A(X)$ of A. The set of attributes A is partitioned into two disjoint subsets A_S and A_I, the set of synthesized attributes and the set of inherited attributes: $A=A_S \cup A_I$ and $A_S \cap A_I = \emptyset$. Thus $A(X)$ is partitioned into two disjoint subsets $A_S(X)$ and $A_I(X)$, so that $A_S(X) \subseteq A_S$, $A_I(X) \subseteq A_I$ and $A(X)=A_S(X) \cup A_I(X)$.

If $p: X_0 ::= X_1 X_2 \dots X_{n_p} \in P$ is a production, $X \in N$ occurs in p and $a \in A(X)$, then $X \cdot a$ denotes the attribute occurrence of a in p associated to X. The set A_p of attribute occurrences of a production p is defined by $A_p = \bigcup_{i=0}^{n_p} A_p(X_i)$, where $A_p(X_i) = \{X_i \cdot a : a \in A(X_i)\}$ if $X_i \in N$, and $A_p(X_i) = \emptyset$ if $X_i \in T$. The set OA_p of output attribute occurrences of a production p is defined by

$$OA_p = \{X_i \cdot a \in A_p : (i = 0 \text{ and } a \in A_S(X_i)) \text{ or } (i > 0 \text{ and } X_i \in N \text{ and } a \in A_I(X_i))\},$$

and the set IA_p of input attribute occurrences of a production p is defined by

$$IA_p = \{X_i \cdot a \in A_p : (i = 0 \text{ and } a \in A_I(X_i)) \text{ or } (i > 0 \text{ and } X_i \in N \text{ and } a \in A_S(X_i))\} = A_p \setminus OA_p.$$

For each $p \in P$ there is a subset SR_p of SR and a subset SC_p of SC, the set of the production semantic rules and the set of the production semantic conditions, so that

$$SR = \bigcup_{p \in P} SR_p \quad \text{and} \quad SC = \bigcup_{p \in P} SC_p.$$

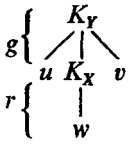
For each production $p \in P$, for each attribute occurrence $X_i \cdot a \in OA_p$ there is one and only one semantic rule $f \in SR_p$ which determines the value of $X_i \cdot a$, and each semantic rule $f \in SR_p$ determines the value of some output attribute occurrence $X_i \cdot a \in OA_p$.

Let s be a sentence of $L(CFG)$ derived by

$$S \xrightarrow{*} xYy \xrightarrow[q]{} xuXvy \xrightarrow[r]{} xuwuy \xrightarrow{*} s$$

A node K_X represents the symbol X in the derivation tree t_s corresponding to that derivation and it is called an instance of X. For each attribute occurrence $X \cdot a$ an attri-

bute instance $K_X \cdot a$ is associated to K_X. The values of the inherited attribute instances associated to K_X are defined by rules in SF_g, and the values of the synthesized attribute instances associated to K_X are defined by rules in SF_r.



A derivation tree augmented with the attribute instances is called an attributed derivation tree. An attribute evaluations strategy is an algorithm to calculate the value of each attribute instance. The single natural condition for the application of a semantic rule $f \in SR$ is that the value of the attribute instances which appear as arguments of f were calculated previously. This condition generates a dependence relation on the attribute instances of the attributed derivation tree.

An attribute grammar is well defined if and only if for each attributed derivation tree the graph belonging to the generated dependence relation is noncircular. A well defined attribute grammar is also called noncircular. The problem of the decision of attribute grammars noncircularity is NP-complete [5], but subclasses of the class of noncircular attribute grammars have been introduced in which we can decide in polynomial time whether an attribute grammar belongs to the subclass. Such subclasses are for example the LR [1], the ASE [6] and the OAG [7] attribute grammars.

3. Giving mathematical semantics by means of attribute grammars

As usual in the mathematical semantics we consider a program as a function on the set of the possible states

$$S_p = \{\{(v_1, t_1), \ldots, (v_n, t_n)\}: t_1 \in T'_{v_1}, \ldots, t_n \in T'_{v_n}\}$$

where $v_1, \ldots, v_n$ are all the variables which appear in the program, $T'_{v_i} = T_{v_i} \cup \{\text{unvalued, undefined}\}$. T_{v_i} is the set from which the variable v_i takes its values. The variable v_i in a state is unvalued if it has no value and is undefined, if its name is not valid in that state. Later in section 5 we have to extend and consequently redefine the notion of the possible states.

The semantics of the statements and a program p too are functions f_p: $S_p \to S_p$. In this section we define such functions for the programs of a very simple sequential language. For this purpose we need the following attributes:
Synthesized attributes:

name — to give a unique identifier for each variable of the program
T — to give the type of each variable and arithmetical expression
V — to give the set of declared variables and their types
S — to give the set of the possible states
g — a function g: $S \to T'$ to give the value of an arithmetical expression in a given state, where T' is the type of the arithmetical expression

h — a function h: $S \to \{\text{true, false}\}$ to give the value of a logical expression in a given state

f — a function f: $S \to S$ to give the semantics of the statements and the programs of the language.

Inherited attributes:

V' — to give the variables valid in the environment and their types

S' — to transmit towards the levels of the tree the set of the possible states

Nonterminal symbols and their attributes:

program has V, S, f
declaration_statement has V, f
declaration_list has V
declaration has V
variable_list has V
variable has name
type has T
statement_list has V, f, V', S'
statement has V, f, V', S'
expression has g, T, V', S'
bool_expression has h, V', S'

Syntactical rules and their semantic rules and semantic conditions:
(In a production $X_0 ::= X_1 X_2 \dots X_{n_p}$ $(n_p \geqq 0)$ we will omit the semantic rules of the form $X_0 \cdot a = X_i \cdot a$ $(1 \leqq i \leqq n_p)$ if there is no X_j $(1 \leqq j \leqq n_p$ and $i \neq j)$ which has the synthesized attribute a, and we will omit the semantic rules of the form $X_i \cdot a = X_0 \cdot a$ $(1 \leqq i \leqq n_p)$).

i) program ::= **begin** declaration_statement; statement_list **end**
program.V = declaration_statement.$V \cup$ statement_list.V
program.S = $\{\{(v, t_v)$: $(v, T) \in$ program.$V\}$: $t_v \in T \cup \{$unvalued, undefined$\}\}$
program.$f(s)$ = statement_list.f(declaration_statement.$f(s)$)
statement_list.V' = declaration_statement.V
statement_list.S' = program.S

ii) declaration_statement ::= **var** declaration_list
declaration_statement.$f(s) = \{(v, s(v))$: there is not $(v_1, T_1) \in$ declaration_list.V for which $v_1 = v\} \cup \{(v,$ unvalued): there is $(v_1, T_1) \in$ declaration_list.V for which $v_1 = v\}$

iii) declaration_list$_1$ ::= declaration; declaration_list$_2$
declaration_list$_1$.V = declaration.$V \cup$ declaration_list$_2$.V
condition: if $(v_1, T_1) \in$ declaration.V and $(v_2, T_2) \in$ declaration_list$_2$.V then $v_1 \neq v_2$

iv) declaration_list ::= declaration

v) declaration ::= variable_list **of** type
declaration.$V = \{(v,$ type.$T)$: $(v, \emptyset) \in$ variable_list.$V\}$

vi) variable_list$_1$ ::= variable, variable_list$_2$
variable_list$_1$.V = variable_list$_2$.$V \cup \{($variable.name$, \emptyset)\}$
condition: if $(v, \emptyset) \in$ variable_list$_2$.V then variable.name $\neq v$

vii) variable_list::=variable
variable_list.$V=\{$(variable.name, $\emptyset$)$\}$

viii) statement_list$_1$::=statement; statement_list$_2$
statement_list$_1$.V=statement.$V\cup$statement_list$_2$.V
statement_list$_1$.$f(s)$=statement_list$_2$.f(statement.$f(s)$)
condition: if $(v_1, T_1)\in$statement.V and $(v_2, T_2)\in$ statement_list$_2$.V then $v_1\neq v_2$

ix) statement_list::=statement

x) statement::=variable:=expression
statement.$V=\emptyset$
statement.$f(s)=\{(v, s(v))$: $v\neq$variable.name$\}\cup$ $\{$(variable.name, expression.$g(s)$)$\}$
condition: there is $(v, T_1)\in$statement.V' for which $v=$variable.name and $T_1=$expression.T

xi) statement$_1$::=**if** bool_expression **then** statement$_2$ **else** statement$_3$ **fi**
statement$_1$.V=statement$_2$.$V\cup$statement$_3$.V

$$\text{statement}_1.f(s)=\begin{cases}\text{statement}_2.f(s), \text{ if bool\_expression}.\\ \qquad h(s)=\text{true}\\ \text{statement}_3.f(s), \text{ if bool\_expression}.\\ \qquad h(s)=\text{false}\end{cases}$$

xii) statement$_1$::=**while** bool-expression **do** statement$_2$ **od**

$$\text{statement}_1.f(s)=\begin{cases}\text{statement}_1.f(\text{statement}_2.f(s)),\\ \qquad \text{if bool\_expression}.h(s)=\\ \qquad \text{true}\\ s, \text{ if bool\_expression}.h(s)=\text{false}\end{cases}$$

xiii) statement::=**begin** statement_list **end**

xiv) statement::=**begin** declaration_statement; statement_list **end**
statement.V=declaration_statement.$V\cup$statement_list.V
statement.$f(s)=\{(v,$ statement_list.$f($declaration_statement.$f(s))(v))$: there is not $(v_1, T_1)\in$declaration_statement.V for which $v_1=v\}\cup\{(v,$ undefined): there is $(v_1, T_1)\in$declaration_statement.V for which $v_1=v\}$
statement_list.V'=statement.$V'\cup$declaration_statement.V
condition: if $(v_1, T_1)\in$declaration_statement.V and $(v_2, T_2)\in$ statement_list.V then $v_1\neq v_2$

It is easy to show that the attribute grammar given above is well defined (non-circular), but we do not deal with this in our paper.

4 Semantics of nondeterministic structures

The syntax of the nondeterministic programming structures introduced by Dijkstra can be given by a context-free grammar as follows:

i) statement::=alternative_construct
ii) statement::=repetitive_construct
iii) alternative_construct::=**if** guarded_command_set **fi**
iv) repetitive_construct::=**do** guarded_command_set **od**
v) guarded_command_set::=guarded_command □ guarded_command_set
vi) guarded_command_set::=guarded_command
vii) guarded_command::=guard→guarded_list
viii) guard::=bool_expression
ix) guarded_list::=statement_list

From the context-free grammar given above it is clear that Dijkstra introduced two new statements: the alternative construct and the repetitive construct, based on the concept of guarded commands. The semantics of these statements was given by Dijkstra in [2] with the following words: "The alternative construct is written by enclosing a guarded command set by the special bracket pair **if**...**fi**. If in the initial state none of the guards is true, the program will abort; otherwise an arbitrary guarded list with a true guard will be selected for execution. The repetitive construct is written down by enclosing a guarded command set by the special bracket pair **do**...**od**. Here a state in which none of the guards is true will not lead to abortion but to proper termination; the complementary rule, however, it will only terminate in a state in which none of the guards is true: when initially or upon completed execution of a selected guarded list one or more guards are true, a new selection of a guarded list with a true guard will take place, and so on. When the repetitive construct has terminated properly, we know that all its guards are false".

In the case of nondeterministic structures to give the semantics of the alternative construct and the semantics of the repetitive construct it is necessary to use functions f: $S_p \rightarrow 2^{S_p}$ which can be obtained as synthesized attributes. It is clear that for nondeterministic statements it is not sufficient to use functions of the type f: $S_p \rightarrow S_p$ because the state valid at the beginning of the execution of a nondeterministic statement do not determine a unique state valid at the termination of the statement, that is more than one state can be the real state when the program finished the execution of the nondeterministic statement.

Now we give an attribute grammar to obtain as synthesized attribute the functions which give the semantics of the nondeterministic statements:

Synthesized attributes:

f — to give the function f: $S_p \rightarrow 2^{S_p}$ which describes the semantics
h — a function h: $S_p \rightarrow$ {true, false} to give the value of a logical expression in a given state

Inherited attributes:

—

Nonterminals and their attributes:
statement has f

alternative_construct has f
repetitive_construct has f
guarded_command_set has f
guarded_command has f
guarded_list has f
statement_list has f
guard has h
bool_expression has h

Syntactical rules and their semantic rules:

i) statement::=alternative_construct
statement.f=alternative_construct.f

ii) statement::=repetitive_construct
statement.f=repetitive_construct.f

iii) alternative_construct::=**if** guarded_command_set **fi**
alternative_construct.f=guarded_command_set.f

iv) repetitive_construct::=**do** guarded_command_set **od**

$$\text{repetitive\_construct}.f(s)=\begin{cases}\bigcup \text{repetitive\_construct}.f(s') \\ s'\in \text{guarded\_command\_set}.f(s) \\ \text{if guarded\_command\_set}.f(s)\neq\emptyset \\ s, \text{ if guarded\_command\_set}.f(s)=\emptyset\end{cases}$$

v) guarded_command_set$_1$::=guarded_command □ guarded_command_set$_2$
guarded_command_set$_1$.$f(s)$=guarded_command.$f(s)\cup$ guarded_command_set$_2$.$f(s)$

vi) guarded_command_set::=guarded_command
guarded_command_set.f=guarded_command.f

vii) guarded_command::=guard→guarded_list

$$\text{guarded\_command}.f(s)=\begin{cases}\text{guarded\_list}.f(s), \text{ if} \\ \quad \text{guard}.h(s)=\text{true} \\ \emptyset, \text{ if guard}.h(s)=\text{false}\end{cases}$$

viii) guard::=bool_expression
guard.h=bool_expression.h

ix) guarded_list::=statement_list
guarded_list.f=statement_list.f

Note: it is easy to see that the program aborts in an alternative construct if and only if the alternative construct is executed in such a state s for which the function f associated with the synthesized attribute to the alternative construct takes the empty set $\emptyset$ as its value. Furthermore this occurs if and only if all the guards in the guarded command set of the alternative construct are false.

5. Mathematical semantics of parallel programs and monitors

We will deal with parallel programs which have the following structure:

begin
(definition of the monitors);
process$_1$: **process var** $v_1^1, \ldots, v_{n_1}^1$;

. (statements of the process$_1$)

end of process$_1$
and
. (description of the process$_2$, ..., process$_{m-1}$)
and
process$_m$: **process var** $v_1^m, \ldots, v_{n_m}^m$;

. (statements of the process$_m$)

end of process$_m$
end

The processes communicate with each other through Hoare's monitors [4]. A monitor is a collection of local data and procedures and has the following structure:

monitor_name: **monitor**
begin
(declaration of data local to the monitor)
procedure proc_name (...formal parameters ...);
begin

. (procedure body)

end;
(declaration of other procedures local to the monitor);
(initialization of local data of the monitor)
end

To call a procedure of the monitor, it is necessary to give the name of the monitor and the name of the desired procedure:

monitor_name.proc_name (... actual parameters ...)

The procedures of a monitor are common to all existing processes, any process can at any time attempt to call such a procedure. However, it is essential that only one process at a time can be executing a procedure body, and any subsequent call must be held up until the previous call has been completed or has been held up. A process in execution can be held up by a wait statement and can be resumed by a signal statement. The structures of these statements are:

cond_variable.**wait**; cond_variable.**signal**

The cond_variable is a new type of variable, a condition variable, which is suitable to differentiate the reason for waiting. In practice, a condition variable is an initially empty queue of processes which are waiting on the condition.

A signal operation is followed immediately by resumption of a waiting process, without the possibility of an intervening procedure call from a third process. Wait operation is followed immediately by resumption of a process delayed by a signal instruction. New procedure can be executed only if there are not processes delayed by signal.

Now, after this short and necessary introduction, we will define the set of the possible program states by

$$S_p = \{\{(v_1^1, t_1^1), \ldots, (v_{n_1}^1, t_{n_1}^1), \ldots, (v_1^m, t_1^m), \ldots, (v_{n_m}^m, t_{n_m}^m), (1, q_1), \ldots, (m, q_m)\}:$$

$$t_1^1, \ldots, t_{n_1}^1, \ldots, t_1^m, \ldots, t_{n_m}^m \in T \text{ and } q_1, \ldots, q_m \in (\bigcup_{j \in M} (Z^j \times MC^j))^*\}$$

where T — is the type of the variables (for simplicity all the variables have the same type),
M is the set of declared monitors,
Z^j is the set of the possible states of the monitor j,
MC^j is the set of the possible monitor calls relative to the monitor j.

For each monitor j we define a function g_j: $MC^j \to 2^{Z^j}$ which is obtained as a synthesized attribute and is the function which gives the semantics of the monitor j. The g_j is a many-valued function because in the general case, for the termination of a monitor call the execution of other monitor calls are necessary which can not be predetermined, and furthermore the calculation of g_j have to be started from all possible states of the monitor j in which the call might occur. To each process i we associate a function f_i: $S_p \to 2^{S_p}$ which is also obtained as a synthesized attribute.

Let e: $j \cdot j_k(v') \in MC^j$ be a monitor call in the process i, where j is the called monitor, j_k is the called procedure and v' the actual parameter. We associate to this monitor call statement a function f_e: $S_p \to 2^{S_p}$ which is defined by

$f_e(s) = S' \subseteq S_p$ and $s' \in S'$ if and only if:

a) $s'(v) = s(v)$ for all v $(v \neq v')$
b) $s'(p) = s(p)$ for all p $(1 \leqq p \leqq m,\ p \neq i)$
c) $s'(v') = z_j'$ (parameter of j_k)
d) $s'(i) = s(i) \circ z_j'$ (call)
e) $z_j' \in g_j(e)$

where z_j' (parameter of j_k) gives the value of the parameter of the procedure j_k at the state z_j' of the monitor j, and z_j' (call) gives the execution sequence of monitor calls which leads to z_j' and the monitor states in which the monitor calls were executed. Because of this it is clear that in the elements of Z^j there is a pair of the form (call, x), where $x \in (Z^j \times MC^j)^*$.

It is necessary to introduce the pairs (i, q_i) for $1 \leqq i \leqq m$ in the set of possible program states because it is possible to decide whether a program state is really a possible program state only by the comparision of the sequences q_i $(1 \leqq i \leqq m)$. Only when the sequences q_i are in correspondence with each other the program state is really a possible program state. The condition for this correspondence is that it is possible to find a sequence $a_1 a_2 \ldots a_n \in (\bigcup_{j \in M} (Z^j \times MC^j))^*$ for which the following sentences are true:

i) if $a_i \in Z^j \times MC^j$ $(1 \leqq i \leqq n)$, then $a_i \in q_j$
ii) if b appears in q_i $(1 \leqq i \leqq m)$, then b appears in $a_1 a_2 \ldots a_n$
iii) if b precedes c in q_i, then b precedes c in $a_1 a_2 \ldots a_n$
iv) if b precedes c immediately in q_i, and b and c belong to the same monitor call of the process i; then b precedes c in $a_1 a_2 \ldots a_n$, and if b precedes d and d precedes c in $a_1 a_2 \ldots a_n$ then the monitor called in d is different from the monitor called in b or c.
v) if b is the first in $a_1 a_2 \ldots a_n$ which belongs to $(Z^j \times MC^j)$, then the first component of b is the initial state of the monitor j.

We do not give the attribute grammar for monitors and parallel processes because it is very long and can be constructed from the principles given in this section and the method described in the preceding sections.

6. Conclusions

In our opinion the attribute grammars are a powerful tool to give the mathematical semantics of programming languages in the case of nondeterministic programming structures or in the case of parallel processes communicating through Hoare's monitors too.

Attribute grammars give a mechanizable method to obtain for any program of the language the function described by the program, and consequently an attribute evaluation strategy can be viewed as a "compiler" which translates the program into a mathematical function.

Acknowledgement

The author wants to thank Dr. Árpád Makay for valuable discussions and suggestions.

Abstract

This paper describes attribute grammars for the description of the mathematical semantics of programming languages. It concentrates on the nondeterministic programming structures introduced by Dijkstra and parallel programming structures in which sequential processes communicate through Hoare's monitors.

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

References

[1] BOCHMAN, G. V., Semantic evaluation form Left to Right. CACM 19, 2, (February, 1976), 55—62.
[2] DIJKSTRA, E. W., Guarded Commands, Nondeterminancy and Formal Derivation of Programs. CACM 18, 8, (August, 1975), 453—457.
[3] GANZINGER, H., Transforming denotational semantics into practical attribute grammars. In Lecture Notes in Computer Sciences 94, 1980, 1—69.
[4] HOARE, C. A. R., Monitors: An Operating Systems Structuring Concept. CACM 17, 10, (October, 1974), 549—557.
[5] JAZAYERI, M., OGDEN, W. F. and ROUNDS, W. C., The intrinsically exponential complexity of the circularity problem for attribute grammars. CACM 18, 12, (December, 1975), 697—721.
[6] JAZAYERI, M. and WALTER, K. G., Alternating Semantic Evaluator. In Proc. of ACM 1975 Ann. Conf., 230—234.
[7] KASTENS, U., Ordered Attributed Grammars. Acta Informatica 15, 1980, 229—256.
[8] KNUTH, D. E., Semantics of context-free languages. Math. Systems Theory 2, 1968, 127—145.
[9] MAYOH, B. H., Attribute grammars and mathematical semantics. SIAM J COMPUT. 10, 3, (August, 1981), 503—518.
[10] SCOTT, D. and STRACHEY, C., Towards a mathematical semantics for computer languages. Tech. Mon. PRG-6, Oxford U. Comp. Lab., 1971.

(Received Sept. 13, 1985.)

Acta Cybernetica, Tom. 7, Fasc. 4, Szeged, 1986.

A new programming methodology using attribute grammars

E. SIMON

Keywords: attribute grammars, automatic program generation, modular decomposition, programming methodology, program specification.

Abstract

Attribute grammars have been constructed for describing the static semantics of programming languages and have been shown useful in a wide variety of automatic compiler generations. This paper presents a new application of attribute grammars to specify hierarchical and functional programs. An algorithm to evaluate attribute grammars is demonstrated. Several attributes can be evaluated in parallel too. A simple model for generating PASCAL like programs is given. A new metalanguage PLASTIC is introduced as an adequate tool for specifying hierarchical and functional programs. A simple PLASTIC program is presented to help attain the new programming methodology.

1. Introduction

Over the last decade there has developed an acute awareness of the need to introduce abstraction and mathematical rigour into the programming process. This increased formality allows for the automatic manipulation of software, increasing productivity, and, even more importantly, the manageability of complex systems. Along those lines, attribute grammars (AG) of Knuth [6] constitute a formal mechanism for specifying translations between languages [2, 8, 11]. By automatically generating the inverse translators we would be able to translate any program written for one processor into the command language of any other processor [13]. There are some methods for incremental evaluation of AG to produce so called incremental compilers [3]. An essential question is how to verify the correctness of the AG specification. In contrast with the attribute evaluation problem, this has not been studied well and only a few results have been reported up to now [1, 5].

Although several efforts have been made to obtain efficient evaluators, the first good algorithm for attribute evaluation has been proposed by T. Katayama [4]. Principally this algorithm accepts absolutely noncircular AG although extension to general noncircular AG is straightforward. In the model nonterminal symbols are considered to be functions which map their inherited attributes to their synthesized attributes and associate procedures to realize these functions with the nonterminal

symbols. The entire AG is then transformed into a set of mutually recursive procedures. When applied to an AG whose attribute evaluation process can be performed in a single pass from left-to-right, the algorithm can generate an evaluator which can be combined with the top-down parsers to result in the so-called recursive-descendent compilers if the underlying CF grammars are LL(k). However data dependency sometimes allows several attributes be evaluated parallel supposing that we have associated one procedure for each synthesized attribute.

As it is widely recognized, hierarchical specification techniques are the most promising methods in constructing complex and large softwares in well structured way, and in fact they are the most successfully used ones in practice as it is represented for example by SYCOMAP [10]. In these methodologies softwares are hierarchically decomposed into modules and they are successively refined until concrete and machine executable programs are obtained from their abstract specifications cf. CDL2. Although they are extremely natural and useful the current states seems to be that automatic program generation from the specifications and their verification are prevented due to the lack of strict formalization.

The hierarchical and functional programming methodology presented in this paper is based on attribute grammars. Applying the results of [4], we obtain a new program specification technique which stands mechanical program generation. In our approach we consider a program specification as an AG where program modules are represented by nonterminal symbols of the grammar, module decompositions correspond to production rules, input and output data of the modules correspond to attributes of the nonterminal symbols and computations done in the modules are specified by the semantic rules. Our methodology has the following three desirable properties. It allows hierarchical descriptions of complex functional programs in a very natural way. We have means to mechanically generate efficient procedural type programs from the descriptions and verification of their correctness can also be performed hierarchically.

In this paper we give our formalism and then the metalanguage PLASTIC is stated. Before presenting the program generation algorithm a simple example is shown. The PLASTIC system, implemented in PASCAL is now under development. The PLASTIC compiler is specified in HLP/PASCAL metalanguage [12].

2. Formal description

Essence of our approach is to use a mechanism based on the Knuth's attribute grammar [6] to describe programs. Therefore a hierarchical and functional program (or simply HFP) is a 6-tuple

$$(M, m_0, A, D, V, F)$$

where

(1) M is a set of modules. We assume that M contains the special modul called a *null module* which is used to terminate decomposition. The null modul is denoted by null symbol.

(2) $m_0 \in M$ is an *initial module*.

(3) A is a set of input and output *attributes* of modules. With any modul except the null module, there is associated a set of input and output data called attributes and the set of attributes of $X \in M$ is denoted by $A[X]$. $A[X]$ is a disjoint union of the set $\mathrm{IN}[X]$ of *input* attributes and the set $\mathrm{OUT}[X]$ of *output attributes*. They are called inherited and synthesized attributes, respectively, in the AG terminology.

(4) $D \subset M \times M^*$ is a finite set of *module decompositions*. An element $d \in D$ is called a decomposition and is denoted by

$$d\colon\ X_0 \to X_1 X_2 \ldots X_n \quad cond \quad C_d$$

for $X_0, \ldots, X_n \in M$. We say that the module X_0 can be decomposed into modules $X_1, X_2, \ldots, X_n$ if a *decomposition* condition C_d is satisfied. C_d specifies the condition in terms of input attributes of X_0. When a is an attribute of X_k, that is, $a \in A[X_k]$, $X_k \cdot a$ is called an attribute occurrence of the decomposition d. It is called an *input occurrence* (by an alternative denotation $X_k \downarrow a$) if $a \in \mathrm{IN}[X_k]$ and an output occurrence $(X_k \uparrow a)$ if $a \in \mathrm{OUT}[X_k]$.

(5) V is a set of value domains of attributes.

(6) F is a set of *attribute mappings* for describing functional equalities among attributes. Let d be a decomposition $X_0 \to X_1 X_2 \ldots X_n \in D$. For each output occurrence $v = X_0 \uparrow a$ with $a \in \mathrm{OUT}[X_0]$ and input occurrence $v = X_k \downarrow a$ with $a \in \mathrm{IN}[X_k]$, $1 \leqq k \leqq n$, there exists a function $f_{d,v}$ to compute the value of v from the values of other attribute occurrences $v_1, \ldots, v_m$ in d. The set $D_{d,v} = \{v_1, \ldots, v_m\}$ is called *dependency set* of $f_{d,v}$. If we denote the value domain of v by domain (v), $f_{d,v}$ is a mapping domain $(v_1) \times \ldots \times$ domain $(v_m) \to$ domain (v).

That is, in every decomposition functions are specified to compute the values of outputs for main module and inputs to submodules.

Let us define a *decomposition tree* which shows the result of all decompositions applied to the initial module m_0. It corresponds the derivation tree of CF grammars and is defined recursively by the following

(1) the null module is a decomposition tree, and

(2) if $T_1, \ldots, T_n$ are decomposition trees with the root module $X_1, \ldots, X_n$, respectively, and $X_0 \to X_1 \ldots X_n$ *cond* C is a decomposition, then the tree

$$X_0[T_1, \ldots, T_n]$$

which consists of the root X_0 and the subtrees $T_1, \ldots, T_n$ is a decomposition tree.

A *computation tree* T is a decomposition tree whose nodes are labelled by attribute values in such a way that for any module X_0 in T and the decomposition $d\colon X_0 \to X_1 \ldots X_n$ *cond* C_d applied at the module the following conditions are satisfied

(i) the decomposition condition C_d is true,

(ii) for any output occurrence v of X_0 or input occurrence v of X_k $1 \leqq k \leqq n$, the following functional equality holds

$$v = f_{d,v}(v_1, \ldots, v_m) \quad \text{where} \quad D_{d,v} = \{v_1, \ldots, v_m\}.$$

It should be noted that a computation tree represents a particular execution of an HFP corresponding to the particular values of input data fed to the initial module.

3. The PLASTIC metalanguage

PLASTIC is a new metalanguage designed to support the use of abstractions in program construction. Work in programming methodology has led to the realization that three kinds of abstractions — procedural, control, and especially data abstractions — are useful in the programming process. Among these, only the procedural abstraction is supported well by conventional languages, through the procedure or subroutine. ALPHARD [9] and CLU [7] provide, in addition to procedures, novel linguistic mechanisms that support the use of data and control abstractions. In contradiction to these languages the PLASTIC system is altogether based on a few results of AG. In the module specifications, control abstraction is realized by the semantic functions and decomposition conditions. Data types can be refined successively as the decomposition proceeds.

A PLASTIC program consists of five parts. We first define some global data types for the procedures and functions. The auxiliary functions and procedures that are used in decomposition rules are declared in procedure declarations. The allowed primitive functions and procedures form a subset of those of PASCAL, since both the procedure type and the parameter types are restricted to allowed input-output attribute types. The interpretation of procedures and functions is the same as in PASCAL. Comments are indicated by the character %, whose appearance outside a proper string means that the rest of the line is interpreted as a comment and is skipped by the system. The strings belonging to the token class IDENTIFIER begin with a letter which is followed by letters or digits or underscores.

Before the module specifications the name of the initial module is given. The values of the input attributes of the initial module are assigned by read operations. The main part of a PLASTIC program is the module specification. We associate a set of input and output data with each module X. Computations done in the module X_0 is specified decompositionwise by giving a set of functional equalities which hold among attributes of X_0 and its submodules $X_1, \ldots, X_n$, and thus they are reduced to the computations done in submodules. Repeating the module decomposition process until terminal modules are reached completes the program design. If there are recursive modules or if there are modules whose decompositions are not unique there may occur numbers of trees each of which corresponds to a specific computation. We have attached declarations for data types of attributes to decompositions. They are refined successively as the decomposition proceeds. Different decompositions for a module are separated to versions. The input attribute occurrence can be denoted by $\downarrow$ while the output occurrence by $\uparrow$. In the attribute occurrences the name of the module to be decomposed must not be specified.

Simple copy rules of the form "$X \cdot a := Y \cdot b$" can often be left unwritten by applying the so-called elimination principle, if so desired. It is applicable in two situations. First, if a is an output attribute, then X must be the left-hand side of the decomposition and Y must be the only module on the right-hand side of the decomposition having an occurrence of attribute a. Alternatively, if a is an input attribute, then Y must be the left-hand side of the decomposition and X can be any of the modules on the right-hand side of the decomposition. In both cases the nonexistence of a rule for $X \cdot a$ is an indication to the PLASTIC system to include the copy rule in the decomposition. In the module and submodule specification the input and output attributes

are separated by semicolon. The keywords "description", "specification", "module", "submodule", "version", "condition", etc. can be abbreviated to "descr", "spec", "mod", "submod", "vers", "cond" etc. We assumed that a PLASTIC program is deterministic, that is, decomposition conditions of distinct decompositions with the same left-hand side module do not become true simultaneously for any value of its input attributes.

In the last part of a PLASTIC description the user can prescribe the implementation commands. As we shall see data dependency sometimes allows several attributes to be evaluated simultaneously. In our system these attributes are evaluated in a single procedure call, because this reduces overheads due to procedure activations and increases chances of parallel execution. The keyword "parallel" stands for these output attributes which have to be evaluated simultaneously if it is possible. The default option for attribute evaluation is sequential. One of the major goals of PLASTIC is to provide a mechanism to support the use of good programming methodology. To meet this goal, we must provide more than just the language mechanism for the generator: we must also provide a way to specify their effects. A natural means of doing this for implementation is to specify how to realize the evaluation of an attribute. There are three different kinds of realization. The default option is procedural. In this case for each module X and output attributes a single procedure will be generated. The keyword "macro" stands for those output attributes which are evaluated by executing a macro call. If there are same precompiled procedures for so called null modules, they can be activated by a call "statement".

The problem of data abstraction and its detailed discussion is beyond the scope of this paper except giving a comment that every hierarchical specification methodology should be equipped with a hierarchical data abstraction mechanism and in the case of PLASTIC the algebraic abstraction would be most appropriate.

Figure 1 shows a PLASTIC solution of binary conversion. Suppose we have a file containing record of binary characters. In order to verify the conversional algorithm we have to compute the value of binary number $b=b_1b_2\ldots b_n$ in two ways. Design a program that reads the character file and compute the binary numbers val1 and val2. The initial modul is START. We have attached declarations for data types of attributes to decompositions. We have assumed the existence of several functions on primitive data types, which are denoted by bold-face type letters. Their meaning will be selfexplanatory from their names. The common declarations for types, symbols and rule are written in the head of module descriptions. Copy-rules should not be specified, because they are generated automatically by the system.

4. Translation of PLASTIC program

Besides its static description, one of the outstanding features of PLASTIC specification technique is that we have means to translate mechanically the specification into machine executable forms. This is called attribute evaluation in the attribute grammar theory.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%   PLASTIC description for computing the value of binary   %%%%
%%%%        number b = b1b2...bn   in two ways given by        %%%%
%%%%        val1 (b1b2...bn) = b1*2↑(n-1) + val1 (b2...bn)     %%%%
%%%%        val2 (b1b2...bn) = 2*val2 (b1...bn-1) + bn         %%%%
%%%%        val1 () = val2 () = 0                              %%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

begin description bin_conv
common data types
val1, val2, pos: integer; neg: boolean;
procedures
procedure read (var input: file of elem); ...;
function last (input: file of elem); elem; ...;
function remain (input: file of elem): boolean; ...;
...
initial module is start
specifications
%%1%%
module start (↓input; ↑val1, ↑val2);
types input: file of elem;
submodule sign (↓elem; ↑neg);
          list (↓input, ↓pos; ↑val1, ↑val2);
version: 1
rule start=sign list;
do input<=read (input);
   list↓pos:=0;
   val1:=if sign↑neg then -list↑val1 else list↑val1;
   val2:=if sign↑neg then -list↑val2 else list↑val2;
   sign↓elem:=head (input);
   list↓input:=tail (input);
cond not empty (input);
version: 2
rule start=; do val1:=0; val2:=0;
cond always;
end start;
```

```
%%2%%
module sign(↓elem; ↑neg);
types elem: character;
rule sign=;
version: 1
do neg:=true;
cond elem="—";
version: 2
do neg:=false; cond elem="+"; end sign;
%%3%%
module list (↓input, ↓pos; ↑val1, ↑val2);
submodule list, digit (↓input, ↓pos; val1, ↑val2);
version: 1
rule list=digit;
do % digit↓pos:=pos;                    copy-rule
% digit↓input:=input;                   copy-rule
% val1:=digit↑val1;                     copy-rule
% val2:=digit↑val2;                     copy-rule
% copy-rule will be generated without specification
cond empty (remain (input));
version: 2
rule list=list digit;
do digit↓input:=last(input);
   list↓input  :=remain(input);
   list↓pos    :=pos+1;
   val1        :=list↑val1+digit↑val1;
   val2        :=2*list↑val2+digit↑val2;
cond always;
end list;
%%4%%
module digit (↓elem, pos; ↑val1, val2);
types elem: character;
rule digit =;
version: 1
do val1:=0; val2:=0;
cond elem="0";
version: 2
do val1:=2**pos; val2:=1;
cond elem="1";
end digit;
implementation
val1, val2: parallel;
%         : statement;
sign, digit: macro;
start, list : procedure;
end description bin_conv.
```

Figure 1

4.1. Notations

Let d: $X_0 \to X_1 X_2 \ldots X_n$ be a decomposition. A *dependency graph* DG_d *for the decomposition* d, which gives dependency relationship among attribute occurrences of d, is defined by

$$DG_d = (DV_d, DE_d)$$

where the node set DV_d is the set of all attribute occurrences of d and the edge set DE_d is the set dependency pairs for d. Formally

$$DV_d = \{X_k \cdot a \mid k = 0, \ldots, n \text{ and } a \in A[X_k]\}$$

$$DE_d = \{(v_1, v_2) \mid v_1 \in D_{d, v_2}\}.$$

When a computation tree T is given a *dependency graph* DG_T *for the computation tree* T is defined to represent dependencies among attributes of nodes in T. DG_T is obtained by merging together DG_d's according to the decompositions in T.

Let T be a computation tree with root node $X \in M$. DG_T determines an *IO graph* $IO[X, T]$ of X with respect to T. It gives an I/O relationship among attributes of X, which is realized by the decomposition tree T. That is

$$IO[X, T] = (A[X], E_{IO})$$

when an edge (i, s) is in $E_{IO} \subset IN[X] \times OUT[X]$ iff there is in DG_T a path connecting the attribute occurrences $X\downarrow i$ and $X\downarrow s$ of the root T.

For general PLASTIC programs there may be finitely many IO graphs for $X \in M$ and we denote the set of these IO graphs by $IO(X)$, that is

$$IO(X) = \{IO[X, T] \mid T \text{ is a computation tree}\}.$$

Let $IO(X) = \{IO_1, \ldots, IO_N\}$ where $IO_k = (A[X_k], E_k)$. A *superposed IO* graph $IO[X]$ is defined by

$$IO[X] = (A[X], E), \quad E = \bigcup_{k=1}^{N} E_k$$

to represent possible IO relationship.

In order to define a set of attributes to be evaluated in parallel, let us introduce an OI graph the dual concept of IO graph, which specifies how the values of inherited attributes are effected by other attributes.

Let T be a computation tree which contains $X \in M$ as one of its leaf nodes. An *OI graph* $OI[X, T]$ of X with respect to T is given by

$$OI[X, T] = (A[X], E_{OI}[T]), \; E_{OI}[T] \subset A[X] \times IN[X]$$

where $(a, i) \in E_{OI}[T]$ iff there is in DG_T a path from v_a to v_i, where v_a and v_i are nodes for attributes a and i of the leaf node X. A *superposed OI graph* is defined in a similar way as $IO[X]$.

We further define a *dependency graph* $DG[X]$ *of the module* X as the union of IO graph and OI graph, that is

$$DG[X] = (A[X], E_{IO} \cup E_{OI}).$$

For an absolutely noncircular PLASTIC description D a set $O \subset \mathrm{OUT}[X]$ of output attributes is said evaluable in parallel iff no $s_1, s_2 \in O$ are connected in $\mathrm{DG}[X]$.

An augmented dependency DG_d^* for the decomposition d is

$$\mathrm{DG}_d^* = (\mathrm{DV}_d^*, \mathrm{DE}_d^*)$$

where $\mathrm{DV}_d^* = \mathrm{DV}_d$, the set of attribute occurrences in d, and $e \in \mathrm{DE}_d^*$ iff $e \in \mathrm{DE}_d$ or $e = (X_k \cdot i, X_k \cdot s)$ for some $(i, s) \in \mathrm{IO}[X_k]$ and $k = 1, \ldots, n$. DG_d^* represents a relationship among attribute occurrences in d which is realized partly by attribute mappings and partly by computation trees.

A PLASTIC description is said to be absolutely noncircular [2] iff DG_d^* does not contain cycles for any $d \in D$. For an output attribute s of a module X of a PLASTIC program, its input set in $[s, X]$ is defined to be a set of input attributes which are required to evaluate s, that is

$$\mathrm{in}[s, X] = \{i \mid (i, s) \text{ is an edge of } \mathrm{IO}[X]\}.$$

We extend the function in $[s, X]$ to allow such O as its first argument

$$\mathrm{in}[O, X] = \bigcup_{s \in O} \mathrm{in}[s, X].$$

4.2. Translation algorithm

Let X be a module of an absolutely noncircular PLASTIC description $P = (M, m_0, A, D, V, F)$ and s an output attribute of X. We associate with each pair X, s a procedure

$$R_{X,s}(v_1, \ldots, v_m;\ v)$$

where $v_1, \ldots, v_m$ are parameters corresponding to the input attributes in $I = \mathrm{in}[s, X]$ and v is a parameter for s. It should be noted that input and output parameters are separated by semicolon. This procedure is intended to evaluate the output attribute when supplied the values of input attributes in I.

When given the value of the inherited attribute i_0 of the initial module m_0 we begin to evaluate the output attribute s_0 of m_0 by executing the procedure call statement

$$\textbf{call } R_{m_0, s_0}(u_0;\ v_0)$$

where u_0 and v_0 are variables corresponding to i_0 and s_0, respectively.

Now we are ready to describe how to construct the procedure $R_{X,s}(v_1, \ldots, v_m; v)$. The first thing the procedure $R_{X,s}$ must do in its body is to know the decomposition d which is applicable to the module X and perform a sequence $H_{d,s}$ of statements to compute the value of attribute occurrences in d, therefore $R_{X,s}$ is constructed in the following form,

procedure $R_{X,s}(v_1, \ldots, v_m;\ v)$
if C_{d_1} **then** $H_{d_1,s}$ **else**
if C_{d_2} **then** $H_{d_2,s}$ **else**
...
end

where $d_1, d_2, \ldots$ are decompositions (versions) with left side module X. We have assumed that the PLASTIC description is deterministic, that is decomposition conditions of disctinct decompositions with the same left side module do not become simultaneously true for any value of its input attributes.

The sequence $H_{d,s}$ is obtained in the following steps.

(1) Make the augmented dependency graph DG_d^*.

(2) Remove from DG_d^* nodes and edges which are not located on any path leading to $X_0 \downarrow s$ for $i \in I = \text{in}[s, X_0]$. Denote the resulting graph by

$$DG_d^*[s] = (V, E).$$

(3) To each attribute occurrence $x \in V' = V - \{X_0 \downarrow i \mid i \in \text{IN}[X_0]\}$ assign a statement st[x] for evaluating X as follows.

Case 1. If $x = X_k \downarrow i$ for some $i \in \text{IN}[X_k]$ and $k = 1, \ldots, n$ or $x = X_0 \uparrow s (= v)$ for the attributes $s \in \text{OUT}[X_0]$, then st[x] is the assignment statement

$$x := f_{d,x}(z_1, \ldots, z_r)$$

where $f_{d,x}$ is the attribute mapping for the attribute occurrence x and $D_{d,x} = \{z_1, \ldots, z_r\}$.

Case 2. If $x = X_k \uparrow t$ for some $t \in \text{OUT}[X_k]$ and $k = 1, \ldots, n$, then st[x] is the procedure call statement

call $R_{X_k,t}(w_1, \ldots, w_k; x)$

where $$w_1, \ldots, w_k = \{X_k \downarrow i \mid i \in \text{in}[t, X_k]\}.$$

(4) Let $x_1, \ldots, x_N$ be elements in V' which are listed according to the topological ordering determined by E, i.e., if $(x_a, x_b) \in E$ then $a < b$. Then $H_{d,s}$ becomes as follows.

$$\text{st}[x_1]; \ldots; \text{st}[x_N]$$

Note that statements in $H_{d,s}$ satisfy the single assignment rule. It is easy to see that the ordering $x_1, \ldots, x_N$ ensures values of attribute occurrences are determined consistently if the PLASTIC description is absolutely noncircular.

We first construct the procedure R_{m_0,s_0} by the algorithm we have stated. Body of R_{m_0,s_0} may contain calls for other procedures $R_{X,s}$'s and they are constructed in the same way. Repeat this process until no more new procedures appear.

In the case of parallel evaluation we assign a single procedure

$$R_{X,O}(v_1, \ldots, v_m; u_1, \ldots, u_n)$$

to each set O which is evaluable in parallel instead of assigning n procedures, where $u_1, \ldots, u_n$ are parameters corresponding to output attributes in O and $v_1, \ldots, v_m$ are those for attributes in in$[O, X]$.

Construction of $R_{X,O}$ parallels to that of $R_{X,s}$ except a few points. As in the case of $R_{X,s}$, the procedure $R_{X,O}$ has the following form.

procedure $R_{X,O}(v_1, \ldots, v_m; u_1, \ldots, u_n)$
if C_{d_1} **then** $H_{d_1,O}(v_1, \ldots, v_m; u_1, \ldots, u_n)$ **else**
if C_{d_2} **then** $H_{d_2,O}(v_1, \ldots, v_m; u_1, \ldots, u_n)$ **else**
...
end

For a decomposition $X_0 \to X_1 X_2 \ldots X_n$ and $O \in S[X_0]$ which is evaluable in parallel, construction of statement sequence $H_{d,O}$ proceeds in the following steps.

(1) Make DG_d^*.

(2) Make $DG_d^*[O]=(V, E)$ by removing from DG_d^* nodes and edges which are not located on any path leading to $X_0 \downarrow s$ for $s \in O$.

(3) For each $k=1, \ldots, n$ decompose the set

$$OUT^*[X_k] = OUT[X_k] \cap \{t | X_k \cdot t \in V\}$$

into a set of mutually disjoint subsets

$$O_{k1}, O_{k2}, \ldots, O_{kr}$$

such that each O_{kj} is evaluable in parallel. When the decomposition is not unique, we should choose a maximal decomposition, that is, one where the number v becomes minimum, to attain high efficiency of evaluation.

(4) Let $DG'_d[O]=(V', E')$ be a graph obtained from $DG_d^*[O]$ by grouping elements of each O_{kj} into a single node $v_{kj} \in V$. Formally

$$V' = \{g[v] | v \in V\}$$

$$E' = \{(g[u], g[v]) | (u, v) \in E\}$$

where g is a function defined by

$$g[v] = \begin{cases} v_{kj} & \text{if } v = X_k \cdot s \text{ for some } s, k \text{ and } j \text{ such that } s \in O_{kj} \\ v & \text{otherwise.} \end{cases}$$

(5) To each element x in $V_0 = V' - \{X_0 \cdot i | i \in IN[X_0]\}$ assign a statement $st[x]$ as follows.

Case 1. If $X = X_k \downarrow i$ for some $i \in IN[X_k]$ and $k=1, \ldots, n$, or $X = X_0 \uparrow s$ then $st[x]$ is the assignment statement

$$x := f_{d,x}(z_1, \ldots, z_r)$$

where

$$D_{d,x} = \{z_1, \ldots, z_r\}.$$

Case 2. If $X = v_{kj}$ then $st[x]$ is the procedure call statement

$$\textbf{call}\ R_{X_k, O_{kj}}(w_1, \ldots, w_h;\ x_1, \ldots, x_c)$$

where

(1) $$\{w_1, \ldots, w_h\} = \{X_k \downarrow i | i \in in[O_{kj}, X_k]\}$$

and

(2) $$\{x_1, \ldots, x_c\} = \{X_k \uparrow t | t \in O_{kj}\}.$$

(6) Same as 4. for $H_{d,s}$ in the sequential case.

Translation of the entire attribute grammar into the corresponding program is similar to the one given in this section. Let O be a set of output attributes of the initial modul. We start from constructing the procedure $R_{S,O}$ and then proceed to procedures which are called in it.

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

References

[1] DERANSART, P., Logical attribute grammars, *In:* Information Processing' 83, E. Mason ed., North-Holland, 463—470.
[2] GYIMÓTHY, T., E. SIMON, and Á. MAKAY, An implementation of the HLP, Acta Cybernetica 6, 3, (1983), 316—327.
[3] JALILI, F., A general incremental evaluator for attribute grammars, Science of Computer Programming 5, 1 (1985), 83—96.
[4] KATAYAMA, T., Translation of attribute grammar into procedures, Department of Comp. Science, Tokyo Institute of Technology, TR CS-K8001.
[5] KATAYAMA, T., and Y. HOSHINO, Verification of attribute grammars, Department of Comp. Science, Tokyo Institute of Technology, TR CS-K8003.
[6] KNUTH, D. E., Semantics of context-free languages, Math. Syst. Theory 2, 2 (1968), 127—145.
[7] LISKOV, B., et al, Abstraction mechanisms in CLU, CACM 20, 8 (1977), 564—572.
[8] RÄIHÄ, K-J., et al, Revised report on the compiler writing system HLP78, Department of Comp. Science, University of Helsinki, TR A-1983-1.
[9] SHAW, M., et al, Abstraction and verification in ALPHARD: Defining and specifying iteration and generators, CACM 20, 8 (1977), 553—563.
[10] SIMON, E., Formal definition of the SYCOMAP system, *In:* Preprints of the Second Hungarian Computer Science Conference, Budapest, 1977, 738—759.
[11] SIMON, E., Language extension in the HLP/SZ system, Acta Cybernetica 7, 1 (1984), 89—97.
[12] TOCZKI, J., et al., On the PASCAL implementation of the HLP, to be published *In:* Proc. of 4th Hungarian Computer Science Conference, Győr, 1985 (to appear).
[13] YELLIN, D., and Eva-Maria. M. Mueckstein, Two-way translators based on attribute grammar inversion, to be published *In:* Proc. of 8th International Conference on Software Engineering, 1985, 17 pp.

(Received Aug. 27, 1985.)

Acta Cybernetica, Tom. 7, Fasc. 4, Szeged, 1986.

Problem solving based on knowledge representation and program synthesis

S. S. LAVROV

Computers cannot solve problems, they are only able to execute programs. A man can solve problems if he has necessary knowledge and experience. In informatics to solve a problem means to find and to describe a sequence of computing operations leading to the intended result. If a problem is solved in that sense then a program capable to get an answer to that single problem is created. Possibly the program can supply solutions of a number of problems differing however only in their input data.

A man can usually do more than this. He knows the field of his activity — an object area as we shall call it. He is able to solve many essentially different problems in this area.

From this standpoint a challenging problem arises — how to transfer human's knowledge and experience to a computer, how to make it capable to solve a large class of problems, not just one, in a specific area. The problem is by no means a new one. It is known a number of ways to solve it.

These traditional ways are: program packages (if one takes an algorithmic approach), data bases (when an informational approach is preferable), expert systems. Every program package or expert system usually has its own built-in control device. In the case of data bases this role is played by a data base management system. Such a system enables us to create different data bases oriented to different object areas. A similar approach is known in connection with expert systems.

There exist also systems based on more abstract form of knowledge representation. Among them the language PROLOG [1] and its implementations and applications should be mentioned first of all.

However in all these cases we have one large program or system which directly uses a computer to solve various problems. The system does not try to generate a program for each specified problem. In other words all these systems are rather interpretive than compilative by their nature.

Only one form of knowledge representation is used in every kind of system. This form is: a computing procedure or a program module in the case of a program package, a table in the case of a data base, a rule of the form "condition→action" in the case of an expert system, a Horn clause in the case of a PROLOG program.

It occurs sometimes that two very similar application systems having almost the same purpose and possibilities are classified differently by their authors, e.g. as an expert system and a data base, depending on the authors' tastes and points of view.

A very interesting problem solving system called PRIZ was developed in Tallinn by E. H. Tyugu and his colleages as early as the first half of 60-ies [2—4]. The system has very much progressed since then of cause. The work which will be reported here was inspired in many aspects by this system.

The approach adopted in the informatics division of our institute is intended to overcome the drawbacks and the restrictions mentioned above. In our system called SPORA ("СПОРА") we wanted to develop a unified approach combining the principles accepted by the designers of many application systems. On the other hand we tried not to mix up different concepts. Moreover our intention was to find the most appropriate place in the system for every independent concept known. We wanted to use any such concept with maximal effect.

Similarly we tried to take the greatest possible advantage of different experience and tastes of different people working in any chosen area. There are always people with strong mathematical attitude and a good mathematical education. There are people also who like computer programming and are eager to contact, to cooperate with a computer. Surely most people are using computers only by necessity because without them they could not reach the desirable result.

Starting from all these considerations we have built our system in the following way.

The main part of the system is a knowledge base. We distinguish at least three kinds of knowledge: conceptual, algorithmic and factual ones. Conceptual knowledge is a set of terms (words) naming the basic concepts or notions of a given object area together with their properties and relationships. All this is expressed on the most abstract level. A description of an object area on this level is called conceptual model of the area. Abstraction is made from physical representation of entities (i.e. from their measurement units), from their programming representation (possible data types in some algorithmic language) and even from their mathematical representation. E.g. we prefer to write the Ohm law in the form

Ohm (voltage, current, resistance)

or

voltage=times (current, resistance)

where the word "times" denotes a map with no predefined mathematical properties instead of the usual $u=i*r$.

A conceptual model contains some entity types and functional dependencies (maps) called primary. They are just names and on the abstract level do not posses any directly stated properties. However we consider the possibility to add a sort of axiomatics describing such properties to a model.

The model also contains secondary types and maps which are described in a relational manner. The components of an object, i.e. the attributes of a type or the arguments and the result of a map should be explicitly listed. In addition to this the dependencies between the components should be described.

There are tree kinds of dependencies. A functional dependency has the form $v:=t$ where v is a component of an object, t is a functional term constructed from such components. Such a dependency prescribes the value of v to be computed as a result of the term t evaluation.

An equational dependency (or simply an equation) having the form $t_1 = t_2$ prescribes values of functional terms t_1 and t_2 to be equal.

A relational dependency looks like this:

$$\text{is } R_1(v_{i_1} = t_1, \ldots, v_{i_k} = t_k)$$

where R_1 is another type described elsewhere, $v_{i_1}, \ldots, v_{i_k}$ are some of the attributes of R_1 and $t_1, \ldots, t_k$ are functional terms constructed from the components of the currently defined object. Such a dependency allows one to use the dependencies associated with the type R_1 in the actual definition. It means that an object with the components $v_{i_1}, \ldots, v_{i_k}$ having values supplied by the terms $t_1, \ldots, t_k$ must belong to the type R_1.

There is a possibility for an object to have optional components and conditional dependencies between them. The definition of either a type or a map may be recursive.

This approach has his pros and cons. The main gain is that abstraction from many details usually opens the shortest and the most natural way to a solution of the given problem. The main drawback is that a solution (if one is found) is an abstract one and cannot be directly used for computation. Another difficulty arises from the fact that equational dependencies cannot be resolved on the abstract level. This is so because on that level we abstract from the mathematical representation of primary maps and cannot use their mathematical properties.

Therefore an abstract conceptual model of an object area needs an algorithmic and informational support. At this point two other kinds of knowledge mentioned above step on the scene.

Algorithmic knowledge is a collection of ways to represent each entity as an object of some algorithmic language and each map as a procedure written in such a language. Such a representation is needed only for the entities of primary types and for primary maps. The secondary types and maps have a standard representation based on their desription sketched above. The algorithmic languages used for the representation of algorithmic knowledge are called base languages of the system. Currently base languages are Pascal, FORTRAN and ALGOL 60. The system itself is written mainly in Pascal and partially in the assembly language of the BESM 6 computer.

Factual knowledge is a set of values and qualitative characteristics of objects under consideration. The most natural way to represent the factual knowledge is to put it in a data base. The data manipulation language of the data base is also considered as a base language of the system.

Thus the whole knowledge base consists of a conceptual model, a program modules package forming the algorithmic support of the model and a data base forming its informational support. Certainly an interface between these three parts of the knowledge base should be described. We consider both the program package and the data base equally important parts of the knowledge base having equal rights and status.

Our problem solving system has a number of input languages oriented to different needs and to different classes of users.

The most complicate and still rather simple language is the language for object area description. People using this language to construct conceptual models should be experts in the object area. They must be mathematically educated as well. We call them model designers. They are working in close contact with (if not being the

same) people creating algorithmic and informational support of the model, i.e., program packages and data bases.

To describe the interface between the model and its support one uses another input language called representation language or more exactly — primary types and maps representation language. This language is essentially a kind of universal macro-language. For each primary data type one has to describe a way to translate a name of an object having this type into a base language construction. For each primary map a way to call corresponding procedure should be described.

Users of the representation language should be good programmers first of all. Their main task is to describe the interface between the conceptual model and its algorithmic and informational support. Both program modules and contents of a data base included in the support may be written by other people. The use of a macro-language allows one to include arbitrary modules or data collections in the conceptual model support.

A knowledge base of an object area being constructed, it is a rather simple task to specify a problem from the area. Essentially one has to list all the input quantities and the desirable result, in other words, to point out the places which these data occupy in the model. Additionally if the data base part does not contain the numerical values of some input quantities one has to supply the system with these values.

All this information may be expressed in a rather simple language called request language and oriented to the most numerous category of users, viz. terminal ones.

A closer consideration reveals however a gap between the conceptual model and a problem specification existing in that scheme of problem solving. In fact there may be no places for the input data and the result of a problem in the model. Let us take geometry as an example of an object area. Its model contains such notions as a point, a straight line, a triangle, a circle, a distance, an angle and so on, such relations as the incidence of a point and a line, the tangency of a line to a circle etc. To state a problem however one needs a collection of objects having various relationships between themselves to be described or drafted. The conceptual model of geometry cannot contain all such intented collections.

Therefore before stating a problem one has to describe a more or less concrete object on which the problem will be stated. We meet here another kind of knowledge which may be called a constructive one. Constructive knowledge is a set of rules and methods allowing one to describe an object under investigation on the basis of a conceptual model. A number of problems may be posed with respect to the object.

In our system we have a device useful by itself which can serve this purpose. The device is that of submodels related to a given model. Indeed the means to describe an investigated object are essentially the same as those used in the description of any secondary object.

To conclude a couple of words about the system functioning should be said.

A problem specification together with the model of the corresponding object area (or part of the model) is translated into a logical language. The result of the translation is an existence theorem for a solution of the given problem. Then the system tries to prove the theorem. From the proof if one is found the system extracts an algorithm leading to a solution of the problem. We call this algorithm an abstract program because it is expressed in terms of the conceptual model.

Next the abstract program is translated into one of the base languages of the system. The interface between the model and its support is also used on this stage.

The resulting base language program can be compiled into machine code and then be executed by computer. The input data for the computation can be either taken from the data base or given by the author of the problem specification.

Variants of the scheme just described are possible. The abstract program can be printed for examination by the user instead of being processed further. If the problem stated is of a rather general nature then the abstract program can be added to the conceptual model while its translation into a base language is added to the algorithmic support of the model. The compiled program can be included into a library. Thus not only the final result of computation but all the intermediate ones starting from the abstract program can be considered as a solution of the problem.

When one of the steps described above fails the user gets a diagnostic message.

The first version of the system SPORA was written in 1977—82. Recently the second version was developed and tested.

References

[1] Van Emden, M. E., Kowalsky R. A., The semantics of predicate logic as a programming language. — J. of the ACM, 1976, v. 23, N 4, p.p. 23—32.

[2] Tyugu, E. H., A data base and problem solver for computer-aided design. — *In:* Information Processing 71. — North-Holland Publ. Co., 1972, p.p. 1046-49.

[3] Tyugu, E., Towards practical synthesis of programs. — *In:* Information Processing 80. — North-Holland Publ. Co., 1980, p.p. 207—19.

[4] *Тыугу Э. Х.* Концептуальное программирование. — Москва, изд—во „Наука", 1984.

(Received Sept. 26, 1985.)

ИНСТИТУТ ТЕОРЕТИЧЕСКИЙ АСТРОНОМИИ
АКАДЕМИИ НАУК СССР
191187 ЛЕНИНГРАД
НАБЕРЕЖНАЯ КУТУЗОВА, 10
СССР

Acta Cybernetica, Tom. 7, Fasc. 4, Szeged, 1986.

On compositions of root-to-frontier tree transformations

By S. VÁGVÖLGYI

0. Introduction

It is well known that the family of (nondeterministic) root-to-frontier tree transformations is not closed with respect to the composition, see [2]. In this paper we introduce the notion of k-synchronized root-to-frontier tree transducer. Transducers of this type are capable of inducing all the relations which are compositions of k root-to-frontier tree transformations. Conversely, we shall show that any relation induced by a k-synchronized tree transducer is a composition of k root-to-frontier tree transformations. We mention that similar results are obtained by M. Dauchet in his dissertation [1] using the theory of magmoids.

1. Preliminaires

In this chapter we shall review the basic notions and notations used in the paper and give a reformalized notation of root-to-frontier tree transducers.

Definition 1.1. An operator domain is a set G together with a mapping $\nu: G \to \{0, 1, 2, \ldots\}$ that assigns to every $g \in G$ an arity, or rank, $\nu(g)$. For any $m \geqq 0$,

$$G^m = \{g \in G \mid \nu(g) = m\}$$

is the set of m-ary operators.

From now on, by an operator domain we mean a finite one, that means G is a finite set. The letters F and G always denote operator domains.

Definition 1.2. Let Y be a set disjoint from the operator domain G. The set $T_G(Y)$ of G-trees over Y is defined as follows:

(1) $G^0 \cup Y \subseteq T_G(Y)$,

(2) $g(p_1, \ldots, p_m) \in T_G(Y)$ whenever $m \geqq 1$, $g \in G^m$ and $p_1, \ldots, p_m \in T_G(Y)$, and

(3) every G-tree over Y can be obtained by applying the rules (1) and (2) a finite number of times.

The set $T \subseteq T_G(Y)$ is called a G-forest over Y.

Definition 1.3. Let $p \in T_G(Y)$ be a G-tree over Y. The set sub (p) of subtrees of p is defined by the following rules:

(1) $\text{sub}(p) = \{p\}$ if $p \in G^0 \cup Y$,

(2) $\text{sub}(p) = \{p\} \cup \bigcup_{i=1}^{m} \text{sub}(p_i)$ if

$p = g(p_1, \ldots, p_m)$, $g \in G^m$ and

$p_1, \ldots, p_m \in T_G(Y)$.

Definition 1.4. Let $p \in T_G(Y)$ be a tree. The root root (p) and height $h(p)$ are defined as follows:

(1) If $p \in G^0 \cup Y$, then root $(p) = p$, $h(p) = 0$.

(2) If $p = g(p_1, \ldots, p_m)$ $(m > 0)$, then root $(p) = g$ and $h(p) = \max(h(p_i) | i = 1, \ldots, m) + 1$.

Definition 1.5. Let $u \in N^*$ be a word over the set of natural numbers. The word u induces a partial function $u: T_G(Y) \to T_G(Y)$ in the following way:

(1) If $u = e$ then $u(p) = p$ for every $p \in T_G(Y)$, where e denotes the empty word.

(2) If $u = iv$, $i \in N$, $v \in N^*$ and $p \in T_G(Y)$, then

$$u(p) = \begin{cases} v(p_i) & \text{if } p = g(p_1, \ldots, p_m),\ g \in G^m,\ 1 \leq i \leq m \\ \text{else undefined.} \end{cases}$$

The elements of $T_G(Y)$ may be visualized as tree like directed ordered labelled graphs. In this case every path from the root to a given node in the graph is determined by a word over N. For every word $u \in N^*$, if there exists a node r such that u is the path from the root of p to r, then $u(p)$ denotes the subtree (subgraph) with root r.

Definition 1.6. Let Y be a set disjoint from G. We may assume without loss of generality that $N^* \cap T_G(Y) = \emptyset$ and $G \cap N^* = \emptyset$ hold in the rest of the paper. The set $P_G(Y)$ of quasi G-trees over Y is defined by the following rule:

$$P_G(Y) = \{p \in T_G(Y \cup N^*) | \forall u \in N^* \text{ if } u(p) \in N^* \text{ then } u(p) = u\}.$$

Definition 1.7. The mapping $S: P_G(Y) \to 2^{N^*}$ assigns a subset $S(p)$ of N^* to every quasi tree p which is defined by

$$S(p) = \{u(p) | u \in N^*\} \cap N^*.$$

It is clear that $S(p)$ is a finite set for every $p \in P_G(Y)$. The set $S(p)$ is also denoted by S_p. Members of S_p are called arguments of p.

Definition 1.8. Let Z be an arbitrary set and let $\varphi: S_p \to Z$ be a given function for a given quasi tree $p \in P_G(Y)$. Replacing every element u of S_p by $\varphi(u)$ in the tree p we obtain a G-tree over $Y \cup Z$, which is denoted by $p[S_p, \varphi]$.

Example. Let $G = \{g_1, g_2\}$ be an operator domain with $\nu(g_1) = 1$, $\nu(g_2) = 2$ and let $Y = \{y_1, y_2, y_3\}$. The quasi tree $p = g_2(g_1(11), g_2(21, y_1))$ may be visualized by the graph on Fig. 1.

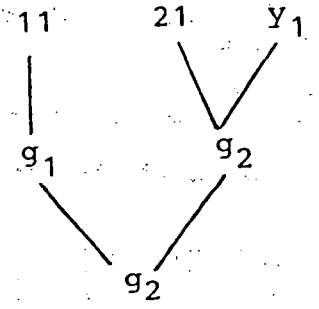


Figure 1

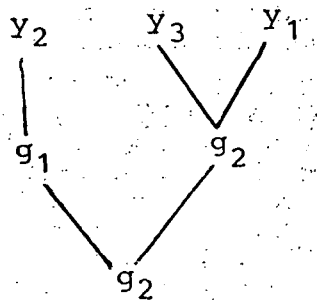


Figure 2

Let us define the mapping φ: $\{11, 21\} \to \{y_2, y_3\}$ as follows:

$$\varphi(11) = y_2, \varphi(21) = y_3.$$

The quasi tree $p[S_p, \varphi]$ may be visualized by the graph on Fig. 2.

Binary relations $\tau \subseteq T_F(X) \times T_G(Y)$ are called tree transformations. The composition $\tau_1 \circ \tau_2$ of the tree transformations $\tau_1 (\subseteq T_F(X) \times T_G(Y))$ and $\tau_2 (\subseteq T_G(Y) \times \times T_H(Z))$ is defined by

$$\tau_1 \circ \tau_2 = \{(p, q) | (p, r) \in \tau_1, (r, q) \in \tau_2 \text{ for some } r\}.$$

The composition $\tau_1 \circ \tau_2 \circ \ldots \circ \tau_l$ $(l \geqq 3)$ of the tree transformations $\tau_1, \tau_2, \ldots, \tau_l$ is defined by

$$\tau_1 \circ \tau_2 \circ \ldots \circ \tau_l = (\tau_1 \circ \ldots \circ \tau_{l-1}) \circ \tau_l.$$

Definition 1.9. A state set A is an operator domain consisting of unary operators only. If A is a state set and D is an arbitrary set then AD will denote the forest

$$AD = \{a(d) | a \in A, d \in D\}.$$

Moreover, if $a \in A$ and $d \in D$ then we generally write ad for $a(d)$.

If $A_1, \ldots, A_j$ are state sets $(j \in N)$ then $A_j \ldots A_1$ denotes the state set $A_j \times \ldots \times A_1$ which is the Cartesian product of the sets A_i $(1 \leqq i \leqq j)$.

Elements of $A_j \ldots A_1$ are denoted by sequences $a_j \ldots a_1$, where $a_i \in A_i$, $i = 1, \ldots, j$.

For every non-negative integer l, $\{1, \ldots, l\}$ denotes the set $\{i | 1 \leqq i \leqq l\}$.

Definition 1.10. A root-to-frontier tree transducer (R-transducer) is a system $\mathfrak{A} = (F, X, A, G, Y, A', \Sigma)$, where

(1) F and G are operator domains.

(2) A is an operator domain consisting of unary operators, the state set of $\mathfrak{A}$. (It will be assumed that $A \cap T_F(X) = \emptyset$ and that $A \cap T_G(Y) = \emptyset$.)

(3) X and Y are finite sets.

(4) $A' \subseteq A$ is the set of initial states.

(5) Σ is a finite set of productions (rewriting rules) of the following two types:

(i) $ax \to q(a \in A, x \in X, q \in T_G(Y))$,

(ii) $af \to q[S_q, \varphi](q \in P_G(Y), f \in F^m, \varphi: S_q \to A\{1, \ldots, m\})$.

$\mathfrak{A}$ is said to be a deterministic R-transducer if A' is a singleton and there are no distinct productions in Σ with the same left-hand side.

Definition 1.11. Let $\mathfrak{A}=(F, X, A, G, Y, A', \Sigma)$ be an R-transducer and let $p_1, p_2 \in T_G(Y \cup N^* \cup AT_F(X \cup N^*))$ be trees. We say that p_1 directly derives p_2 in $\mathfrak{A}$, in symbols $p_1 \Rightarrow_{\mathfrak{A}} p_2$, if p_2 can be obtained from p_1 by

(i) replacing an occurrence of a subtree $ax (\in AX)$ in p_1 by the right side q of a production $ax \to q$ in Σ, or by

(ii) replacing an occurrence of a subtree $af(1, \ldots, m)[\{1, \ldots, m\}, \alpha] (f \in F^m, \ \alpha: \{1, \ldots, m\} \to T_F(X \cup N^*))$ in p_1 by $q[S_q, \beta]$, where $af \to q[S_q, \varphi]$ is in Σ and β is a mapping $\beta: S_q \to AT_F(X \cup N^*)$ such that for each $s \in S_q$ if $\varphi(s) = ct (c \in A, t \in \{1, \ldots, m\})$ then $\beta(s) = c\alpha(t)$.

Each application of steps (i) and (ii) is called a direct derivation in $\mathfrak{A}$.

The reflexive-transitive closure of $\Rightarrow_{\mathfrak{A}}$ is denoted by $\Rightarrow^*_{\mathfrak{A}}$.

Using the notation $\Rightarrow^*_{\mathfrak{A}}$ the transformation $\tau_{\mathfrak{A}}$ induced by a root-to-frontier tree transducer $\mathfrak{A}=(F, X, A, G, Y, A', \Sigma)$ is defined by:

$$\tau_{\mathfrak{A}} = \{(p, q) \mid p \in T_F(X), \ q \in T_G(Y), \ ap \Rightarrow^*_{\mathfrak{A}} q \text{ for some } a \in A'\}.$$

The range of a mapping $\varphi: A \to B$ is denoted by $\mathrm{rg}(\varphi)$. Let $U_0, U_1, \ldots, U_l$ be sets, and let V be a subset of the set $(U_0 \times U_1 \times \ldots \times U_l) \cup (U_0 \times U_1 \times \ldots \times U_{l-1}) \cup \ldots \ldots \cup (U_0 \times U_1) \cup U_0$, where $U_0 \times U_1 \times \ldots \times U_l$ the Cartesian product of the sets U_i $(0 \leqq i \leqq l)$. Then for an index j, $(0 \leqq j \leqq l)$ $[V]_j$ denotes the set

$$\{u_j \mid \exists (u_0, \ldots, u_j, \ldots, u_n) \in V, \quad 0 \leqq n \leqq l, \quad 0 \leqq j \leqq n)\}.$$

Definition 1.12. Let u be an element of N^*. The mapping $\omega_u: T_G(Y \cup N^*) \to T_G(Y \cup N^*)$ is defined as follows:

(1) $\omega_u(p) = p$ if $p = y (\in Y)$ or $p = f (\in G^0)$,

(2) $\omega_u(p) = up$ if $p \in N^*$,

(3) $\omega_u(p) = f(\omega_u(p_1), \ldots, \omega_u(p_l))$ if

$p = f(p_1, \ldots, p_l), \ f \in G^l, \ l \geqq 1, \ p_i \in T_G(Y \cup N^*), \quad i = 1, \ldots, l.$

2. Derivation sequences

In this chapter we shall deal with the description of derivations according to root-to-frontier tree transducers.

In the rest of the paper k denotes a natural number, not less than two, moreover let $\mathfrak{A}_i = (G_{i-1}, Y_{i-1}, A_i, G_i, Y_i, A_i', \Sigma_{\mathfrak{A}_i})$ be R-transducers, $1 \leqq i \leqq k$.

Now we give a procedure P. The input of P is a derivation in the form

$$(1) \quad a_j p_{j-1} \Rightarrow^*_{\mathfrak{A}} p_j \ (a_j \in A_j, \ p_{j-1} \in T_{G_{j-1}}(Y_{j-1}), \ p_j \in T_{G_j}(Y_j))$$

for some $j \in \{1, \ldots, k\}$ and a decomposition

$$p_{j-1} = r_{j-1}[S_{r_{j-1}}, \varphi_{j-1}] \ (r_{j-1} \in P_{G_{j-1}}(Y_{j-1}), \ \varphi_{j-1}: S_{r_{j-1}} \to T_{G_{j-1}}(Y_{j-1})).$$

The procedure P produces two derivations denoted by (2) and (3) which are defined by induction on the height of $r_{j-1}(\in T_{G_{j-1}}(Y_{j-1}\cup N^*))$. The derivations (2) and (3) will have the following forms:

(2) $a_j r_{j-1}[S_{r_{j-1}}, \varphi_{j-1}] \Rightarrow^*_{\mathfrak{A}_j} r_j[S_{r_j}, \psi_j] \Rightarrow^*_{\mathfrak{A}_j} r_j[S_{r_j}, \varphi_j] = p_j,$

$$(\psi_j\colon S_{r_j} \to A_j \operatorname{rg}(\varphi_{j-1}), \varphi_j\colon S_{r_j} \to T_{G_j}(Y_j)),$$

for each

$$s_j \in S_{r_j}, \psi_j(s_j) \Rightarrow^*_{\mathfrak{A}_j} \varphi_j(s_j)$$

holds,

(3) $a_j r_{j-1} \Rightarrow^*_{\mathfrak{A}_j} r_j[S_{r_j}, \bar{\psi}_j], (\bar{\psi}_j\colon S_{r_j} \to A_j S_{r_{j-1}}),$

and for each $s_j \in S_{r_j}$ if $\bar{\psi}_j(s_j) = a_j s_{j-1}$ then $\psi_j(s_j) = a_j \varphi_{j-1}(s_{j-1})$ holds.

Let $h(r_{j-1}) = 0$.

Case 1. $r_{j-1} = f$, $f \in G^0_{j-1}$. In this case $S_{r_{j-1}} = \emptyset$, $\varphi_{j-1} = \emptyset$ and $r_{j-1}[S_{r_{j-1}}, \varphi_{j-1}] = f$. Thus $a_j f \to p_j \in \Sigma_{\mathfrak{A}_j}$, where $p_j \in T_{G_j}(Y_j)$. Let $r_j = p_j$, thus $S_{r_j} = \emptyset$. Let $\varphi_j = \emptyset$, $\psi_j = \emptyset$, $\bar{\psi}_j = \emptyset$. Thus the derivation (1) takes the following forms:

(2) $a_j r_{j-1}[S_{r_{j-1}}, \varphi_{j-1}] \Rightarrow^*_{\mathfrak{A}_j} r_j[S_{r_j}, \psi_j] \Rightarrow^*_{\mathfrak{A}_j} r_j[S_{r_j}, \varphi_j],$

(3) $a_j r_{j-1} \Rightarrow_{\mathfrak{A}_j} r_j[S_{r_j}, \bar{\psi}_j].$

Case 2. $r_{j-1} = y (\in Y_{j-1})$. This case is the same as Case 1.

Case 3. $r_{j-1} = e (\in N^*)$. In this case $\varphi_{j-1}(e) = r_{j-1}[S_{r_{j-1}}, \varphi_{j-1}]$. Let $r_j = e$, thus $S_{r_j} = \{e\}$. Let the mappings

$$\psi_j\colon S_{r_j} \to A_j T_{G_{j-1}}(Y_{j-1}), \bar{\psi}_j\colon S_{r_j} \to A_j S_{r_{j-1}}$$

and

$$\varphi_j\colon S_{r_j} \to T_{G_j}(Y_j)$$

be defined as follows:

$$\psi_j(e) = a_j r_{j-1}[S_{r_{j-1}}, \varphi_{j-1}], \bar{\psi}_j(e) = a_j e, \varphi_j(e) = p_j.$$

Thus

$$r_j[S_{r_j}, \psi_j] = a_j r_{j-1}[S_{r_{j-1}}, \varphi_{j-1}], r_j[S_{r_j}, \bar{\psi}_j] = a_j e.$$

Thus we have obtained the desired derivations (2) and (3), and

$$\bar{\psi}_j(e) = a_j e, \psi_j(e) = a_j \varphi_{j-1}(e), \quad \text{where} \quad S_{r_j} = \{e\}.$$

We have proved the basic step of the induction. Let

$$r_{j-1} = f(\bar{p}_1, \ldots, \bar{p}_l) = f(\omega_1(p_1), \ldots, \omega_l(p_l))$$

$$(p_1, \ldots, p_l \in P_{G_{j-1}}(Y_{j-1}), S_{r_{j-1}} = 1 \cdot S_{p_1} \cup 2 \cdot S_{p_2} \cup \ldots \cup l \cdot S_{p_l},$$

where

$$i \cdot S_{p_i} = \{is \mid s \in S_{p_i}\}, \ i = 1, \ldots, l).$$

$$r_{j-1}[S_{r_{j-1}}, \varphi_{j-1}] = f(p_1[S_{p_1}, \mu_1], \ldots, p_l[S_{p_l}, \mu_l]),$$

where for each $i \in \{1, \ldots, l\}$, and $s \in S_{p_i}$ $\mu_i(s) = \varphi_{j-1}(is)$ holds. The production applied in the first step in derivation (1) must be of the form $a_j f \to q[S_q, \varepsilon]$, where

$q \in P_{G_j}(Y_j)$, $f \in G^l_{j-1}$ for some l, $\varepsilon\colon S_q \to A_j\{1, \ldots, l\}$. Consequently derivation (1) can be written in the following form:

$$a_j r_{j-1}[S_{r_{j-1}}, \varphi_{j-1}] \Rightarrow^*_{\mathfrak{A}_j} q[S_q, \varrho] \Rightarrow^*_{\mathfrak{A}_j} q[S_q, \tau]$$

$$(\varrho\colon S_q \to A_j \operatorname{rg}(\varphi_{j-1}), \tau\colon S_q \to T_{G_j}(Y_j)),$$

where the mapping ϱ satisfies the following formula: for every $s \in S_q$ if $\varepsilon(s) = b_j t$ $(1 \leqq t \leqq l, b_j \in A_j)$ then $\varrho(s) = b_j p_t[S_{p_t}, \mu_t]$. This implies that $\varrho(s) = b_j p_t(S_{p_t}, \mu_t] \Rightarrow^*_{\mathfrak{A}_j} \Rightarrow^*_{\mathfrak{A}_j} \tau(s)$ holds. The desired derivations will take the following forms:

(2) $a_j f(p_1[S_{p_1}, \mu_1], \ldots, p_l[S_{p_l}, \mu_l]) \Rightarrow_{\mathfrak{A}_j} q[S_q, \varrho] \Rightarrow^*_{\mathfrak{A}_j} q[S_q, \varkappa] \Rightarrow^*_{\mathfrak{A}_j} q[S_q, \tau]$,

where

$$\varkappa\colon S_q \to T_{G_j}(Y_j \cup A_j T_{G_{j-1}}(Y_{j-1})),$$

$$\tau\colon S_q \to T_{G_j}(Y_j).$$

(3) $a_j f(\omega_1(p_1), \ldots, \omega_l(p_l)) \Rightarrow_{\mathfrak{A}_j} q[S_q, \bar{\varrho}] \Rightarrow^*_{\mathfrak{A}_j} q[S_q, \bar{\varkappa}]$,

where

$$\bar{\varrho}\colon S_q \to A_j T_{G_{j-1}}(Y_{j-1} \cup N^*), \quad \bar{\varkappa}\colon S_q \to T_{G_j}(Y_j \cup A_j S_{q_{j-1}}).$$

We shall define the mappings $\varkappa, \bar{\varrho}, \bar{\varkappa}$. For each $s \in S_q$ let us consider the derivation

(4) $\varrho(s) = b_j p_t[S_{p_t}, \mu_t] \Rightarrow^*_{\mathfrak{A}_j} \tau(s)$,

where

$\varepsilon(s) = b_j t$ holds.

Since $h(p_t) < h(r_{j-1})$ we may apply the induction hypothesis to derivation (4) and decomposition $p_t[S_{p_t}, \mu_t]$. The derivations (5) and (6) are obtained by applying procedure P to (4) and decomposition $p_t[S_{p_t}, \mu_t]$.

(5) $b_j p_t[S_{p_t}, \mu_t] \Rightarrow^*_{\mathfrak{A}_j} q_s[S_{q_s}, \eta_s] \Rightarrow^*_{\mathfrak{A}_j} q_s[S_{q_s}, \xi_s] = \tau(s)$,

$(q_s \in P_{G_j}(Y_j), \eta_s\colon S_{q_s} \to A_j \operatorname{rg}(\mu_t), \xi_s\colon S_{q_s} \to T_{G_j}(Y_j))$,

such that for every $v \in S_{q_s}$, $\eta_s(v) \Rightarrow^*_{\mathfrak{A}_j} \xi_s(v)$ holds.

(6) $b_j p_t \Rightarrow^*_{\mathfrak{A}_j} q_s[S_{q_s}, \bar{\eta}_s]$ $(\bar{\eta}_s\colon S_{q_s} \to A_j S_{p_t})$,

and for every $v \in S_{q_s}$ if $\bar{\eta}_s(v) = b_j z$ for some $b_j \in A_j$ and $z \in S_{p_t}$ then $\eta_s(v) = b_j \mu_t(z)$.

In this case $\varkappa, \bar{\varrho}, \bar{\varkappa}$ are defined by

$$\varkappa(s) = q_s[S_{q_s}, \eta_s], \quad \bar{\varrho}(s) = \omega_t(b_j p_t),$$

$$\bar{\varkappa}(s) = \omega_t(q_s[S_{q_s}, \bar{\eta}_s]).$$

The derivation $\varrho(s) \Rightarrow^*_{\mathfrak{A}_j} \varkappa(s) \Rightarrow^*_{\mathfrak{A}_j} \tau(s)$ is the same as derivation (5).

The derivation $\bar{\varrho}(s) \Rightarrow^*_{\mathfrak{A}_j} \bar{\varkappa}(s)$ is obtained from derivation (6) by applying the mapping ω_t to each step of derivation (6).

We give a procedure S. The input of S is a derivation sequence $D=D_1, \ldots, D_k$ given in the following form:

$$D_1\colon a_1 p_0 \Rightarrow^*_{\mathfrak{A}_1} p_1,\ (p_0\in T_{G_0}(Y_0),\ a_1\in A_1,\ p_1\in T_{G_1}(Y_1)),$$

$$D_2\colon a_2 p_1 \Rightarrow^*_{\mathfrak{A}_2} p_2,\ (a_2\in A_2,\ p_2\in T_{G_2}(Y_2)),$$

$$\vdots$$

$$D_k\colon a_k p_{k-1} \Rightarrow^*_{\mathfrak{A}_k} p_k (a_k\in A_k,\ p_k\in T_{G_k}(Y_k))$$

and a decomposition $p_0=r_0[S_{r_0}, \varphi_0]$. S produces two derivation sequences denoted by $D^{r_0}=D_1^{r_0}, D_2^{r_0}, \ldots, D_k^{r_0}$ and $\bar{D}^{r_0}=\bar{D}_1^{r_0}, \bar{D}_2^{r_0}, \ldots, \bar{D}_k^{r_0}$. The derivation sequences D^{r_0} and $\bar{D}^{r_0}$ will have the following forms:

$$D_1^{r_0}\colon a_1 r_0[S_{r_0}, \varphi_0] \Rightarrow^*_{\mathfrak{A}_1} r_1[S_{r_1}, \psi_1] \Rightarrow^*_{\mathfrak{A}_1} r_1[S_{r_1}, \varphi_1] =$$

$$= p_1\ (r_1\in P_{G_1}(Y_1),\ \psi_1\colon S_{r_1}\to A_1 \operatorname{rg}(\varphi_0),\ \varphi_1\colon S_{r_1}\to T_{G_1}(Y_1))$$

and for each $s_1\in S_{r_1}$ the derivation $\psi_1(s_1)\Rightarrow^*_{\mathfrak{A}_1}\varphi_1(s_1)$ is valid,

$$D_2^{r_0}\colon a_2 r_1[S_{r_1}, \varphi_1] \Rightarrow^*_{\mathfrak{A}_2} r_2[S_{r_2}, \psi_2] \Rightarrow^*_{\mathfrak{A}_2} r_2[S_{r_2}, \varphi_2] =$$

$$= p_2\ (r_2\in P_{G_2}(Y_2),\ \psi_2\colon S_{r_2}\to A_2 \operatorname{rg}(\varphi_1),\ \varphi_2\colon S_{r_2}\to T_{G_2}(Y_2))$$

and for each $s_2\in S_{r_2}$ the derivation $\psi_2(s_2)\Rightarrow^*_{\mathfrak{A}_2}\varphi_2(s_2)$ is valid,

$$\vdots$$

$$D_k^{r_0}\colon a_k r_{k-1}[S_{r_{k-1}}, \varphi_{k-1}] \Rightarrow^*_{\mathfrak{A}_k} r_k[S_{r_k}, \psi_k] \Rightarrow^*_{\mathfrak{A}_k} r_k[S_{r_k}, \varphi_k] =$$

$$= p_k(r_k\in P_{G_k}(Y_k),\quad \psi_k\colon S_{r_k}\to A_k \operatorname{rg}(\varphi_{k-1}),\ \varphi_k\colon S_{r_k}\to T_{G_k}(Y_k))$$

and for each $s_k\in S_{r_k}$ the derivation $\psi_k(s_k)\Rightarrow^*_{\mathfrak{A}_k}\varphi_k(s_k)$ is valid.

$$\bar{D}_1^{r_0}\colon a_1 r_0 \Rightarrow^*_{\mathfrak{A}_1} r_1[S_{r_1}, \bar{\psi}_1](\bar{\psi}_1\colon S_{r_1}\to A_1 S_{r_0}),$$

$$\bar{D}_2^{r_0}\colon a_2 r_1 \Rightarrow^*_{\mathfrak{A}_2} r_2[S_{r_2}, \bar{\psi}_2](\bar{\psi}_2\colon S_{r_2}\to A_2 S_{r_1}),$$

$$\vdots$$

$$\bar{D}_k^{r_0}\colon a_k r_{k-1} \Rightarrow^*_{\mathfrak{A}_k} r_k[S_{r_k}, \bar{\psi}_k](\bar{\psi}_k\colon S_{r_k}\to A_k S_{r_k}).$$

For every $j\in\{1, \ldots, k\}$ and $s_j\in S_{r_j}$ if $\bar{\psi}_j(s_j)=b_j s_{j-1}$ for some $b_j\in A_j$ and $s_{j-1}\in S_{r_{j-1}}$ then $\psi_j(s_j)=b_j\varphi_{j-1}(s_{j-1})$. Applying the procedure P to the derivation D_1 and the decomposition $p_0=r_0[S_{r_0}, \varphi_0]$ we obtain the derivations $D_1^{r_0}, \bar{D}_1^{r_0}$.

Assume that the derivations $D_{j-1}^{r_0}, \bar{D}_{j-1}^{r_0}$ are constructed for an index $j\ (2\leqq j\leqq k)$. Then the derivations $D_j^{r_0}, \bar{D}_j^{r_0}$ are obtained by applying the procedure P to D_j and decomposition $p_{j-1}=r_{j-1}[S_{r_{j-1}}, \varphi_{j-1}]$, where the decomposition $r_{j-1}[S_{r_{j-1}}, \varphi_{j-1}]$ of p_{j-1} is given in the derivation $D_{j-1}^{r_0}$.

Let $\mathfrak{A}_i=(G_{i-1}, Y_{i-1}, A_i, G_i, Y_i, A_i', \Sigma_{\mathfrak{A}_i})$ $(i=1, \ldots, k)$ be R-transducers. Let us denote the arity function of the operator domain G_0 by v. We fix these notations

for this chapter. Let $D=D_1, \ldots, D_k$ be the following derivation sequence:

$$D_1\colon a_1 p_0 \Rightarrow^*_{\mathfrak{A}_1} p_1 \ (p_0\in T_{G_0}(Y_0),\ p_1\in T_{G_1}(Y_1),\ a_1\in A_0'),$$

$$D_2\colon a_2 p_1 \Rightarrow^*_{\mathfrak{A}_2} p_2 \ (p_2\in T_{G_2}(Y_2),\ a_2\in A_2'),$$

$$\vdots$$

$$D_k\colon a_k p_{k-1} \Rightarrow^*_{\mathfrak{A}_k} p_k \ (p_k\in T_{G_k}(Y_k),\ a_k\in A_k'),$$

moreover, we assume that $p_0=q_0[S_{q_0}, \gamma_0]$ holds for some

$$q_0\in P_{G_0}(Y_0),\ \gamma_0\colon S_{q_0}\to T_{G_0}(Y_0).$$

Applying the procedure S to the derivation sequence D and decomposition $p_0=q_0[S_{q_0}, \varphi_0]$ we obtain derivation sequences D^{q_0} and $\bar{D}^{q_0}$.

$$D_1^{q_0}\colon a_1 q_0[S_{q_0}, \gamma_0] \Rightarrow^*_{\mathfrak{A}_1} q_1[S_{q_1}, \alpha_1] \Rightarrow^*_{\mathfrak{A}_1} q_1[S_{q_1}, \gamma_1] = p_1,$$

$$(q_1\in P_{G_1}(Y_1),\ \alpha_1\colon S_{q_1}\to A_1 \operatorname{rg}(\gamma_0),\ \gamma_1\colon S_{q_1}\to T_{G_1}(Y_1)),$$

and for every $s_1\in S_{q_1}$, $\alpha_1(s_1)\Rightarrow^*_{\mathfrak{A}_1}\gamma_1(s_1)$ holds.

$$D_2^{q_0}\colon a_2 q_1[S_{q_1}, \gamma_1] \Rightarrow^*_{\mathfrak{A}_2} q_2[S_{q_2}, \alpha_2] \Rightarrow^*_{\mathfrak{A}_2} q_2[S_{q_2}, \gamma_2] = p_2,$$

$$(q_2\in P_{G_2}(Y_2),\ \alpha_2\colon S_{q_2}\to A_2 \operatorname{rg}(\gamma_1),\ \gamma_2\colon S_{q_2}\to T_{G_2}(Y_2)),$$

and for every $s_2\in S_{q_2}$, $\alpha_2(s_2)\Rightarrow^*_{\mathfrak{A}_2}\gamma_2(s_2)$ holds.

$$\vdots$$

$$D_k^{q_0}\colon a_k q_{k-1}[S_{q_{k-1}}, \gamma_{k-1}] \Rightarrow^*_{\mathfrak{A}_k} q_k[S_{q_k}, \alpha_k] \Rightarrow^*_{\mathfrak{A}_k} q_k[S_{q_k}, \gamma_k] = p_k,$$

$$(q_k\in P_{G_k}(Y_k),\ \alpha_k\colon S_{q_k}\to A_k \operatorname{rg}(\gamma_{k-1}),\ \gamma_k\colon S_{q_k}\to T_{G_k}(Y_k)),$$

and for every $s_k\in S_{q_k}$, $\alpha_k(s_k)\Rightarrow^*_{\mathfrak{A}_k}\gamma_k(s_k)$ holds.

$$\bar{D}_1^{q_0}\colon a_1 q_0 \Rightarrow^*_{\mathfrak{A}_1} q_1[S_{q_1}, \bar{\alpha}_1],\ (\bar{\alpha}_1\colon S_{q_1}\to A_1 S_{q_0}),$$

$$\bar{D}_2^{q_0}\colon a_2 q_1 \Rightarrow^*_{\mathfrak{A}_2} q_2[S_{q_2}, \bar{\alpha}_2],\ (\bar{\alpha}_2\colon S_{q_2}\to A_2 S_{q_1}),$$

$$\vdots$$

$$\bar{D}_k^{q_0}\colon a_k q_{k-1} \Rightarrow^*_{\mathfrak{A}_k} q_k[S_{q_k}, \bar{\alpha}_k],\ (\bar{\alpha}_k\colon S_{q_k}\to A_k S_{q_{k-1}}),$$

and for every $j\in\{1, \ldots, k\}$ and $s_j\in S_{q_j}$ if

$$\bar{\alpha}_j(s_j) = b_j s_{j-1} \quad \text{for some} \quad b_j\in A_j,\ s_{j-1}\in S_{q_{j-1}},$$

then

$$\alpha_j(s_j) = b_j \gamma_{j-1}(s_{j-1}).$$

We shall define a set $Z_{(D, q_0)}$ and mappings

$$\Omega_{(D, q_0)}\colon Z_{(D, q_0)}\to (A_1 \cup A_2 A_1 \cup \ldots \cup A_k \ldots A_1) \operatorname{rg}(\gamma_0),$$

$$\theta_{(D, q_0)}\colon S_{q_k}\to Z_{(D, q_0)}$$

and

$$\psi_{(D, q_0)}\colon S_{q_k}\to A_k \ldots A_1 T_{G_0}(Y_0)$$

in the following way:

$$Z_{(D,q_0)} = \{(s_0, s_1, \ldots, s_j) | s_0 \in S_{q_0}, s_1 \in S_{q_1}, \ldots, s_j \in S_{q_j},$$
$$1 \leqq j \leqq k \text{ and } (j=k \text{ or } (j<k \text{ and there are no } s_{j+1} \in S_{q_{j+1}} \text{ and } b_{j+1} \in A_{j+1} \text{ such that } \bar{\alpha}_{j+1}(s_{j+1}) = b_{j+1} s_j))$$
$$\text{and } \bar{\alpha}_i(s_i) = b_i s_{i-1} \ (b_i \in A_i) \text{ for } i = 1, \ldots, j\}.$$

For every $(s_0, s_1, \ldots, s_j) \in Z_{(D,q_0)}$

$$\Omega_{(D,q_0)}((s_0, s_1, \ldots, s_j)) = b_j \ldots b_1 \gamma_0(s_0)$$

iff

$$\bar{\alpha}_i(s_i) = b_i s_{i-1} \quad \text{for} \quad i = 1, \ldots, j.$$

For every $s_k \in S_{q_k}$, $\theta_{(D,q_0)}(s_k) = (s_0, s_1, \ldots, s_k)$ iff

$$\bar{\alpha}_i(s_i) = b_i s_{i-1} (b_i \in A_i) \quad \text{for} \quad i = 1, \ldots, k.$$

For each $s_k \in S_{q_k}$, $\psi_{(D,q_0)}(s_k) = b_k \ldots b_1 \gamma_0(s_0)$ iff

$$\theta_{(D,q_0)}(s_k) = (s_0, s_1, \ldots, s_k)$$

and

$$\Omega_{(D,q_0)}((s_0, s_1, \ldots, s_k)) = b_k \ldots b_1 \gamma_0(s_0).$$

One can see the equality $\psi_{(D,q_0)} = \theta_{(D,q_0)} \circ \Omega_{(D,q_0)}$ holds.

For the derivation sequence D and a decomposition

$$p_0 = q_0[S_{q_0}, \gamma_0] \ (q_0 \in P_{G_0}(Y_0), \gamma_0 \colon S_{q_0} \to T_{G_0}(Y_0))$$

we can determine the configuration

$$K_{(D,q_0)} \colon (q_k[S_{q_k}, \psi_{(D,q_0)}], \theta_{(D,q_0)}, Z_{(D,q_0)}, \Omega_{(D,q_0)}).$$

For the sake of a unified formalism, in the sequel we use the following convention. Let G be an operator domain with arity function $\bar{\nu}$, and let Y be a set disjoint with G. If $u \in G^0 \cup Y$ then $u(1, \ldots, \bar{\nu}(u))$ means the G-tree u over Y, moreover, $u(1, \ldots, \bar{\nu}(u))[\{1, \ldots, \bar{\nu}(u)\}, \vartheta]$ means u for arbitrary ϑ.

We continue the analysis of derivation sequence D. For each $s_0 \in S_{q_0}$ the tree $\gamma_0(s_0)$ can be written in the following form:

$$\gamma_0(s_0) = u_0(1, \ldots, \nu(u_0))[\{1, \ldots, \nu(u_0)\} \vartheta_0],$$

where $u_0 \in G_0 \cup Y_0$ and $\vartheta_0 \colon \{1, \ldots, \nu(u_0)\} \to T_{G_0}(Y_0)$. There are two cases.

1. *Case* $Z_{(D,q_0)} = \emptyset$. Take the quasi tree $r_0 \in P_{G_0}(Y_0)$ defined by $r_0 = q_0[S_{q_0}, \xi_0]$, where the mapping $\xi_0 \colon S_{q_0} \to T_{G_0}(Y_0 \cup N^*)$ is determined by the following formula: for each

$$s_0 \in S_{q_0} \ \xi_0(s_0) = \omega_{s_0}(u_0(1, \ldots, \nu(u_0))) \quad \text{if} \quad \gamma_0(s_0) = u_0(1, \ldots, \nu(u_0))[\{1, \ldots, \nu(u_0)\}, \vartheta_0]$$
$$(u_0 \in G_0 \cup Y_0, \vartheta_0 \colon \{1, \ldots, \nu(u_0)\} \to T_{G_0}(Y_0)).$$

One can see $K_{(D,q_0)} = K_{(D,r_0)}$ holds.

2. *Case* $Z_{(D,q_0)} \neq \emptyset$. Using these decompositions of the trees $\gamma_0(s_0)$ we obtain the derivation sequence $E = E_1, \ldots, E_k$ from D. For every $i \in \{1, \ldots, k\}$ the derivation E_i

is the same as D_i disregarding the order of direct derivations in D_i. We shall introduce the derivation sequence $\bar{E}=\bar{E}_1, \ldots, \bar{E}_k$ too.

$$E_1\colon a_1 q_0[S_{q_0}, \gamma_0] \Rightarrow^*_{\mathfrak{A}_1} q_1[S_{q_1}, \alpha_1] \Rightarrow^*_{\mathfrak{A}_1} q_1[S_{q_1}, \beta_1] \Rightarrow^*_{\mathfrak{A}_1}$$

$$\Rightarrow^*_{\mathfrak{A}_1} q_1[S_{q_1}, \gamma_1] \ (q_1 \in P_{G_1}(Y_1),\ \alpha_1\colon S_{q_1} \to A_1 \operatorname{rg}(\gamma_0),$$

$$\beta_1\colon S_{q_1} \to T_{G_1}(Y_1 \cup A_1 T_{G_0}(Y_0)),\ \gamma_1\colon S_{q_1} \to T_{G_1}(Y_1)).$$

$$\bar{E}_1\colon a_1 q_0[S_{q_0}, \xi_0] \Rightarrow^*_{\mathfrak{A}_1} q_1[S_{q_1}, \bar{\beta}_1]$$

$$(\xi_0\colon S_{q_0} \to T_{G_0}(Y_0 \cup N^*),\ \bar{\beta}_1\colon S_{q_1} \to T_{G_1}(Y_1 \cup A_1 N^*)).$$

ξ_0 is defined by the following formula: for each

$$s_0 \in S_{q_0} \quad \text{if} \quad \gamma_0(s_0) = u_0(1, \ldots, \nu(u_0))[\{1, \ldots, \nu(u_0)\}, \vartheta_0]$$

for some $u_0 \in G_0 \cup Y_0$ and mapping

$$\vartheta_0\colon \{1, \ldots, \nu(u_0)\} \to T_{G_0}(Y_0) \quad \text{then} \quad \xi_0(s_0) = \omega_{s_0}(u_0(1, \ldots, \nu(u_0))).$$

We shall define the mappings β_1 and $\bar{\beta}_1$. For every $s_1 \in S_{q_1}$ let us consider the subderivation

(1) $\alpha_1(s_1) \Rightarrow^*_{\mathfrak{A}_1} \gamma_1(s_1)$ of D.

Let us assume that $\alpha_1(s_1) = b_1 s_0$ and

$$\alpha_1(s_1) = b_1 \gamma_0(s_0) = b_1 u_0(1, \ldots, \nu(u_0))[\{1, \ldots, \nu(u_0)\}, \vartheta_0],$$

where

$$s_0 \in S_{q_0},\ b_1 \in A_1,\ u_0 \in G_0 \cup Y_0,\ \vartheta_0\colon \{1, \ldots, \nu(u_0)\} \to T_{G_0}(Y_0).$$

Applying procedure P to derivation (1) and decomposition

$$\gamma_0(s_0) = u_0(1, \ldots, \nu(u_0))[\{1, \ldots, \nu(u_0)\}, \vartheta_0]$$

we obtain derivations (2), (3).

(2) $b_1 u_0(1, \ldots, \nu(u_0))[\{1, \ldots, \nu(u_0)\}, \vartheta_0] \Rightarrow_{\mathfrak{A}_1} u_1[S_{u_1}, \delta_1] \Rightarrow^*_{\mathfrak{A}_1}$

$\Rightarrow^*_{\mathfrak{A}_1} u_1[S_{u_1}, \vartheta_1] = \gamma_1(s_1)$, where $u_1 \in P_{G_1}(Y_1)$,

$\delta_1\colon S_{u_1} \to A_1 T_{G_0}(Y_0),\ \vartheta_1\colon S_{u_1} \to T_{G_1}(Y_1)$,

and for each

$v_1 \in S_{u_1},\ \delta_1(v_1) \Rightarrow^*_{\mathfrak{A}_1} \vartheta_1(v_1)$ holds.

(3) $b_1 u_0(1, \ldots, \nu(u_0)) \Rightarrow_{\mathfrak{A}_1} u_1[S_{u_1}, \bar{\delta}_1]$, where

$\bar{\delta}_1\colon S_{u_1} \to A_1\{1, \ldots, \nu(u_0)\}$ and for each $v_1 \in S_{u_1}$ if

$\bar{\delta}_1(v_1) = c_1 t_0$ $(c_1 \in A_1,\ t_0 \in \{1, \ldots, \nu(u_0)\})$ then $\delta_1(v_1) = c_1 \vartheta_0(t_0)$.

In this case β_1 and $\bar{\beta}_1$ are defined by

$$\beta_1(s_1) = u_1[S_{u_1}, \delta_1],\ \bar{\beta}_1(s_1) = \omega_{s_0}(u_1[S_{u_1}, \bar{\delta}_1]).$$

Let l be an index of the transducers in consideration such that $2 \leqq l \leqq k$. The derivations E_l and $\bar{E}_l$ are the following:

$$E_l\colon a_l q_{l-1}[S_{q_{l-1}}, \gamma_{l-1}] \Rightarrow^*_{\mathfrak{A}_l} q_l[S_{q_l}, \alpha_l] \Rightarrow^*_{\mathfrak{A}_l} q_l[S_{q_l}, \beta_l] \Rightarrow^*_{\mathfrak{A}_l} q_l[S_{q_l}, \gamma_l]$$

$$(q_l \in P_{G_l}(Y_l),\ \alpha_l\colon S_{q_l} \to A_l \operatorname{rg}(\gamma_{l-1}),$$

$$\beta_l\colon S_{q_l} \to T_{G_l}(Y_l \cup A_l T_{G_{l-1}}(Y_{l-1})),\ \gamma_l\colon S_{q_l} \to T_{G_l}(Y_l)).$$

$$\bar{E}_l\colon a_l q_{l-1}[S_{q_{l-1}}, \xi_{l-1}] \Rightarrow^*_{\mathfrak{A}_l} q_l[S_q, \bar{\beta}_l]$$

$$(\xi_{l-1}\colon S_{q_{l-1}} \to T_{G_{l-1}}(Y_{l-1} \cup N^*),$$

$$\bar{\beta}_l\colon S_{q_l} \to T_{G_l}(Y_l \cup A_l N^*)).$$

ξ_{l-1} is defined by the following formula: for every $s_{l-1} \in S_{q_{l-1}}$ if $\beta_{l-1}(s_{l-1}) = u_{l-1}[S_{u_{l-1}}, \delta_{l-1}]$ then $\xi_{l-1}(s_{l-1}) = \omega_{s_{l-1}}(u_{l-1})$. We shall define the mappings β_l and $\bar{\beta}_l$. For every $s_l \in S_{q_l}$ let us consider the subderivation

(1) $\alpha_l(s_l) \Rightarrow^*_{\mathfrak{A}_l} \gamma_l(s_l)$ of D_l.

Let us assume that

$$\bar{\alpha}_l(s_l) = b_l s_{l-1} \quad \text{and} \quad \alpha_l(s_l) = b_l \gamma_{l-1}(s_{l-1}) = b_l u_{l-1}[S_{u_{l-1}}, \vartheta_{l-1}],$$

where

$$s_{l-1} \in S_{q_{l-1}},\ b_l \in A_l,\ u_{l-1} \in P_{G_{l-1}}(Y_{l-1}),$$

and the decomposition $\gamma_{l-1}(s_{l-1}) = u_{l-1}[S_{u_{l-1}}, \vartheta_{l-1}]$ of $\gamma_{l-1}(s_{l-1})$ is the same as in E_{l-1}. Applying the procedure P to derivation (1) and decomposition $\gamma_{l-1}(s_{l-1}) = u_{l-1}[S_{u_{l-1}}, \vartheta_{l-1}]$ we obtain derivations (2), (3).

(2) $b_l u_{l-1}[S_{u_{l-1}}, \vartheta_{l-1}] \Rightarrow^*_{\mathfrak{A}_l} u_l[S_{u_l}, \delta_l] \Rightarrow^*_{\mathfrak{A}_l} u_l[S_{u_l}, \vartheta_l] = \gamma_l(s_l)$,

where

$$u_l \in P_{G_l}(Y_l),\ \delta_l\colon S_{u_l} \to A_l T_{G_{l-1}}(Y_{l-1}),\ \vartheta_l\colon S_{u_l} \to T_{G_l}(Y_l),$$

and for every $v_l \in S_{u_l}$ the derivation $\delta_l(v_l) \Rightarrow^*_{\mathfrak{A}_l} \vartheta_l(v_l)$ is valid.

(3) $b_l u_{l-1} \Rightarrow^*_{\mathfrak{A}_l} u_l[S_{u_l}, \bar{\delta}_l]$, where $\bar{\delta}_l\colon S_{u_l} \to A_l S_{u_{l-1}}$

and for each $v_l \in S_{u_l}$ if

$$\bar{\delta}_l(v_l) = c_l t_{l-1} (c_l \in A_l,\ t_{l-1} \in S_{u_{l-1}})$$

then

$$\delta_l(v_l) = c_l \vartheta_{l-1}(t_{l-1}).$$

In this case β_l and $\bar{\beta}_l$ are defined by

$$\beta_l(s_l) = u_l[S_{u_l}, \delta_l],\ \bar{\beta}_l(s_l) = \omega_{s_{l-1}}(u_l[S_u, \bar{\delta}_l]).$$

Take the quasi-tree $r_0 \in P_{G_0}(Y_0)$ defined by $r_0 = q_0[S_{q_0}, \xi_0]$, where the mapping $\xi_0 \colon S_{q_0} \to T_{G_0}(Y_0 \cup N^*)$ as in $\bar{E}_1$. Let $\lambda_0 \colon S_{r_0} \to T_{G_0}(Y_0)$ be the mapping such that

$$\lambda_0(s_0 i) = \vartheta_0(i) \quad \text{if} \quad \gamma_0(s_0) = u_0(1, \ldots, v(u_0))[\{1, \ldots, v(u_0)\}, \vartheta_0],$$

where

$$s_0 \in S_{q_0},\ u_0 \in G_0 \cup Y_0,\ \vartheta_0 \colon \{1, \ldots, v(u_0)\} \to T_{G_0}(Y_0), \quad i \in \{1, \ldots, v(u_0)\}.$$

For these r_0 and λ_0 we have that $q_0[S_{q_0}, \gamma_0] = r_0[S_{r_0}, \lambda_0]$ holds. We take the quasi tree $r_1 \in P_{G_1}(Y_1)$ which is defined by $r_1 = q_1[S_{q_1}, \xi_1]$, $\xi_1 \colon S_{q_1} \to T_{G_1}(Y_1 \cup N^*)$, for every $s_1 \in S_{q_1}$, $\xi_1(s_1) = \omega_{s_1}(u_1)$ if $\beta_1(s_1) = u_1[S_{u_1}, \delta_1]$. It can be seen that

$$S_{r_1} = \{s_1 t_1 | s_1 \in S_{q_1},\ \xi_1(s_1) = \omega_{s_1}(u_1),\ t_1 \in S_{u_1}\}$$

holds. Let us define the mappings $\eta_1 \colon S_{r_1} \to A_1 T_{G_0}(Y_0)$, $\bar{\eta}_1 \colon S_{r_1} \to A_1 S_{r_0}$ and $\lambda_1 \colon S_{r_1} \to T_{G_1}(Y_1)$ as follows: for each $\bar{s}_1 \in S_{r_1}$ let us consider its unique decomposition $\bar{s}_1 = s_1 t_1$, where $s_1 \in S_{q_1}$, $\bar{\alpha}_1(s_1) = b_1 s_0$ for some $b_1 \in A_1$ and

$$s_0 \in S_{q_0},\ \xi_1(s_1) = \omega_{s_1}(u_1),\ t_1 \in S_{u_1},\ \beta_1(s_1) = u_1[S_{u_1}, \delta_1],$$

$$\bar{\beta}_1(s_1) = \omega_{s_0}(u_1[S_{u_1}, \bar{\delta}_1]),\ \gamma_1(s_1) = u_1[S_{u_1}, \vartheta_1]$$

and $\alpha_1, \beta_1, \bar{\beta}_1, \gamma_1, \delta_1, \bar{\delta}_1, \vartheta_1$ as in $E_1, \bar{E}_1$.

Let $\eta_1(s_1 t_1) = \delta_1(t_1)$, $\bar{\eta}_1(s_1 t_1) = \omega_{s_0}(\bar{\delta}_1(t_1))$, $\lambda_1(s_1 t_1) = \vartheta_1(t_1)$. The derivation $\delta_1(t_1) \Rightarrow^*_{\mathfrak{gl}_1} \vartheta_1(t_1)$ holds, which implies that the derivation $\eta_1(s_1 t_1) \Rightarrow^*_{\mathfrak{gl}_1} \lambda_1(s_1 t_1)$ is valid. Thus we obtain the derivations E_1' and $\bar{E}_1'$ from E_1 and $\bar{E}_1$, respectively.

$$E_1' \colon a_1 r_0[S_{r_0}, \lambda_0] \Rightarrow^*_{\mathfrak{gl}_1} r_1[S_{r_1}, \eta_1] \Rightarrow^*_{\mathfrak{gl}_1} r_1[S_{r_1}, \lambda_1],$$

$$\bar{E}_1' \colon a_1 r_0 \Rightarrow^*_{\mathfrak{gl}_1} r_1[S_{r_1}, \bar{\eta}_1],$$

and for each $v_1 \in S_{r_1}$ if $\bar{\eta}_1(v_1) = c_1 v_0$ for some $c_1 \in A_1$, $v_0 \in S_{r_0}$, then $\eta_1(v_1) = c_1 \lambda_0(v_0)$. For each $2 \leqq l \leqq k$ we take the quasi trees $r_l \in P_{G_l}(Y_l)$ which is defined by

$$r_l = q_l[S_{q_l}, \xi_l],\ \xi_l \colon S_{q_l} \to T_{G_l}(Y_l \cup N^*),$$

for every

$$s_l \in S_{q_l},\ \xi_l(s_l) = \omega_{s_l}(u_l) \quad \text{if} \quad \beta_l(s_l) = u_l[S_{u_l}, \delta_l].$$

It can be seen that

$$S_{r_l} = \{s_l t_l | \xi_l(s_l) = \omega_{s_l}(u_l),\ t_l \in S_{u_l}\}$$

holds. Let us define the mappings $\eta_l \colon S_{r_l} \to A_l T_{G_{l-1}}(Y_{l-1})$, $\bar{\eta}_l \colon S_{r_l} \to A_l S_{r_{l-1}}$ and $\lambda_l \colon S_{r_l} \to T_{G_l}(Y_l)$ as follows: for each $\bar{s}_l \in S_{r_l}$ let us consider its unique decomposition

$$\bar{s}_l = s_l t_l, \quad \text{where} \quad s_l \in S_{q_l},\ \xi_l(s_l) = \omega_{s_l}(u_l),\ t_l \in S_{u_l},$$

$$\beta_l(s_l) = u_l[S_{u_l}, \delta_l],\ \bar{\beta}_l(s_l) = u_l[S_{u_l}, \bar{\delta}_l],\ \gamma_l(s_l) = u_l[S_{u_l}, \vartheta_l],$$

$(\bar{\alpha}_l, \beta_l, \bar{\beta}_l, \gamma_l, \delta_l, \bar{\delta}_l, \vartheta_l$ as in $E_l, \bar{E}_l)$ and $\bar{\alpha}_l(s_l) = b_l s_{l-1}$ for some $b_l \in A_l$ and $s_{l-1} \in S_{q_{l-1}}$. In this case $\eta_l, \bar{\eta}_l$ and λ_l are defined by $\eta_l(s_l t_l) = \delta_l(t_l)$, $\bar{\eta}_l(s_l t_l) = \omega_{s_{l-1}}(\bar{\delta}_l(t_l))$, $\lambda_l(s_l t_l) = \vartheta_l(t_l)$. The derivation $\delta_l(t_l) \Rightarrow^*_{\mathfrak{gl}_l} \vartheta_l(t_l)$ holds, which implies that the derivation $\eta_l(s_l t_l) \Rightarrow^*_{\mathfrak{gl}_l} \lambda_l(s_l t_l)$ is valid. Thus we obtain the derivations

E_l' and $\bar{E}_l'$ from E_l and $\bar{E}_l$, respectively.

$$E_l'\colon a_l r_{l-1}[S_{r_{l-1}}, \lambda_{l-1}] \Rightarrow^*_{\mathfrak{A}_l} r_l[S_{r_l}, \eta_l] \Rightarrow^*_{\mathfrak{A}_l} r_l[S_{r_l}, \lambda_l],$$

$$\bar{E}_l'\colon a_l r_{l-1} \Rightarrow^*_{\mathfrak{A}_l} r_l[S_{r_l}, \bar{\eta}_l],$$

and for each $v_l \in S_{r_l}$ if $\bar{\eta}_l(v_l) = c_l v_{l-1}$ for some $c_l \in A_l$, $v_{l-1} \in S_{r_{l-1}}$, then $\eta_l(s_l) = c_l \lambda_{l-1}(v_{l-1})$.

For the sequence of root-to-frontier tree transducers $\mathfrak{A}_1, \ldots, \mathfrak{A}_k$ we shall define the sets $\Sigma(l)$ and V_l, $(0 \leqq l \leqq k)$ in the following way:

$\Sigma(0) = \{u_0 | u_0 \in G_0 \cup Y_0\}$,

$V_0 = \Sigma(0)$;

$\Sigma(1) = \{(b_1, u_0, u_1[S_{u_1}, \varphi_1], \varrho_1, W_1, \tau_1) | b_1 u_0 \to u_1[S_{u_1}, \varphi_1] \in \Sigma_{\mathfrak{A}_1},$

$u_1 \in P_{G_1}(Y_1)$, $\varphi_1\colon S_{u_1} \to A_1 S_{u_0}$,

$W_1 = \{(t_0, t_1) | \varphi_1(t_1) = c_1 t_0, c_1 \in A_1\}$,

$\varrho_1\colon S_{u_1} \to W_1$; $\varrho_1(t_1) = (t_0, t_1)$ if $\varphi_1(t_1) = c_1 t_0$,

$\tau_1\colon W_1 \to A_1 S_{u_0}$; $\tau_1((t_0, t_1)) = c_1 t_0$ if $\varphi_1(t_1) = c_1 t_0\}$.

It can be seen that for each $t_1 \in S_{u_1}$, $\varphi_1(t_1) = \tau_1(\varrho_1(t_1))$, that is, $\varphi_1 = \varrho_1 \circ \tau_1$ holds We say that the element $(b_1, u_0, u_1[S_{u_1}, \varphi_1], \varrho_1, W_1, \tau_1)$ of $\Sigma(1)$ is generated by the production $b_1 u_0 \to u_1[S_{u_1}, \varphi_1]$.

$V_1 = \{(u_0, \sigma_1) | u_0 \in \Sigma(0), \sigma_1 \in \Sigma(1)$ and the second component of σ_1 is $u_0\}$.

Let j be an index such that $2 \leqq j \leqq k$, and assume that for each i $(1 \leqq i < j)$ the sets $\Sigma(i)$ and V_i are defined, and that for each $\sigma_i = (b_i \ldots b_1, u_0, u_i[S_{u_i}, \varphi_i], \varrho_i, W_i, \tau_i) (\in \Sigma_{\mathfrak{A}}(i))$ $\varphi_i = \varrho_i \circ \tau_i$ holds. We shall define $\Sigma(j)$ and V_j as follows:

$\Sigma(j) = \{(b_j \ldots b_1, u_0, u_j[S_{u_j}, \varphi_j], \varrho_j, W_j, \tau_j) |$

$(b_{j-1} \ldots b_1, u_0, u_{j-1}[S_{u_{j-1}}, \varphi_{j-1}], \varrho_{j-1}, W_{j-1}, \tau_{j-1}) \in \Sigma(j-1)$,

$b_j u_{j-1} \Rightarrow^*_{\mathfrak{A}_j} u_j[S_{u_j}, \varepsilon_j]$ holds, where $u_j \in P_{G_j}(Y_j)$,

$\varepsilon_j\colon S_{u_j} \to A_j S_{u_{j-1}}$,

$\varphi_j\colon S_{u_j} \to A_j A_{j-1} \ldots A_1 \{1, \ldots, v(u_0)\}$;

$\varphi_j(t_j) = c_j c_{j-1} \ldots c_1 t_0$ if $\varepsilon_j(t_j) = c_j t_{j-1}$ and

$\varphi_{j-1}(t_{j-1} = c_{j-1} \ldots c_1 t_0$,

$W_j = \{(t_0, \ldots, t_{j-1}, t_j) | \varepsilon_j(t_j) = c_j t_{j-1}, c_j \in A_j, \quad \varrho_{j-1}(t_{j-1}) = (t_0, \ldots, t_{j-1})\} \cup$

$\cup \{(t_0, \ldots, t_{j-1}) \in W_{j-1} |$ there are no t_j in S_{u_j} and $c_j \in A_j$ such that

$\varepsilon_j(t_j) = c_j t_{j-1}\} \cup$

$\cup \{(t_0, \ldots, t_l) \in W_{j-1} | 1 \leqq l \leqq j-2\}$,

$\varrho_j\colon S_{u_j} \to W_j$; $\varrho_j(t_j) = (t_0, \ldots, t_{j-1}, t_j)$ if

$\varepsilon_j(t_j) = c_j t_{j-1}$ and $\varrho_{j-1}(t_{j-1}) = (t_0, \ldots, t_{j-1})$,

τ_j: $W_j \to A_j \dots A_1\{1, \dots, v(u_0)\}$;

$\tau_j|_{W_j \cap W_{j-1}} = \tau_{j-1}|_{W_j \cap W_{j-1}}$ and

if $(t_0, \dots, t_{j-1}, t_j) \in W_j$ and $\varepsilon_j(t_j) = c_j t_{j-1}$

then $\tau_j((t_0, \dots, t_{j-1}, t_j)) = c_j \tau_{j-1}((t_0, \dots, t_{j-1}))\}$.

We say that the element $(b_j \dots b_1, u_0, u_j[S_{u_j}, \varphi_j], \varrho_j, W_j, \tau_j)$ of $\Sigma(j)$ is generated by the derivation $b_j u_{j-1} \Rightarrow^*_{\mathfrak{A}_j} u_j[S_{u_j}, \varepsilon_j]$ and element

$$(b_{j-1} \dots b_1, u_0, u_{j-1}[S_{u_{j-1}}, \varphi_{j-1}], \varrho_{j-1}, W_{j-1}, \tau_{j-1}) \quad \text{of} \quad \Sigma(j-1).$$

It can be seen that for each element $(b_j \dots b_1, u_0, u_j[S_{u_j}, \varphi_j], \varrho_j, W_j, \tau_j)$ of $\Sigma(j)$, $\varphi_j = \varrho_j \circ \tau_j$ hold.

$V_j = \{(u_0, \sigma_1, \dots, \sigma_{j-1}, \sigma_j) | (u_0, \sigma_1, \dots, \sigma_{j-1}) \in V_{j-1}, \sigma_{j-1}$

has the form $(b_{j-1} \dots b_1, u_0, u_{j-1}[S_{u_{j-1}}, \varphi_{j-1}], \varrho_{j-1}, W_{j-1}, \tau_{j-1})$, σ_j has the form $(b_j \dots b_1, u_0, u_j[S_{u_j}, \varphi_j], \varrho_j, W_j, \tau_j)$ and σ_j is generated by the derivation $b_j u_{j-1} \Rightarrow^*_{\mathfrak{A}_j} u_j[S_{u_j}, \varepsilon_j]$ and $\sigma_{j-1}\}$.

We define mappings $\varkappa_i$: $[Z_{(D, q_0)}]_i \to \Sigma(i)$ for $0 \leqq i \leqq k$. Let $s_0 \in [Z_{(D, q_0)}]_0$, which means $s_0 \in S_{q_0}$. $\varkappa_0(s_0)$ is defined by

$$\varkappa_0(s_0) = \text{root}\,(\gamma_0(s_0)) = u_0 \quad \text{if}$$

$$\gamma_0(s_0) = u_0(1, \dots, v(u_0))[\{1, \dots, v(u_0)\}, \vartheta_0](\vartheta_0\colon \{1, \dots, v(u_0)\} \to T_{G_0}(Y_0)).$$

Let $s_1 \in [Z_{(D, q_1)}]_1$, that is, $s_1 \in S_{q_1}$. Let us consider the decomposition $\alpha_1(s_1) = b_1 u_0(1, \dots, v(u_0))[\{1, \dots, v(u_0)\}, \vartheta_0]$ $(u_0 \in G_0 \cup Y_0$, $\vartheta_0\colon \{1, \dots, v(u_0)\} \to T_{G_0}(Y_0))$.

Applying the procedure P to the subderivation (1) $\alpha_1(s_1) \Rightarrow^*_{\mathfrak{A}_1} \gamma_1(s_1)$ of D_1 and decomposition $u_0(1, \dots, v(u_0))[\{1, \dots, v(u_0)\}, \vartheta_0]$ we obtain derivations (2) and (3).

(2) $b_1 u_0(1, \dots, v(u_0))[\{1, \dots, v(u_0)\}, \vartheta_0] \Rightarrow_{\mathfrak{A}_1} u_1[S_{u_1}, \delta_1] \Rightarrow^*_{\mathfrak{A}_1}$

$\Rightarrow^*_{\mathfrak{A}_1} u_1[S_{u_1}, \vartheta_1] = \gamma_1(s_1)$, where $u_1 \in P_{G_1}(Y_1)$,

δ_1: $S_{u_1} \to A_1 T_{G_0}(Y_0)$, ϑ_1: $S_{u_1} \to T_{G_1}(Y_1)$,

and for each

$v_1 \in S_{u_1}$, $\delta_1(v_1) \Rightarrow^*_{\mathfrak{A}_1} \vartheta_1(v_1)$ holds.

(3) $b_1 u_0 \Rightarrow_{\mathfrak{A}_1} u_1[S_{u_1}, \bar{\delta}_1]$, where $\bar{\delta}_1$: $S_{u_1} \to A_1\{1, \dots, v(u_0)\}$ and for each $v_1 \in S_{u_1}$ if $\bar{\delta}_1(v_1) = c_1 t_0$ $(c_1 \in A_1$, $t_0 \in \{1, \dots, v(u_0)\})$, then $\delta_1(v_1) = c_1 \vartheta_0(t_0)$.

$\beta_1(s_1) = u_1[S_{u_1}, \delta_1]$, $\bar{\beta}_1(s_1) = \omega_{s_0}(u_1[S_{u_1}, \bar{\delta}_1])$

for β_1, $\bar{\beta}_1$ given in derivations E_1, $\bar{E}_1$.

Let $\varkappa_1(s_1)$ be the element of $\Sigma(1)$ generated by the production $b_1 u_0 \to u_1[S_{u_1}, \bar{\delta}_1]$.

Assume that $\varkappa_i$ is defined for every $0 \leqq i \leqq j-1$. Then the mapping $\varkappa_j$ $(2 \leqq j \leqq k)$ is defined in the following way: for each $s_j (\in [Z_{(D, q_0)}]_j = S_{q_j})$, $\bar{\alpha}_j(s_j) = b_j s_{j-1}$ for some $b_j \in A_j$ and $s_{j-1} \in [Z_{(D, q_0)}]_{j-1}$. Thus $\alpha_j(s_j) = b_j \gamma_{j-1}(s_{j-1})$. $\varkappa_{j-1}(s_{j-1})$ has the form $(b_{j-1} \dots b_1, u_0, u_{j-1}[S_{u_{j-1}}, \varphi_{j-1}], \varrho_{j-1}, W_{j-1}, \tau_{j-1}) \in \Sigma_{\mathfrak{B}}(j-1)$. Let us consider the decomposition $\gamma_{j-1}(s_{j-1}) = u_{j-1}[S_{u_{j-1}}, \vartheta_{j-1}]$ of $\gamma_{j-1}(s_{j-1})$ which is the same as in E_{j-1}.

Applying the procedure P to the subderivation
(1) $\alpha_j(s_j) \Rightarrow^*_{\mathfrak{A}_j} \gamma_j(s_j)$ of D_j and decomposition $u_{j-1}[S_{u_{j-1}}, \vartheta_{j-1}]$ we obtain derivations (2) and (3).

(2) $b_j u_{j-1}[S_{u_{j-1}}, \vartheta_{j-1}] \Rightarrow^*_{\mathfrak{A}_j} u_j[S_{u_j}, \delta_j] \Rightarrow^*_{\mathfrak{A}_j} u_j[S_{u_j}, \vartheta_j] = \gamma_j(s_j)$,

where

$u_j \in P_{G_j}(Y_j)$, $\delta_j\colon S_{u_j} \to A_j T_{G_{j-1}}(Y_{j-1})$, $\vartheta_j\colon S_{u_j} \to T_{G_j}(Y_j)$,

and for every $v_j \in S_{u_j}$ the derivation $\delta_j(v_j) \Rightarrow^*_{\mathfrak{A}_j} \vartheta_j(v_j)$ is valid.

(3) $b_j u_{j-1} \Rightarrow^*_{\mathfrak{A}_j} u_j[S_{u_j}, \delta_j]$, where $\delta_j\colon S_{u_j} \to A_j S_{u_{j-1}}$ and for each $v_j \in S_{u_j}$ if $\delta_j(v_j) = c_j t_{j-1}$ $(c_j \in A_j, t_{j-1} \in S_{u_{j-1}})$ then $\delta_j(v_j) = c_j \vartheta_{j-1}(t_{j-1})$.

Let $\varkappa_j(s_j)$ be the element of $\Sigma(j)$ generated by derivation (3) and $\varkappa_{j-1}(s_{j-1})$ $(\in \Sigma(j-1))$.

We associate the configuration

$$K_{(D,r_0)} = (r_k[S_{r_k}, \psi_{(D,r_0)}], \Theta_{(D,r_0)}, Z_{(D,r_0)}, \Omega_{(D,r_0)})$$

with the derivation sequence D and decomposition $p_0 = r_0[S_{r_0}, \lambda_0]$.

Using the derivation sequences E'_l and $\bar{E}'_l$ we shall show the connection between the configurations $K_{(D,q_0)}$ and $K_{(D,r_0)}$.

(1) $r_k = q_k[S_{q_k}, \xi_k]$, which was established in E'_k, moreover we know that for each $s_k \in S_{q_k}$, $\xi_k(s_k) = \omega_{s_k}(u_k)$, where

$$\varkappa_k(s_k) = (b_k \dots b_1, u_0, u_k[S_{u_k}, \varphi_k], \varrho_k, W_k, \tau_k).$$

(2) $Z_{(D,r_0)} = \{(s_0 t_0, s_1 t_1, \dots, s_j t_j) | (s_0, s_1, \dots, s_l) \in Z_{(D,q_0)}$

for some l $(1 \leqq j \leqq l \leqq k)$ and

$$\varkappa_l(s_l) = (b_l \dots b_1, u_0, u_l[S_{u_l}, \varphi_l], \varrho_l, W_l, \tau_l) \text{ and } (t_0, t_1, \dots, t_j) \in W_l\}.$$

(3) For every $\bar{s}_k \in S_{r_k}$ let us consider its unique decomposition $\bar{s}_k = s_k t_k$, where $s_k \in S_{q_k}$, $\varkappa_k(s_k)$ has the form

$$\varkappa_k(s_k) = (b_k \dots b_1, u_0, u_k[S_{u_k}, \varphi_k], \varrho_k, W_k, \tau_k),$$

$$\xi_k(s_k) = \omega_{s_k}(u_k) \text{ and } t_k \in S_{u_k}.$$

If $\Theta_{(D,q_0)}(s_k) = (s_0, s_1, \dots, s_k)$ and $\varrho_k(t_k) = (t_0, t_1, \dots, t_k)$ then

$$\Theta_{(D,r_0)}(\bar{s}_k) = (s_0 t_0, s_1 t_1, \dots, s_k t_k).$$

(4) Let $\bar{s}_k \in S_{r_k}$ be arbitrary, and consider its unique decomposition $\bar{s}_k = s_k t_k$, where $s_k \in S_{q_k}$, $\varkappa_k(s_k)$ has the form

$$\varkappa_k(s_k) = (b_k \dots b_1, u_0, u_k[S_{u_k}, \varphi_k], \varrho_k, W_k, \tau_k), \xi_k(s_k) = \omega_{s_k}(u_k)$$

and $t_k \in S_{u_k}$. Then if $\varphi_k(t_k = c_k \dots c_1 t_0$ and

$$\psi_{(D,q_0)}(s_k) = u_0(1, \dots, \nu(u_0))[\{1, \dots, \nu(u_0)\}, \vartheta_0]$$

$$(u_0 \in G_0 \cup Y_0, \vartheta_0\colon \{1, \dots, \nu(u_0)\} \to T_{G_0}(Y_0)), \text{ then}$$

$$\psi_{(D,r_0)}(s_k t_k) = c_k \dots c_1 \vartheta_0(t_0).$$

(5) From the definition of $Z_{(D,r_0)}$ it follows that for every $(\bar{s}_0, \bar{s}_1, \ldots, \bar{s}_j) \in Z_{(D,r_0)}$ there is a vector $(s_0, s_1, \ldots, s_l) \in Z_{(D,q_0)}$ for some $l(\geqq j)$ such that

$$\varkappa_l(s_l) = (b_l \ldots b_1, u_0, u_l[S_{u_l}, \varphi_l], \varrho_l, W_l, \tau_l),$$

and

$$\bar{s}_0 = s_0 t_0, \ \bar{s}_1 = s_1 t_1, \ldots, \bar{s}_j = s_j t_j$$

hold for some $(t_0, t_1, \ldots, t_j) \in W_l$.

If $\tau_l((t_0, t_1, \ldots, t_j)) = c_j \ldots c_1 t_0$ and

$$\Omega_{(D,q_0)}((s_0, s_1, \ldots, s_l)) = b_l \ldots b_1 u_0(1, \ldots, \nu(u_0))[\{1, \ldots, \nu(u_0)\}, \vartheta_0]$$

for some $u_0 \in G_0 \cup Y_0$ and ϑ_0: $\{1, \ldots, \nu(u_0)\} \to T_{G_0}(Y_0)$, then

$$\Omega_{(D,r_0)}((\bar{s}_0, \bar{s}_1, \ldots, \bar{s}_j)) = c_j \ldots c_1 \vartheta_0(t_0).$$

3. k-synchronized R-transducers

In this chapter we shall introduce the notion of a k-synchronized R-transducer and prove that the relations induced by this type of transducers are exactly those relations which can be obtained by compositions of k relations induced by root-to-frontier tree transducers.

Definition 3.1. A k-synchronized R-transducer is a system

$$\mathfrak{B} = (G_0, G_1, \ldots, G_k, Y_0, Y_1, \ldots, Y_k, A_1, \ldots, A_k, A_1', \ldots, A_k', \Sigma_{\mathfrak{B}}, V),$$

where

(1) $k \geqq 2$,
(2) $G_0, G_1, \ldots, G_k$ are operator domains,
(3) $A_1, \ldots, A_k$ are state sets, for $i = 1, \ldots, k$,

$$A_i \ldots A_1 \cap T_{G_0}(Y_0) = \emptyset, \quad \text{and} \quad A_k \ldots A_1 \cap T_{G_k}(Y_k) = \emptyset.$$

(4) $A_1' \subseteqq A_1, \ldots, A_k' \subseteqq A_k$ are the sets of initial states,
(5) $\Sigma_{\mathfrak{B}}$ is a finite set of productions, which is a disjoint union

$$\Sigma_{\mathfrak{B}} = \Sigma_{\mathfrak{B}}(0) \cup \Sigma_{\mathfrak{B}}(1) \cup \ldots \cup \Sigma_{\mathfrak{B}}(k),$$

$V = V_0 \cup V_1 \cup \ldots \cup V_k$, where $V_0 = \Sigma_{\mathfrak{B}}(0)$, and for $i = 1, \ldots, k$, $V_i \subseteqq \Sigma_{\mathfrak{B}}(0) \times \Sigma_{\mathfrak{B}}(1) \times \ldots \times \Sigma_{\mathfrak{B}}(i)$ and $[V]_i = \Sigma_{\mathfrak{B}}(i)$. $\Sigma_{\mathfrak{B}}(0) = \{u_0 | u_0 \in G_0 \cup Y_0\}$ and the members σ_j of the production sets $\Sigma_{\mathfrak{B}}(j)$ $(j \geqq 1)$ have the form:

$$\sigma_j = (b_j \ldots b_1, u_0, u_j[S_{u_j}, \varphi_j], \varrho_j, W_j, \tau_j), \quad \text{where}$$

$b_i \in A_i$ for $i = 1, \ldots, j$, $u_0 \in G_0 \cup Y_0$,

$u_j \in P_{G_j}(Y_j)$, φ_j: $S_{u_j} \to A_j \ldots A_1\{1, \ldots, \nu(u_0)\}$,

W_j is a finite subset of $(N^*)^2 \cup \ldots \cup (N^*)^{j+1}$, where $(N^*)^1 = N^*$ and for each $l \geqq 1$, $(N^*)^{l+1} = (N^*)^l \times N^*$.

$$\varrho_j\colon S_{u_j} \to W_j,\ \tau_j\colon W_j \to (A_j \ldots A_2 A_1 \cup \ldots \cup A_2 A_1 \cup A_1)[W_j]_0,$$

and the following requirements are satisfied:

a) $j = 1$

i) $W_1 = \{(t_0, t_1) | t_1 \in S_{u_1}, \varphi_1(t_1) = c_1 t_0$ for some $c_1 \in A_1\}$,

ii) for every $t_1 \in S_{u_1}$ if $\varphi_1(t_1) = c_1 t_0$ then $\varrho_1(t_1) = (t_0, t_1)$,

iii) $\varphi_1 = \varrho_1 \circ \tau_1$,

iv) $V_1 = \{(u_0, \sigma_1) | u_0 \in G_0 \cup Y_0$, and the second component of $\sigma_1 (\in \Sigma_{\mathfrak{B}}(1))$ is $u_0\}$.

b) $j > 1$ if $(u_0, \sigma_1, \ldots, \sigma_{j-1}, \sigma_j) \in V_j$ and σ_{j-1} has the form $(b_{j-1} \ldots b_1, u_0, u_{j-1}[S_{u_{j-1}}, \varphi_{j-1}], \varrho_{j-1}, W_{j-1}, \tau_{j-1})$ then $(u_0, \sigma_1, \ldots, \sigma_{j-1}) \in V_{j-1}$ and there is a mapping $\varepsilon_j\colon S_{u_j} \to A_j[W_{j-1}]_{j-1}$ such that i)—iv) hold:

i) $W_j = \{(t_0, \ldots, t_{j-1}, t_j) | \varepsilon_j(t_j) = c_j t_{j-1}, c_j \in A_j, t_j \in S_{u_j}, t_{j-1} \in S_{u_{j-1}},$

$\varrho_{j-1}(t_{j-1}) = (t_0, \ldots, t_{j-1}, t_j)\} \cup$

$\cup \{(t_0, \ldots, t_l) \in W_{j-1} | 1 \leqq l \leqq j-2\} \cup$

$\cup \{(t_0, \ldots, t_{j-1}) \in W_{j-1} |$ there are no $t_j \in S_{u_j}$ and $c_j \in A_j$

such that $\varepsilon_j(t_j) = c_j t_{j-1}\}$.

ii) For

$\tau_j, \tau_j|_{W_j \cap W_{j-1}} = \tau_{j-1}|_{W_j \cap W_{j-1}}$ and

if $(t_0, \ldots, t_{j-1}, t_j) \in W_j$, $\varepsilon_j(t_j) = c_j t_{j-1}$ $(c_j \in A_j, t_{j-1} \in [W_{j-1}]_{j-1})$ and

$\tau_{j-1}((t_0, \ldots, t_{j-1})) = c_{j-1} \ldots c_1 t_0$ then $\tau_j((t_0, \ldots, t_{j-1}, t_j)) = c_j \ldots c_1 t_0$.

iii) For each $t_j \in S_{u_j}$ if

$\varepsilon_j(t_j) = c_j t_{j-1} (c_j \in A_j, t_{j-1} \in [W_{j-1}]_{j-1})$ and

$\varrho_{j-1}(t_{j-1}) = (t_0, \ldots, t_{j-1})$ then $\varrho_j(t_j) = (t_0, \ldots, t_{j-1}, t_j)$.

iv) $\varphi_j = \varrho_j \circ \tau_j$.

(One can see that for each $t_j \in S_{u_j}$, $\varepsilon_j(t_j) = c_j t_{j-1}$ $(c_j \in A_j, t_{j-1} \in S_{u_{j-1}})$ iff $\varrho_j(t_j) = (t_0, \ldots, t_{j-1}, t_j)$ and $\tau_j((t_0, \ldots, t_{j-1}, t_j)) = c_j \ldots c_1 t_0$.)

In the rest of the paper we shall denote the arity function of G_0 by ν.

Definition 3.2. Let $\mathfrak{B}$ be a k-synchronized R-transducer as in Definition 3.1. A configuration of $\mathfrak{B}$ is a system $(q[S_q, \psi], \Theta, Z, \Omega)$, where $q \in P_{G_k}(Y_k)$, $\psi\colon S_q \to A_k \ldots A_1 T_{G_0}(Y_0)$, $\Theta\colon S_q \to Z$; for each $s_k \in S_q$, $\Theta(s_k) = (s_0, \ldots, s_{k-1}, s_k)$ for some $s_0, \ldots, s_{k-1} \in N^*$.

Z is a finite subset of $(N^*)^2 \cup \ldots \cup (N^*)^k \cup (N^*)^{k+1}$ such that the following two conditions hold:

i) for $j = 0, \ldots, k$ and arbitrary $s_j, \bar{s}_j \in [Z]_j$ if $s_j = \bar{s}_j \bar{\bar{s}}_j$, then $s_j = \bar{s}_j$ and $\bar{\bar{s}}_j = e$,

ii) for each $s_k \in S_q$ $\Theta(s_k)$ is the only element of Z which has the form $(s_0, \ldots, s_{k-1}, s_k)$ for some $s_0, \ldots, s_{k-1} \in N^*$.

Ω: $Z \to (A_k A_{k-1} \ldots A_1 \cup A_{k-1} \ldots A_1 \cup \ldots \cup A_1) T_{G_0}(Y_0)$ is a mapping such that $\psi = \Theta \circ \Omega$ holds, that is, the diagram in Figure 3 is commutative.

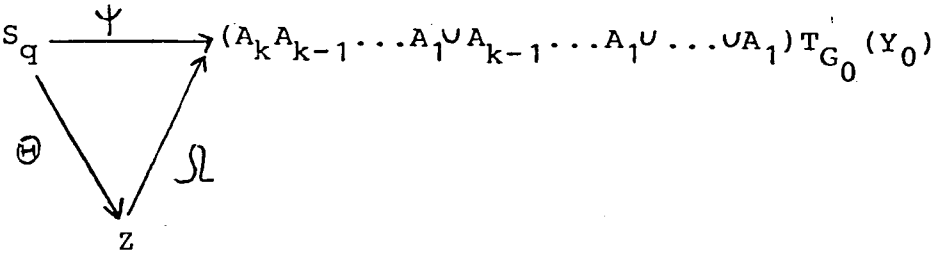


Figure 3

A configuration $(q[S_q, \psi], \Theta, Z, \Omega)$ is said to be a starting configuration, if q is the quasi tree $e \in N^*$ (empty word) and $\psi(e) \in A'_k \ldots A'_1 T_{G_0}(Y_0)$, moreover $Z = \{\underbrace{(e, \ldots, e)}_{k \text{ times}}\}$.

Definition 3.3. Let $K_1 = (q^1[S_{q^1}, \psi^1], \Theta^1, Z^1, \Omega^1)$ and $K_2 = (q^2[S_{q^2}, \psi^2], \Theta^2, Z^2, \Omega^2)$ be configurations of a k-synchronized R-transducer $\mathfrak{B} = (G_0, G_1, \ldots, G_k, Y_0, Y_1, \ldots, Y_k, A_1, \ldots, A_k, A'_1, \ldots, A'_k, \Sigma_{\mathfrak{B}}, V)$. It is said that there is a transition from K_1 to K_2 in $\mathfrak{B}$ which is denoted by $K_1 \Rightarrow_{\mathfrak{B}} K_2$ if there are mappings $\varkappa_j$: $[Z^1]_j \to \Sigma_{\mathfrak{B}}(j)$ for $j = 0, 1, \ldots, k$ such that the following requirements hold:

(1) For each $(s_0, s_1, \ldots, s_j) \in Z^1$ $(1 \leqq j \leqq k)$ if

$$\Omega^1((s_0, s_1, \ldots, s_j)) = b_j \ldots b_1 \, u_0(1, \ldots, \nu(u_0))[\{1, \ldots, \nu(u_0)\}, \vartheta_0] \text{ for some}$$

$$u_0 \in G_0 \cup Y_0, \; b_j \ldots b_1 \in A_j \ldots A_1 \text{ and } \vartheta_0 \colon \{1, \ldots, \nu(u_0)\} \to T_{G_0}(Y_0) \text{ then}$$

$$\varkappa_0(s_0) = u_0, \; \varkappa_i(s_i) = (b_i \ldots b_1, u_0, u_i[S_{u_i}, \varphi_i], \varrho_i, W_i, \tau_i)$$

for some u_i, ϱ_i, W_i and τ_i $(i = 1, 2, \ldots, j)$, and $(\varkappa_0(s_0), \varkappa_1(s_1), \ldots, \varkappa_j(s_j)) \in V_j$.

(2) $q^2 = q^1[S_{q^1}, \xi]$ for the mapping ξ: $S_{q^1} \to T_{G_k}(Y_k \cup N^*)$ which is defined by the following formula: for every $s_k \in S_{q^1}$, $\xi(s_k) = \omega_{s_k}(u_k)$ if $\varkappa_k(s_k) = (b_k \ldots b_1, u_0, u_k[S_{u_k}, \varphi_k], \varrho_k, W_k, \tau_k)$.

(3) $Z^2 = \{(s_0 t_0, s_1 t_1, \ldots, s_j t_j) | (s_0, s_1, \ldots, s_l) \in Z^1$ for some l, $(1 \leqq j \leqq l \leqq k)$ and $\varkappa_l(s_l) = (b_l \ldots b_1, u_0, u_l[S_{u_l}, \varphi_l], \varrho_l, W_l, \tau_l)$ and $(t_0, t_1, \ldots, t_j) \in W_l\}$.

(4) For each $\bar{s}_k = S_{q^2}$ consider its unique decomposition $\bar{s}_k = s_k t_k$, where $s_k \in S_{q^1}$, $\varkappa_k(s_k)$ has the form $\varkappa_k(s_k) = (b_k \ldots b_1, u_0, u_k[S_{u_k}, \varphi_k], \varrho_k, W_k, \tau_k)$, $\xi_k(s_k) = \omega_{s_k}(u_k)$, and $t_k \in S_{u_k}$. If $\Theta^1(s_k) = (s_0, s_1, \ldots, s_k)$ and $\varrho_k(t_k) = (t_0, t_1, \ldots, t_k)$ then $\Theta^2(\bar{s}_k) = (s_0 t_0, s_1 t_1, \ldots, s_k t_k)$.

(5) Let $\bar{s}_k \in S_{q^2}$ be arbitrary and consider its unique decomposition $\bar{s}_k = s_k t_k$, where $s_k \in S_{q^1}$, $\varkappa_k(s_k)$ has the form $\varkappa_k(s_k) = (b_k \ldots b_1, u_0, u_k[S_{u_k}, \varphi_k], \varrho_k, W_k,$

$\tau_k)$, $\xi_k(s_k)=\omega_{s_k}(u_k)$ and $t_k\in S_{u_k}$. If $\varphi_k(t_k)=c_k\dots c_1 t_0$ and

$\psi^1(s_k)=u_0(1,\dots,\nu(u_0))[\{1,\dots,\nu(u_0)\},\vartheta_0]$

$(u_0\in G_0\cup Y_0,\ \vartheta_0\colon \{1,\dots,\nu(u_0)\}\to T_{G_0}(Y_0))$,

then $\psi^2(\bar{s}_k)=c_k\dots c_1\vartheta_0(t_0)$.

(6) For every $(\bar{s}_0,\bar{s}_1,\dots,\bar{s}_j)\in Z^2$ there is a vector $(s_0,s_1,\dots,s_l)\in Z^1$ for some l $(1\leqq j\leqq l\leqq k)$ such that $\varkappa_l(s_l)=(b_l\dots b_1,u_0,u_l[S_{u_l},\varphi_l],\varrho_l,W_l,\tau_l)$, and $\bar{s}_0=s_0t_0$, $\bar{s}_1=s_1t_1$, $\dots$, $\bar{s}_j=s_jt_j$ hold for some $(t_0,t_1,\dots,t_j)\in W_l$. If $\tau_l((t_0,t_1,\dots,t_j))=c_j\dots c_1t_0$ and

$\Omega^1((s_0,s_1,\dots,s_l))=b_l\dots b_1u_0(1,\dots,\nu(u_0))[\{1,\dots,\nu(u_0)\},\vartheta_0]$

for some $u_0\in G_0\cup Y_0$ and $\vartheta_0\colon\{1,\dots,\nu(u_0)\}\to T_{G_0}(Y_0)$ then

$\Omega^2((\bar{s}_0,\bar{s}_1,\dots,\bar{s}_j))=c_j\dots c_1\vartheta_0(t_0)$.

Notice, that given configuration K_1 and mappings $\varkappa_i$, for $i=0,\dots,k$ satisfying condition (1), uniquely determine configuration K_2.

The reflexive and transitive closure of relation $\Rightarrow_{\mathfrak{B}}$ between configurations is denoted by $\Rightarrow_{\mathfrak{B}}^*$.

Definition 3.4. Take a k-synchronized R-transducer

$$\mathfrak{B}=(G_0,G_1,\dots,G_k,Y_0,Y_1,\dots,Y_k,A_1,\dots,A_k,A_1',\dots,A_k',\Sigma_{\mathfrak{B}},V).$$

Then the relation

$$\tau_{\mathfrak{B}}=\{(p,q)\mid p\in T_{G_0}(Y_0),\ q\in T_{G_k}(Y_k),$$
$$K_0=(e[\{e\},\psi_0\colon e\mapsto bp],\Theta^0,Z^0,\Omega^0)\Rightarrow_{\mathfrak{B}}^*(q,\emptyset,\emptyset,\emptyset)$$
$$\text{for some starting configuration } K_0\}$$

is called the transformation induced by $\mathfrak{B}$.

Configurations of the form $(q,\emptyset,\emptyset,\emptyset)$, where $q\in T_{G_k}(Y_k)$, are said to be final.

Theorem 3.5. Let $\mathfrak{A}_i=(G_{i-1},Y_{i-1},A_i,G_i,Y_i,A_i',\Sigma_{\mathfrak{A}_i})$ $(i=1,\dots,k,k\geqq 2)$ be R-transducers. Then there is a k-synchronized R-transducer $\mathfrak{B}$ such that $\tau_{\mathfrak{B}}=\tau_{\mathfrak{A}_1}\circ\dots\circ\tau_{\mathfrak{A}_k}$.

Proof: We construct a k-synchronized R-transducer $\mathfrak{B}$ as follows:

$$\mathfrak{B}=(G_0,G_1,\dots,G_k,Y_0,Y_1,\dots,Y_k,A_1,\dots,A_k,A_1',\dots,A_k',\Sigma_{\mathfrak{B}},V),$$

where $\Sigma_{\mathfrak{B}}(0)=\Sigma(0),\dots,\Sigma_{\mathfrak{B}}(k)=\Sigma(k)$ for the sets $\Sigma(i)$, which are defined in the previous chapter. $V=V_0\cup V_1\cup\dots\cup V_k$, where the sets $V_0,V_1,\dots,V_k$ are defined in the previous chapter. We may assume without loss of generality that $A_k\dots A_1\cap T_{G_k}(Y_k)=\emptyset$ and that, for $i=1,\dots,k$, $A_i\dots A_1\cap T_{G_0}(Y_0)=\emptyset$. Thus $\mathfrak{B}$ satisfies requirement (3) of Definition 3.1.

First we shall prove the inclusion

$$\tau_{\mathfrak{A}_1}\circ\dots\circ\tau_{\mathfrak{A}_k}\subseteq\tau_{\mathfrak{B}}.$$

Assume that $(p_0, p_k) \in \tau_{\mathfrak{A}_1} \circ \tau_{\mathfrak{A}_2} \circ \ldots \circ \tau_{\mathfrak{A}_k}$. Then there are initial states $a_1 \in A_1', \ldots, a_k \in A_k'$ and there is a derivation sequence D: $a_1 p_0 \Rightarrow_{\mathfrak{A}_1}^* p_1$, $a_2 p_1 \Rightarrow_{\mathfrak{A}_2}^* p_2, \ldots, a_k p_{k-1} \Rightarrow_{\mathfrak{A}_k}^* p_k$, where $p_i \in T_{G_i}(Y_i)$ for $i=0, \ldots, k$.

Take an arbitrary decomposition $p_0 = q_0[S_{q_0}, \gamma_0]$ of the tree p_0, where $q_0 \in P_{G_0}(Y_0)$ and γ_0: $S_{q^0} \to T_{G_0}(Y_0)$. We have constructed a configuration

$$K_{(D, q_0)} = (q_k[S_{q_k}, \psi_{(D, q_0)}], \Theta_{(D, q_0)}, Z_{(D, q_0)}, \Omega_{(D, q_0)})$$

for D and q_0 in Chapter 2.

One can see that $K_{(D, q_0)}$ is a configuration of the k-synchronized R-transducer $\mathfrak{B}$. Let $r_0 = q_0[S_{q^0}, \xi_0]$ for the mapping ξ_0: $S_{q_0} \to T_{G_0}(Y_0 \cup N^*)$ which is defined by $\xi_0(s_0) = \omega_{s_0}(u_0(1, \ldots, v(u_0)))$ for each $s_0 \in S_{q_0}$, where

$$\gamma_0(s_0) = u_0(1, \ldots, v(u_0))[\{1, \ldots, v(u_0)\}, \vartheta_0],$$

$$(u_0 \in G_0 \cup Y_0, \vartheta_0\colon \{1, \ldots, v(u_0)\} \to T_{G_0}(Y_0)).$$

$K_{(D, r_0)}$ is again a configuration of $\mathfrak{B}$.

It follows from the definition of the relation $\Rightarrow_{\mathfrak{B}}$ that

$$K_{(D, q_0)} = K_{(D, r_0)} \quad \text{or} \quad K_{(D, q_0)} \Rightarrow_{\mathfrak{B}} K_{(D, r_0)}$$

holds.

Let $p^0, p^1, \ldots, p^l \in P_{G_0}(Y_0)$ be quasi trees for $l = h(p)+1$ such that for every i

$$(0 \leqq i \leqq l), \quad p_0 = p^i[S_{p^i}, \gamma^i] \quad (\gamma^i\colon S_{p^i} \to T_{G_0}(Y_0)),$$

where

i) $p^0 = e$, $\gamma^0(e) = p_0$, and

ii) $p^{i+1} = p^i[S_{p^i}, \xi^{i+1}]$ for the mapping ξ^{i+1}: $S_{p^i} \to T_{G_0}(Y_0 \cup N^*)$ such that for every $s^i \in S_{p^i}$

$$\xi^{i+1}(s^i) = \omega_{s^i}(u_0(1, \ldots, v(u_0))),$$

where

$$\gamma^i(s^i) = u_0(1, \ldots, v(u_0))[\{1, \ldots, v(u_0)\}, \vartheta_0]$$

for some $u_0 \in G_0 \cup Y_0$ and ϑ_0: $\{1, \ldots, v(u_0)\} \to T_{G_0}(Y_0)$. In this case every $s^{i+1} \in S_{p^{i+1}}$ has a unique decomposition $s^{i+1} = s^i t^i$,

$$s^i \in S_{p^i}, \gamma^i(s^i) = u_0(1, \ldots, v(u_0))[\{1, \ldots, v(u_0)\}, \vartheta_0]$$

for some

$$u_0 \in G_0 \cup Y_0, \vartheta_0\colon \{1, \ldots, v(u_0)\} \to T_{G_0}(Y_0) \quad \text{and} \quad t^i \in \{1, \ldots, v(u_0)\}.$$

Then $\gamma^{i+1}(s^{i+1}) = \vartheta_0(t^i)$.

We know that $K_{(D, p^i)} = K_{(D, p^{i+1})}$ or $K_{(D, p^i)} \Rightarrow_{\mathfrak{B}} K_{(D, p^{i+1})}$ holds for $i=0, \ldots, l-1$. It has remained to prove that $K_{(D, p^0)}$ is a starting configuration and $K_{(D, p^l)}$ is a final configuration of $\mathfrak{B}$. The first part of the statement trivially holds. Since $p^l = p_0$ and $p_0 \in T_{G_0}(Y_0)$, S_{p^l} must be the empty set, thus $K_{(D, p^l)} = (p_k, \emptyset, \emptyset, \emptyset)$. We have proved that $(p_0, p_k) \in \tau_{\mathfrak{B}}$.

We shall prove the reverse inclusion:

$$\tau_{\mathfrak{B}} \subseteq \tau_{\mathfrak{A}_1} \circ \tau_{\mathfrak{A}_2} \circ \ldots \circ \tau_{\mathfrak{A}_k}.$$

Let $K_0 \Rightarrow_{\mathfrak{B}} K_1 \Rightarrow_{\mathfrak{B}} \ldots \Rightarrow_{\mathfrak{B}} K_n$ be a sequence of transitions in $\mathfrak{B}$, where $n \geqq 1$, $K_0 = (e[\{e\}, \psi^0], \Theta^0, Z^0, \Omega^0)$ is a starting configuration and $K_i = (q^i[S_{q^i}, \psi^i], \Theta^i, Z^i, \Omega^i)$ for $i = 1, \ldots, n$. Assume that $\psi^0(e) = a_k \ldots a_1 p$. Let $p^0, p^1, \ldots, p^l$ be the sequence of quasi trees constructed in the first part of the proof, where $p^l = p$. It can be seen that $n \leqq l$. Then there is a derivation sequence $D = D_1, \ldots, D_k$,

$$D_1\colon a_1 p^n \Rightarrow^*_{\mathfrak{A}_1} p_1[S_{p^1}, \eta_1],\ (p_1 \in P_{G_1}(Y_1),\ \eta_1\colon S_{p_1} \to A_1 S_{p^n}),$$

$$D_2\colon a_2 p_1 \Rightarrow^*_{\mathfrak{A}_2} p_2[S_{p_2}, \eta_2],\ (p_2 \in P_{G_2}(Y_2),\ \eta_2\colon S_{p_2} \to A_2 S_{p_1}),$$

$$\vdots$$

$$D_k\colon a_k p_{k-1} \Rightarrow^*_{\mathfrak{A}_k} p_k[S_{p_k}, \eta_k],\ (p_k \in P_{G_k}(Y_k),\ \eta_k\colon S_{p_k} \to A_k S_{p_{k-1}})$$

such that the following equalities hold:

i) $p_k = q^n$,

ii) $\psi_{(D, p^n)} = \psi^n$,

iii) $Z^n = Z_{(D, p^n)}$,

iv) $\Theta^n = \Theta_{(D, p^n)}$,

v) $\Omega^n = \Omega_{(D, p^n)}$,

where the sets $Z_{(D, p^n)}$, $\Theta_{(D, p^n)}$ and the mappings

$$\psi_{(D, p^n)}\colon S_{p_k} \to (A_k \ldots A_2 A_1 \cup \ldots \cup A_2 A_1 \cup A_1) S_{p^n},$$

$$\Omega_{(D, p^n)}\colon Z_{(D, p^n)} \to (A_k \ldots A_2 A_1 \cup \ldots \cup A_2 A_1 \cup A_1)\, \mathrm{rg}(\gamma^n)$$

are defined as follows:

(1) $Z_{(D, p^n)} = \{(s_0, s_1, \ldots, s_j) | s_0 \in S_{p^n}, s_1 \in S_{p_1}, \ldots, s_j \in S_{p_j},\ 1 \leqq j \leqq k,$

and $(j = k$ or $(j < k$ and there are no $s_{j+1} \in S_{q_{j+1}}$ and $b_{j+1} \in A_{j+1}$ such that $\eta_{j+1}(s_{j+1}) = b_{j+1} s_j))$ and $\eta_i(s_i) = b_i s_{i-1}$ $(b_i \in A_i)$ for $i = 1, \ldots, j\}$.

(2) For every $(s_0, s_1, \ldots, s_j) \in Z_{(D, p^n)}$ $(1 \leqq j \leqq k)$, $\Omega_{(D, p^n)}((s_0, s_1, \ldots, s_j)) = b_j \ldots b_1 \gamma^n(s_0)$ iff $\eta_i(s_i) = b_i s_{i-1} (b_i \in A_i)$ for $i = 1, \ldots, j$.

(3) For every $s_k \in S_{q^n}$, $\Theta_{(D, p^n)}(s_k) = (s_0, s_1, \ldots, s_k)$ iff

$$\eta_i(s_i) = b_i s_{i-1} (b_i \in A_i) \quad \text{for} \quad i = 1, \ldots, k.$$

(4) $\psi_{(D, p^n)} = \Theta_{(D, p^n)} \circ \Omega_{(D, p^n)}$.

We proceed by induction on n. Let $n = 1$. In this case $p^1 = u_0(1, \ldots, \nu(u_0))$, $u_0 = \mathrm{root}(p)$. From the definition of the transition in $\mathfrak{B}$ it follows that there are mappings $\varkappa_i\colon \{e\} \to \Sigma_{\mathfrak{B}}(i)$ $(i = 0, 1, \ldots, k)$ such that $\varkappa_0(e) = u_0$,

$$\varkappa_1(e) = (a_1, u_0, u_1[S_{u_1}, \varphi_1], \varrho_1, W_1, \tau_1),$$

and so on,

$$\varkappa_k(e) = (a_k \ldots a_1, u_0, u_k[S_{u_k}, \varphi_k], \varrho_k, W_k, \tau_k),$$

and $(\varkappa_0(e), \varkappa_1(e), \ldots, \varkappa_k(e)) \in V_k$, and configuration K_0 and mappings $\varkappa_i$ $(i=0, \ldots, k)$ determine the configuration K_1.

According to the construction of the transducer $\mathfrak{B}$ and the definition of the transition in $\mathfrak{B}$, there is a derivation sequence $D=D_1, \ldots, D_k$,

$$D_1\colon\ a_1 u_0(1, \ldots, \nu(u_0)) \Rightarrow^*_{\mathfrak{A}_1} u_1[S_{u_1}, \varphi_1],$$

$$D_i\colon\ a_i u_{i-1} \Rightarrow^*_{\mathfrak{A}_i} u_i[S_{ui}, \eta_i]$$

for some

$$\eta_i\colon\ S_{u_i} \to A_i S_{u_{i-1}}, \quad (i = 2, \ldots, k)$$

such that the following equalities hold:

i) $q^1 = u_k$,

ii) $\psi_{(D, p^1)} = \psi^1$,

iii) $Z^1 = W_k = Z_{(D, p^1)}$,

iv) $\Theta^1 = \varrho_k = \Theta_{(D, p^1)}$,

v) $\Omega^1 = \tau_k = \Omega_{(D, p^1)}$.

The proof of the basic step is complete.

Assume that the statement is true for $n-1$. It means that there is a derivation sequence

$$D_1\colon\ a_1 p^{n-1} \Rightarrow^*_{\mathfrak{A}_1} p_1[S_{p_1}, \eta_1]\ (p_1 \in P_{G_1}(Y_1), \eta_1\colon\ S_{p_1} \to A_1 S_{p^{n-1}}),$$

$$D_2\colon\ a_2 p_1 \Rightarrow^*_{\mathfrak{A}_2} p_2[S_{p_2}, \eta_2]\ (p_2 \in P_{G_2}(Y_2), \eta_2\colon\ S_{p_2} \to A_2 S_{p_1}),$$

$$\vdots$$

$$D_k\colon\ a_k p_{k-1} \Rightarrow^*_{\mathfrak{A}_k} p_k[S_{p_k}, \eta_k]\ (p_k \in P_{G_k}(Y_k), \eta_k\colon\ S_{p_k} \to A_k S_{p_{k-1}})$$

such that the following equalities hold:

i) $p_k = q^{n-1}$,

ii) $\psi_{(D, p^{n-1})} = \psi^{n-1}$,

iii) $Z^{n-1} = Z_{(D, p^{n-1})}$,

iv) $\Theta^{n-1} = \Theta_{(D, p^{n-1})}$,

v) $\Omega^{n-1} = \Omega_{(D, p^{n-1})}$.

Because of the transition $K_{n-1} \Rightarrow_{\mathfrak{B}} K_n$ there are mappings $\varkappa_i\colon [Z^{n-1}]_i \to \Sigma_{\mathfrak{B}}(i)$ $(i=0, \ldots, k)$ which satisfy condition (1) in Definition 3.3.

Take the sequence $r_0, \ldots, r_k$ of quasi trees given as follows:

$$r_0 = p^n, \quad \text{for } i = 2, \ldots, k \quad \text{let} \quad r_i = p_i[S_{p_i}, \varepsilon_i],$$

where $\varepsilon_i\colon\ S_{p_i} \to T_{G_i}(Y_i \cup N^*)$ such that for every $s \in S_{p_i}$, $\varepsilon_i(s) = \omega_s(u_i)$ holds, where $u_i[S_{u_i}, \varphi_i]$ is the third component of

$$\varkappa_i(s) = (b_i \ldots b_1, u_0, u_i[S_{u_i}, \varphi_i]\, \varrho_i, W_i, \tau_i).$$

Let ξ_i: $S_{r_i} \to A_i S_{r_{i-1}}$ $(i \in \{1, \ldots, k\})$ be the mapping satisfying the following requirements: we know that for each $\bar{s}_i \in S_{r_i}$ there is a unique decomposition $\bar{s}_i = s_i t_i$ of $\bar{s}_i$, where $s_i \in S_{p_i}$, $\varkappa_i(s_i) = (b_i \ldots b_1, u_0, u_i[S_{u_i}, \varphi_i], \varrho_i, W_i, \tau_i)$ and $t_i \in S_{u_i}$. If $\varrho_i(t_i) = (t_0, \ldots, t_{i-1}, t_i)$ $((t_0, \ldots, t_{i-1}, t_i) \in W_i)$ and $\tau_i((t_0, \ldots, t_{i-1}, t_i)) = c_i \ldots c_1 t_0$ for some $c_i \in A_i, \ldots, c_1 \in A_1$, and $\eta_i(s_i) = b_i s_{i-1}$ for some $s_{i-1} \in S_{p_{i-1}}$ then $\xi_i(s_i t_i) = c_i s_{i-1} t_{i-1}$. We obtain that for $i = 1, \ldots, k$, E_i: $a_i r_{i-1} \Rightarrow^*_{\mathfrak{A}_i} r_i[S_{r_i}, \xi_i]$ holds.

From the definition of the transition in $\mathfrak{B}$ and from the definitions of r_k, $Z_{(E, p^n)}$, $\Theta_{(E, p^n)}$, $\Omega_{(E, p^n)}$, $\psi_{(E, p^n)}$ it follows that

i) $r_k = q^n$,

ii) $\psi_{(E, p^n)} = \psi^n$,

iii) $Z^n = Z_{(E, p^n)}$,

iv) $\Theta^n = \Theta_{(E, p^n)}$,

v) $\Omega^n = \Omega_{(E, p^n)}$.

Assume that $(p, q) \in \tau_{\mathfrak{B}}$. Then there are configurations $K_0, \ldots, K_n$ $(n \geqq 1)$ such that K_0 is a starting configuration, $K_0 = (e[\{e\}, \psi^0], \Theta^0, Z^0, \Omega^0)$ where $\psi^0(e) = a_k \ldots a_1 p$ for some $a_k \in A'_k, \ldots, a_1 \in A'_1$, K_n is a final configuration, $K_n = (q, \emptyset, \emptyset, \emptyset)$, moreover, $K_{i-1} \Rightarrow_{\mathfrak{B}} K_i$ holds for $i = 1, \ldots, n$.

According to the above proposition there is a derivation sequence

$$D = D_1, \ldots, D_k,$$

$$D_1\colon\ a_1 p^n \Rightarrow^*_{\mathfrak{A}_1} p_1[S_{p_1}, \eta_1], (p_1 \in P_{G_1}(Y_1), \eta_1\colon\ S_{p_1} \to A_1 S_{p^n}),$$

$$D_2\colon\ a_2 p_1 \Rightarrow^*_{\mathfrak{A}_2} p_2[S_{p_2}, \eta_2], (p_2 \in P_{G_2}(Y_2), \eta_2\colon\ S_{p_2} \to A_2 S_{p_1}),$$

$$\vdots$$

$$D_k\colon\ a_k p_{k-1} \Rightarrow^*_{\mathfrak{A}_k} p_k[S_{p_k}, \eta_k], (p_k \in P_{G_k}(Y_k), \eta_k\colon\ S_{p_k} \to A_k S_{p_{k-1}})$$

such that the following equalities hold:

i) $p_k = q$,

ii) $\psi_{(D, p^n)} = \psi^n = \emptyset$,

iii) $Z^n = Z_{(D, p^n)}$,

iv) $\Theta^n = \Theta_{(D, p^n)}$,

v) $\Omega^n = \Omega_{(D, p^n)}$.

Thus $Z_{(D, p^n)} = \emptyset$, $\Theta_{(D, p^n)} = \emptyset$, $\Omega_{(D, p^n)} = \emptyset$. According to the definition of $Z_{(D, p^n)}$, $S_{p^i} = [Z_{(D, p^n)}]_i$ for $i = 1, \ldots, k$. Thus $p_i \in T_{G_i}(Y_i)$ for $i = 1, \ldots, k$. One can see that $a_1 p^n[S_{p^n}, \gamma^n] \Rightarrow^*_{\mathfrak{A}_1} p_1$ holds. Thus $(p^n[S_{p^n}, \gamma^n], q) \in \tau_{\mathfrak{A}_1} \circ \tau_{\mathfrak{A}_2} \circ \ldots \circ \tau_{\mathfrak{A}_k}$. The proof of the theorem is complete.

Theorem 3.6. Let $\mathfrak{B} = (G_0, G_1, \ldots, G_k, Y_0, Y_1, \ldots, Y_k, A_1, \ldots, A_k, A'_1, \ldots, A'_k, \Sigma_{\mathfrak{B}}, V)$ be a k-synchronized R-transducer. Then there are R-transducers $\mathfrak{A}_1, \ldots, \mathfrak{A}_k$ such that $\tau_{\mathfrak{B}} = \tau_{\mathfrak{A}_1} \circ \ldots \circ \tau_{\mathfrak{A}_k}$.

Proof. The production sets $\Sigma_{\mathfrak{B}}(i)$ for $i=1, \ldots, k-1$ are considered to be operator domains with arity function ν^i: $\Sigma_{\mathfrak{B}}(i) \to \{0, 1, 2, \ldots\}$ as follows: for $1 \leqq i \leqq k-1$,

$$\sigma = (b_i \ldots b_1, u_0, u_i[S_{u_i}, \varphi_i], \varrho_i, W_i, \tau_i) \in \Sigma_{\mathfrak{B}}(i)$$

let $\nu^i(\sigma) = |S_{u_i}|$, where $|S_{u_i}|$ denotes the cardinality of the set S_{u_i}.

Remember, the arity function of the operator domain G_0 is denoted by ν.

Convention: Let $S \subseteq N^*$. If $S \neq \emptyset$, then ξ_0, ξ_k: $S \to S$ denote the identity function. If $S = \emptyset$ then ξ_0, ξ_k: $S \to S$ denote the empty function.

For every j $(1 \leqq j \leqq k)$ if $S \neq \emptyset$ then ξ_j: $S \to \{1, \ldots, |S|\}$ denotes the function whose value on $s \in S$ is the ordinal number of s in S with respect to the lexicographic ordering. If $S = \emptyset$ then ξ_j: $S \to S$ denotes the empty function. Thus ξ_j always denotes a bijective function which is determined by its domain.

Take the R-transducer

$$\mathfrak{A}_1 = (G_0, Y_0, A_1, \Sigma_{\mathfrak{B}}(1), \emptyset, \Sigma_{\mathfrak{A}_1}, A_1'),$$

where

$$\Sigma_{\mathfrak{A}_1} = \{b_1 u_0 \to \sigma_1(1, \ldots, \nu^1(\sigma_1))[\{1, \ldots, \nu^1(\sigma_1)\}, \beta_1]\|$$

σ_1 has the form $(b_1, u_0, u_1[S_{u_1}, \varphi_1], \varrho_1, W_1, \tau_1) \in \Sigma_{\mathfrak{B}}(1)$,
$(\sigma_0, (b_1, u_0, u_1[S_{u_1}, \varphi_1], \varrho_1, W_1, \tau_1)) \in V_1$ and the mapping
β_1: $\{1, \ldots, \nu^1(\sigma_1)\} \to A_1\{1, \ldots, \nu(\sigma_0)\}$ is defined as follows:
Let ξ_0: $\{1, \ldots, \nu(u_0)\} \to \{1, \ldots, \nu(u_0)\}$, ξ_1: $S_{u_1} \to \{1, \ldots, |S_{u_1}|\}$.
For each $t_1 \in S_{u_1}$, $\beta_1(\xi_1(t_1)) = c_1 \xi_0(t_0)$ iff $\varrho_1(t_1) = (t_0, t_1)$ and $\tau_1((t_0, t_1)) = c_1 t_0$. (Thus for each $t_1 \in S_{u_1}$, $\beta_1(\xi_1(t_1)) = c_1 \xi_0(t_0)$ iff $\varphi_1(t_1) = c_1 t_0$.)}

For $j=2, \ldots, k-1$ consider the R-transducer $\mathfrak{A}_j = (\Sigma_{\mathfrak{B}}(j-1), \emptyset, A_j, \Sigma_{\mathfrak{B}}(j), \emptyset, \Sigma_{\mathfrak{A}_j}, A_j')$, where the production set $\Sigma_{\mathfrak{A}_j}$ is defined as follows:

$$\Sigma_{\mathfrak{A}_j} = \{b_j \sigma_{j-1} \to \sigma_j(1, \ldots, \nu^j(\sigma_j))[\{1, \ldots, \nu^j(\sigma_j)\}, \beta_j]\|$$

There is an element $(\sigma_0, \ldots, \sigma_{j-1}, \sigma_j) \in V$ such that σ_{j-1} has the form $(b_{j-1} \ldots b_1, u_0, u_{j-1}[S_{u_{j-1}}, \varphi_{j-1}], \varrho_{j-1}, W_{j-1}, \tau_{j-1})$,
σ_j has the form $(b_j \ldots b_1, u_0, u_j[S_{u_j}, \varphi_j], \varrho_j, W_j, \tau_j)$.
There is a mapping ε_j: $S_{u_j} \to A_j[W_{j-1}]_{j-1}$ such that conditions i)—iv) in part (5).b of Definition 3.1 hold.
The mapping β_j: $\{1, \ldots, \nu^j(\sigma_j)\} \to A_j\{1, \ldots, \nu^{j-1}(\sigma_{j-1})\}$ is defined as follows:
Let ξ_{j-1}: $S_{u_{j-1}} \to \{1, \ldots, |S_{u_{j-1}}|\}$, ξ_j: $S_{u_j} \to \{1, \ldots, |S_{u_j}|\}$. For every $t_j \in S_{u_j}$, $\beta_j(\xi_j(t_j)) = c_j \xi_{j-1}(t_{j-1})$ $(c_j \in A_j, t_{j-1} \in S_{u_{j-1}})$ iff $\varrho_j(t_j) = (t_0, \ldots, t_{j-1}, t_j)$ and $\tau_j((t_0, \ldots, t_{j-1}, t_j)) = c_j \ldots c_1 t_0$.
(Thus for every $t_j \in S_{u_j}$, $\beta_j(\xi_j(t_j)) = c_j \xi_{j-1}(t_{j-1})$ $(c_j \in A_j, t_{j-1} \in S_{u_{j-1}})$ iff $\varepsilon_j(t_j) = c_j t_{j-1}$.)}

Take the R-transducer $\mathfrak{A}_k = (\Sigma_{\mathfrak{B}}(k-1), \emptyset, A_k, G_k, Y_k, \Sigma_{\mathfrak{A}_k}, A_k')$ where the production set $\Sigma_{\mathfrak{A}_k}$ is defined as follows:

$$\Sigma_{\mathfrak{A}_k} = \{a_k \sigma_{k-1} \to u_k[S_{u_k}, \beta_k]\|$$
There is an element $(\sigma_0, \ldots, \sigma_{k-1}, \sigma_k) \in V$ such that σ_{k-1} has the form $(b_{k-1} \ldots b_1, u_0, u_{k-1}[S_{u_{k-1}}, \varphi_{k-1}], \varrho_{k-1}, W_{k-1}, \tau_{k-1})$,

σ_k has the form $(b_k \dots b_1, u_0, u_k[S_{u_k}, \varphi_k], \varrho_k, W_k, \tau_k)$. There is a mapping ε_k: $S_{u_k} \to A_k[W_{k-1}]_{k-1}$ such that conditions i)—iv) in part (5).b of Definition 3.3 hold. The mapping β_k: $S_{u_k} \to A_k\{1, \dots, v^{k-1}(\sigma_{k-1})\}$ is defined as follows: Let ξ_{k-1}: $S_{u_{k-1}} \to \{1, \dots, |S_{u_{k-1}}|\}$, ξ_k: $S_{u_k} \to S_{u_k}$. For every $t_k \in S_{u_k}$, $\beta_k(t_k) = c_k \xi_{k-1}(t_{k-1})$ iff $\varrho_k(t_k) = (t_0, \dots, t_{k-1}, t_k)$ and $\tau_k((t_0, \dots, t_{k-1}, t_k)) = c_k \dots c_1 t_0$. (Thus for every $t_k \in S_{u_k}$, $\beta_k(t_k) = c_k \xi_{k-1}(t_{k-1})(c_k \in A_k, t_{k-1} \in S_{u_{k-1}})$ iff $\varepsilon_k(t_k) = c_k t_{k-1}$.)}

We may assume without loss of generality that for

$$i = 2, \dots, k-1, \; A_i \cap T_{\Sigma_{\mathfrak{B}}(i-1)}(\emptyset) = \emptyset, \quad A_i \cap T_{\Sigma_{\mathfrak{B}}(i)}(\emptyset) = \emptyset$$

and that $A_1 \cap T_{\Sigma_{\mathfrak{B}}(0)}(\emptyset) = \emptyset$, $A_k \cap T_{\Sigma_{\mathfrak{B}}(k-1)}(\emptyset) = \emptyset$.

Thus $\mathfrak{A}_1, \dots, \mathfrak{A}_k$ satisfy requirement (2) of Definition 1.10.

We shall prove that $\tau_{\mathfrak{B}} = \tau_{\mathfrak{A}_1} \circ \dots \circ \tau_{\mathfrak{A}_k}$. Let $\mathfrak{C}$ be the k-synchronized R-transducer that can be obtained from $\mathfrak{A}_1, \dots, \mathfrak{A}_k$ by the construction of Theorem 3.5. In this case

$$\mathfrak{C} = (G_0, \Sigma_{\mathfrak{B}}(1), \dots, \Sigma_{\mathfrak{B}}(k-1), G_k, Y_0, \underbrace{\emptyset, \dots, \emptyset}_{k-1 \text{ times}}, Y_k, A_1, \dots, A_k, A_1', \dots, A_k', \Sigma_{\mathfrak{C}}, \bar{V}).$$

We may assume without loss of generality that for $i = 1, \dots, k$, $A_i \dots A_1 \cap T_{G_0}(Y_0) = \emptyset$ and that $A_k \dots A_1 \cap T_{G_k}(Y_k) = \emptyset$. Thus $\mathfrak{C}$ satisfies the requirements of Definition 3.1.

By Theorem 3.5, $\tau_{\mathfrak{C}} = \tau_{\mathfrak{A}_1} \circ \tau_{\mathfrak{A}_2} \circ \dots \circ \tau_{\mathfrak{A}_k}$, so it is sufficient to prove that $\tau_{\mathfrak{B}} = \tau_{\mathfrak{C}}$. In order to prove this equality we shall introduce bijective mappings γ_j: $\Sigma_{\mathfrak{B}}(j) \to \Sigma_{\mathfrak{C}}(j)$ for $j = 0, \dots, k-1$ and a surjective mapping γ_k: $\Sigma_{\mathfrak{B}}(k) \to \Sigma_{\mathfrak{C}}(k)$, and we shall show that for $j = 0, \dots, k$ the mappings $\gamma_0, \dots, \gamma_j$ satisfy assumption (1) and that for $j = 0, \dots, k$ the mapping γ_j satisfies assumption (2).

(1) There are two cases.

Case 1. $0 \leqq j \leqq k-1$. In this case if $(\sigma_0, \dots, \sigma_j) \in V_j$ then $(\gamma_0(\sigma_0), \dots, \gamma_j(\sigma_j)) \in \bar{V}_j$, and if $(\bar{\sigma}_0, \dots, \bar{\sigma}_j) \in \bar{V}_j$ then $(\gamma_0^{-1}(\bar{\sigma}_0), \dots, \gamma_j^{-1}(\bar{\sigma}_j)) \in V_j$.

Case 2. $j = k$. In this case if $(\sigma_0, \dots, \sigma_k) \in V_k$ then $(\gamma_0(\sigma_0), \dots, \gamma_k(\sigma_k)) \in \bar{V}_k$, and if $(\bar{\sigma}_0, \dots, \bar{\sigma}_{k-1}, \bar{\sigma}_k) \in \bar{V}_k$ then there is a unique $\sigma_k \in \Sigma_{\mathfrak{B}}(k)$ such that $\gamma_k(\sigma_k) = \bar{\sigma}_k$ and $(\gamma_0^{-1}(\bar{\sigma}_0), \dots, \gamma_{k-1}^{-1}(\bar{\sigma}_{k-1}), \sigma_k) \in V_k$.

(2) There are three cases.

Case 1. $j = 0$. In this case $\Sigma_{\mathfrak{C}}(0) = \Sigma_{\mathfrak{B}}(0)$ and γ_0 is the identity function.

Case 2. $1 \leqq j \leqq k-1$. Let $(\sigma_0, \dots, \sigma_j) \in V_j$ and $\sigma_0 = u_0$. Assume that σ_l $(1 \leqq l \leqq j)$ has the form $(b_l \dots b_1, u_0, u_l[S_{u_l}, \varphi_l], \varrho_l, W_l, \tau_l)$. Let ξ_0: $\{1, \dots, v(u_0)\} = [W_j]_0 \to \{1, \dots, v(u_0)\}$, ξ_l: $S_{u_l} = [W_j]_l \to \{1, \dots, |S_{u_l}|\}$ for $l = 1, \dots, j$. Then $\gamma_j(\sigma_j)$ has the form $\gamma_j(\sigma_j) = (b_j \dots b_1, u_0, \sigma_j(1, \dots, v^j(\sigma_j))[\{1, \dots, v^j(\sigma_j)\}, \bar{\varphi}_j], \bar{\varrho}_j, \bar{W}_j, \bar{\tau}_j)$, and the following hold:

i) $[\bar{W}_j]_0 = [W_j]_0$ and for $i = 1, \dots, j-1$, $[\bar{W}_j]_i = \{1, \dots, |[W_j]_i|\} = \{1, \dots, |S_{u_i}|\} = \operatorname{rg}(\xi_i)$.

ii) $(t_0, t_1, \dots, t_l) \in W_j$ iff $(\xi_0(t_0), \xi_1(t_1), \dots, \xi_l(t_l)) \in \bar{W}_j$ $(1 \leqq l \leqq j)$ $(t_0 \in N^*, t_1 \in N^*, \dots, t_l \in N^*)$.

iii) For every

$$t_j \in S_{u_j},\ \varphi_j(t_j) = c_j \dots c_1 t_0 \quad \text{iff} \quad \bar{\varphi}_j(\xi_j(t_j)) = c_j \dots c_1 \xi_0(t_0),$$

$$(t_0 \in S_{u_0},\ c_1 \in A_1,\ \dots,\ c_j \in A_j).$$

iv) For each

$$t_j \in S_{u_j},\ \varrho_j(t_j) = (t_0, t_1, \dots, t_j) \quad \text{iff} \quad \bar{\varrho}_j(\xi_j(t_j)) = (\xi_0(t_0), \xi_1(t_1), \dots, \xi_j(t_j)).$$

v) For every

$$(t_0, t_1, \dots, t_l) \in W_j \quad (1 \leqq l \leqq j),\quad \tau_j((t_0, t_1, \dots, t_l)) = c_l \dots c_1 t_0 \quad \text{iff}$$

$$\bar{\tau}_j((\xi_0(t_0), \xi_1(t_1), \dots, \xi_l(t_l))) = c_l \dots c_1 \xi_0(t_0),$$

$$(t_0 \in \{1, \dots, \nu(u_0)\},\ c_1 \in A_1,\ \dots,\ c_l \in A_l).$$

Case 3. $j=k$. Let $(\sigma_0, \sigma_1, \dots, \sigma_k) \in V_k$ and $\sigma_0 = u_0$.
Assume that σ_l $(1 \leqq l \leqq k)$ has the form

$$(b_l \dots b_1, u_0, u_l[S_{u_l}, \varphi_l], \varrho_l, W_l, \tau_l).$$

Let

$$\xi_0\colon \{1, \dots, \nu(u_0)\} = [W_k]_0 \to \{1, \dots, \nu(u_0)\},$$

$$\xi_l\colon S_{u_l} = [W_k]_l \to \{1, \dots, |S_{u_l}|\} \quad \text{for} \quad l = 1, \dots, k-1,\ \xi_k\colon S_{u_k} = [W_k]_k \to S_{u_k}.$$

Then $\gamma_k(\sigma_k)$ has the form $\gamma_k(\sigma_k) = (b_k \dots b_1, u_0, u_k[S_{u_k}, \bar{\varphi}_k], \bar{\varrho}_k, \overline{W}_k, \bar{\tau}_k)$ and the following hold:

i) $[\overline{W}_k]_0 = [W_k]_0$, for $i = 1, \dots, k-1$, $[\overline{W}_k]_i = \{1, \dots, [W_k]_i\} = \{1, \dots, |S_{u_i}|\} = \operatorname{rg}(\xi_i)$ and $[\overline{W}_k]_k = [W_k]_k = S_{u_k} = \operatorname{rg}(\xi_k)$.

ii) $(t_0, t_1, \dots, t_l) \in W_k$ iff $(\xi_0(t_0), \xi_1(t_1), \dots, \xi_l(t_l)) \in \overline{W}_k$

$(1 \leqq l \leqq k,\ t_0, t_1, \dots, t_l \in N^*)$.

iii) For every

$$t_k \in S_{u_k},\ \varphi_k(t_k) = c_k \dots c_1 t_0 \quad \text{iff} \quad \bar{\varphi}_k(\xi_k(t_k)) = c_k \dots c_1 \xi_0(t_0),$$

$$(t_0 \in S_{u_0},\ c_1 \in A_1,\ \dots,\ c_k \in A_k).$$

iv) For each

$$t_k \in S_{u_k},\ \varrho_k(t_k) = (t_0, t_1, \dots, t_k) \quad \text{iff} \quad \bar{\varrho}_k(\xi_k(t_k)) = (\xi_0(t_0), \xi_1(t_1), \dots, \xi_k(t_k)).$$

v) For every

$$(t_0, t_1, \dots, t_l) \in W_k \quad (1 \leqq l \leqq k),\ \tau_k((t_0, t_1, \dots, t_l)) = c_l \dots c_1 t_0 \quad \text{iff}$$

$$\bar{\tau}_k((\xi_0(t_0), \xi_1(t_1), \dots, \xi_l(t_l))) = c_l \dots c_1 t_0,$$

$$(t_0 \in \{1, \dots, \nu(u_0)\},\ c_1 \in A_1,\ \dots,\ c_l \in A_l).$$

We shall define mappings $\gamma_j\colon \Sigma_{\mathfrak{B}}(j) \to \Sigma_{\mathfrak{C}}(j)$ according to the construction of $\Sigma_{\mathfrak{B}}(j)$ and $\mathfrak{A}_j$.

Let $j=0$. Since $\Sigma_{\mathfrak{C}}(0)=\Sigma_{\mathfrak{B}}(0)$, let γ_0 be the identity mapping.
Let $j=1$. In this case

$\Sigma_{\mathfrak{C}}(1)=\{(b_1, u_0, \sigma_1(1, \ldots, v^1(\sigma_1))[\{1, \ldots, v^1(\sigma_1)\}, \bar{\varphi}_1], \bar{\varrho}_1, \overline{W}_1, \bar{\tau}_1)|$

i) $(u_0, \sigma_1)\in V$ such that σ_1 has the form $(b_1, u_0, u_1[S_{u_1}, \varphi_1], \varrho_1, W_1, \tau_1)$.
ii) $\gamma_0(u_0)=u_0$, the production

$$b_1 u_0 \to \sigma_1(1, \ldots, v^1(\sigma_1))[\{1, \ldots, v^1(\sigma_1)\}, \beta_1]\in\Sigma_{\mathfrak{A}_1},$$

where the mapping β_1: $\{1, \ldots, v^1(\sigma_1)\}\to A_1\{1, \ldots, v(u_0)\}$ is defined as follows:
Let

$$\xi_0\colon \{1, \ldots, v(u_0)\}\to\{1, \ldots, v(u_0)\}, \quad \xi_1\colon S_{u_1}\to\{1, \ldots, |S_{u_1}|\}.$$

For each

$t_1\in S_{u_1}$, $\beta_1(\xi_1(t_1))=c_1\xi_0(t_0)$ iff $\varrho_1(t_1)=(t_0, t_1)$ and $\tau_1((t_0, t_1))=c_1 t_0$.

(Thus for each $t_1\in S_{u_1}$, $\beta_1(\xi_1(t_1))=c_1\xi_0(t_0)$ iff $\varphi_1(t_1)=c_1 t_0$.)

iii) $\bar{\varphi}_1=\beta_1$,

iv) $\bar{\varrho}_1\colon \{1, \ldots, v^1(\sigma_1)\}\to\overline{W}_1$;

for every

$\xi_1(t_1)\in\{1, \ldots, v^1(\sigma_1)\}$, if $\beta_1(\xi_1(t_1))=c_1\xi_0(t_0)$ $(c_1\in A_1, t_0\in\{1, \ldots, v(u_0)\})$ then

$\bar{\varrho}_1(\xi_1(t_1))=(\xi_0(t_0), \xi_1(t_1))$.

v) $\bar{\tau}_1\colon \overline{W}_1\to A_1\{1, \ldots, v(u_0)\}$;

for every

$(\xi_0(t_0), \xi_1(t_1))\in\overline{W}_1$, if $\beta_1(\xi_1(t_1))=c_1\xi_0(t_0)$ $(c_1\in A_1, t_0\in\{1, \ldots, v(u_0)\})$ then

$\bar{\tau}_1((\xi_0(t_0), \xi_1(t_1)))=c_1\xi_0(t_0)$.}

It can be seen that

$\overline{V}_1=\{(\gamma_0(\sigma_0), \bar{\sigma}_1)|\sigma_0=u_0\in\Sigma_{\mathfrak{B}}(0), \bar{\sigma}_1\in\Sigma_{\mathfrak{C}}(1)\}$ has the form

$$(b_1, u_0, \sigma_1(1, \ldots, v^1(\sigma_1))[\{1, \ldots, v^1(\sigma_1)\}, \bar{\varphi}_1], \bar{\varrho}_1, \overline{W}_1, \bar{\tau}_1)$$

and $\bar{\sigma}_1$ is generated by the production

$$b_1 u_0 \to \sigma_1(1, \ldots, v^1(\sigma_1))[\{1, \ldots, v^1(\sigma_1)\}, \bar{\varphi}_1]\in\Sigma_{\mathfrak{B}}(1).\}$$

We define $\gamma_1\colon \Sigma_{\mathfrak{B}}(1)\to\Sigma_{\mathfrak{C}}(1)$ as follows:
Let $\sigma_1=(b_1, u_0, u_1[S_{u_1}, \varphi_1], \varrho_1, W_1, \tau_1)\in\Sigma_{\mathfrak{B}}(1)$, then by the construction of $\mathfrak{A}_1$ and $\mathfrak{C}$ there is a unique production $b_1 u_0\to\sigma_1(1, \ldots, v^1(\sigma_1))[\{1, \ldots, v^1(\sigma_1)\}, \beta_1]\in\Sigma_{\mathfrak{A}_1}$ which generates a unique $\bar{\sigma}_1\in\Sigma_{\mathfrak{C}}(1)$. We define $\gamma_1(\sigma_1)$ to be $\bar{\sigma}_1$. One can see by the definition of $\Sigma_{\mathfrak{C}}(1)$ that γ_1 is onto, hence γ_1 is bijective.

It is routine work to check according to the construction of $\mathfrak{A}_1$ and $\Sigma_{\mathfrak{C}}(1)$ that γ_0, γ_1 satisfy condition (1) and that γ_1 satisfies condition (2).

Let j be an index between 2 and $k-1$. We can assume that $\Sigma_{\mathfrak{C}}(0)$, $\Sigma_{\mathfrak{C}}(1)$, ..., $\Sigma_{\mathfrak{C}}(j-1)$ and $\gamma_0, \ldots, \gamma_{j-1}$ are defined such that $\gamma_0, \ldots, \gamma_{j-1}$ satisfy condition (1) and that γ_{j-1} satisfies condition (2).

We know that $\Sigma_{\mathfrak{C}}(j)$ is the set

$$\Sigma_{\mathfrak{C}}(j) = \{(b_j \dots b_1, u_0, \sigma_j(1, \dots, v^j(\sigma_j))[\{1, \dots, v^j(\sigma_j)\}, \bar{\varphi}_j], \bar{\varrho}_j, \overline{W}_j, \bar{\tau}_j) |$$

i) There is an element $(\sigma_0, \dots, \sigma_{j-1}, \sigma_j) \in V$ such that $\sigma_0 = u_0$, σ_{j-1} has the form

$$(b_{j-1} \dots b_1, u_0, u_{j-1}[S_{u_{j-1}}, \varphi_{j-1}], \varrho_{j-1}, W_{j-1}, \tau_{j-1}),$$

σ_j has the form $(b_j \dots b_1, u_0, u_j[S_{u_j}, \varphi_j], \varrho_j, W_j, \tau_j)$. There is a mapping ε_j: $S_{u_j} \to A_j[W_{j-1}]_{j-1}$ such that conditions i)—iv) in part (5).b of Definition 3.1 hold.

ii) $\gamma_{j-1}(\sigma_{j-1}) = (b_{j-1} \dots b_1, u_0, \sigma_{j-1}(1, \dots, v^{j-1}(\sigma_{j-1}))[\{1, \dots, v^{j-1}(\sigma_{j-1})\}, \bar{\varphi}_{j-1}]$, $\bar{\varrho}_{j-1}, \overline{W}_{j-1}, \bar{\tau}_{j-1}) \in \Sigma_{\mathfrak{C}}(j-1)$

and the production $b_j \sigma_{j-1} \to \sigma_j(1, \dots, v^j(\sigma_j))[\{1, \dots, v^j(\sigma_j)\}, \beta_j]$ is in $\Sigma_{\mathfrak{A}_j}$, where the mapping

$$\beta_j \colon \{1, \dots, v^j(\sigma_j)\} \to A_j\{1, \dots, v^{j-1}(\sigma_{j-1})\}$$

is defined as follows:

Let ξ_{j-1}: $S_{u_{j-1}} \to \{1, \dots, |S_{u_{j-1}}|\}$, ξ_j: $S_{u_j} \to \{1, \dots, |S_{u_j}|\}$. For every $t_j \in S_{u_j}$, $\beta_j(t_j) = c_j \xi_{j-1}(t_{j-1})$ iff $\varrho_j(t_j) = (t_0, \dots, t_{j-1}, t_j)$ and $\tau_j((t_0, \dots, t_{j-1}, t_j)) = c_j \dots c_1 t_0$. (Thus for each $t_j \in S_{u_j}$, $\beta_j(\xi_j(t_j)) = c_j \xi_{j-1}(t_{j-1})$ iff $\varepsilon_j(t_j) = c_j t_{j-1}$.)

iii) $\overline{W}_j = \{(\bar{t}_0, \dots, \bar{t}_{j-2}, \xi_{j-1}(t_{j-1}), \xi_j(t_j)) | \beta_j(\xi_j(t_j)) = c_j \xi_{j-1}(t_{j-1}),$
$\bar{\varrho}_{j-1}(\xi_{j-1}(t_{j-1})) = (\bar{t}_0, \dots, \bar{t}_{j-2}, \xi_{j-1}(t_{j-1}))\} \cup$
$\cup \{(\bar{t}_0, \dots, \bar{t}_{j-1}) \in W_{j-1} |$ there are no $\bar{t}_j$ in $\{1, \dots, v^j(\sigma_j)\}$ and
$c_j \in A_j$ such that $\beta_j(\bar{t}_j) = c_j \bar{t}_{j-1}\} \cup$
$\cup \{(\bar{t}_0, \dots, \bar{t}_l) \in W_{j-1} | 1 \leqq l \leqq j-2\}$.

iv) $\bar{\varrho}_j$: $\{1, \dots, v^j(\sigma_j)\} \to \overline{W}_j$ satisfies the following requirement: for every
$t_j \in S_{u_j}$ if $\beta_j(\xi_j(t_j)) = c_j \xi_{j-1}(t_{j-1})$ and
$\bar{\varrho}_{j-1}(\xi_{j-1}(t_{j-1})) = (\xi_0(t_0), \dots, \xi_{j-1}(t_{j-1}))$ then
$(\xi_0(t_0), \dots, \xi_{j-1}(t_{j-1}), \xi_j(t_j)) \in \overline{W}_j$ and
$\bar{\varrho}_j(\xi_j(t_j)) = (\xi_0(t_0), \dots, \xi_{j-1}(t_{j-1}), \xi_j(t_j))$.

v) For $\bar{\tau}_j$: $\overline{W}_j \to A_j \dots A_1\{1, \dots, v(u_0)\}$,
$\bar{\tau}_j|_{W_j \cap W_{j-1}} = \bar{\tau}_{j-1}|_{W_j \cap W_{j-1}}$ and if
$(t_0, \dots, t_{j-2}, \xi_{j-1}(t_{j-1}), \xi_j(t_j)) \in \overline{W}_j$,
$\beta_j(\xi_j(t_j)) = c_j \xi_{j-1}(t_{j-1})$ and $\bar{\varrho}_{j-1}(\xi_{j-1}(t_{j-1})) = (\bar{t}_0, \dots, \bar{t}_{j-2}, \xi_{j-1}(t_{j-1}))$
then $\bar{\tau}_j((\bar{t}_0, \dots, \bar{t}_{j-2}, \xi_{j-1}(t_{j-1}), \xi_j(t_j))) = c_j \bar{\tau}_{j-1}((\bar{t}_0, \dots, \bar{t}_{j-2}, \xi_{j-1}(t_{j-1})))$.

vi) $\bar{\varphi}_j = \bar{\varrho}_j \circ \bar{\tau}_j$.}

It can be seen that

$\bar{V}_j = \{(\gamma_0(\sigma_0), \ldots, \gamma_{j-1}(\sigma_{j-1}), \bar{\sigma}_j) |$ for $i = 0, \ldots, j-1$

$\sigma_i \in \Sigma_{\mathfrak{B}}(i), (\gamma_0(\sigma_0), \ldots, \gamma_{j-1}(\sigma_{j-1})) \in V_{j-1}$,

σ_0 has the form u_0, and σ_{j-1} has the form $(b_{j-1} \ldots b_1, u_0, u_{j-1}[S_{u_{j-1}}, \varphi_{j-1}], \varrho_{j-1}, W_{j-1}, \tau_{j-1})$. There is an element

$$\sigma_j = (b_j \ldots b_1, u_0, u_j[S_{u_j}, \varphi_j], \varrho_j, W_j, \tau_j) \in \Sigma_{\mathfrak{B}}(j)$$

such that $(\sigma_0, \ldots, \sigma_{j-1}, \sigma_j) \in V_j$, and there is a mapping ε_j: $S_{u_j} \to A_j[W_{j-1}]_{j-1}$ such that conditions i)—iv) in part (5).b of Definition 3.1. hold.

$$\bar{\sigma}_j = (b_j, \ldots, b_1, u_0, \sigma_j(1, \ldots, \nu^j(\sigma_j))[\{1, \ldots, \nu^j(\sigma^j)\}, \bar{\varphi}_j], \bar{\varrho}_j, \overline{W}_j, \bar{\tau}_j)$$

satisfies the requirements ii)—vi) of $\Sigma_{\mathfrak{C}}(j)$.}

We define γ_j: $\Sigma_{\mathfrak{B}}(j) \to \Sigma_{\mathfrak{C}}(j)$ as follows: Let us consider the set

$\Gamma_j = \{\gamma$: $\Sigma_{\mathfrak{B}}(j) \to \Sigma_{\mathfrak{C}}(j) |$ for each $\sigma_j \in \Sigma_{\mathfrak{B}}(j), \gamma(\sigma_j) = \bar{\sigma}_j$

has the form

$$(b_j \ldots b_1, u_0, \sigma_j(1, \ldots, \nu^j(\sigma_j))[\{1, \ldots, \nu^j(\sigma^j)\}, \bar{\varphi}_j], \bar{\varrho}_j, \overline{W}_j, \bar{\tau}_j)$$

and there is a vector $(\sigma_0, \ldots, \sigma_{j-1}, \sigma_j) \in V_j$ such that σ_0 has the form u_0, σ_{j-1} has the form $(b_{j-1} \ldots b_1, u_0, u_{j-1}[S_{u_{j-1}}, \varphi_{j-1}], \varrho_{j-1}, W_{j-1}, \tau_{j-1})$, σ_j has the form $(b_j \ldots b_1, u_0, u_j[S_{u_j}, \varphi_j], \varrho_j, W_j, \tau_j)$, and there is a mapping ε_j: $S_{u_j} \to A_j[W_{j-1}]_{j-1}$ such that conditions i)—iv) in part (5).b of Definition 3.1 hold, and $\bar{\sigma}_j$ satisfies the requirements ii)—vi) of $\Sigma_{\mathfrak{C}}(j)$.}

One can see that if $\bar{\gamma}_j \in \Gamma_j$ then $\bar{\gamma}_j$ is injective and $\bar{\gamma}_j$ satisfies condition (2) because of the construction of $\mathfrak{A}_j$ and $\Sigma_{\mathfrak{C}}(j)$. Using this fact one can see that $|\Gamma_j| = 1$. Let γ_j be the only element of Γ_j. One can see that γ_j is bijective, and $\gamma_0, \ldots, \gamma_{j-1}, \gamma_j$ satisfy condition (1).

Let $j = k$. We can assume that $\Sigma_{\mathfrak{C}}(0), \Sigma_{\mathfrak{C}}(1), \ldots, \Sigma_{\mathfrak{C}}(k-1)$ and $\gamma_0, \ldots, \gamma_{k-1}$ are defined such that $\gamma_0, \ldots, \gamma_{k-1}$ satisfy condition (1) and that γ_{k-1} satisfies condition (2).

We know that $\Sigma_{\mathfrak{C}}(k)$ is the set

$\Sigma_{\mathfrak{C}}(k) = \{(b_k \ldots b_1, u_0, u_k[S_{u_k}, \varphi_k], \bar{\varrho}_k, \overline{W}_k, \bar{\tau}_k) |$

i) there is an element $(\sigma_0, \ldots, \sigma_{k-1}, \sigma_k) \in V_k$ such that $\sigma_0 = u_0$, σ_{k-1} has the form $(b_{k-1} \ldots b_1, u_0, u_{k-1}[S_{u_{k-1}}, \varphi_{k-1}], \varrho_{k-1}, W_{k-1}, \tau_{k-1})$, σ_k has the form $(b_k \ldots b_1, u_0, u_k[S_{u_k}, \varphi_k], \varrho_k, W_k, \tau_k)$. There is a mapping ε_k: $S_{u_k} \to A_k[W_{k-1}]_{k-1}$ such that conditions i)—iv) in part (5).b of Definition 3.1 hold.

ii) $\gamma_{k-1}(\sigma_{k-1}) = (b_{k-1} \ldots b_1, u_0, \sigma_{k-1}(1, \ldots, \nu^{k-1}(\sigma_{k-1}))[\{1, \ldots, \nu^{k-1}(\sigma_{k-1})\}, \bar{\varphi}_{k-1}]$, $\bar{\varrho}_{k-1}, \overline{W}_{k-1}, \bar{\tau}_{k-1}) \in \Sigma_{\mathfrak{B}}(k-1)$

and the production

$b_k\sigma_{k-1} \to u_k\ [S_{u_k}, \beta_k]$ is in $\Sigma_{\mathfrak{B}_k}$,

where the mapping

$\beta_k\colon S_{u_k} \to A_k\{1, \ldots, \nu^{k-1}(\sigma_{k-1})\}$

is defined as follows: Let

$\xi_{k-1}\colon S_{u_{k-1}} \to \{1, \ldots, |S_{u_{k-1}}|\}$, $\xi_k\colon S_{u_k} \to S_{u_k}$.

For every

$t_k \in S_{u_k}$, $\beta_k(t_k) = c_k\xi_{k-1}(t_{k-1})$ iff $\varrho_k(t_k) = (t_0, \ldots, t_{k-1}, t_k)$ and

$\tau_k((t_0, \ldots, t_{k-1}, t_k)) = c_k \ldots c_1 t_0$.

(Thus for each $t_k \in S_{u_k}$, $\beta_k(\xi_k(t_k)) = c_k\xi_{k-1}(t_{k-1})$ iff $\varepsilon_k(t_k) = c_k t_{k-1}$.)

iii) $\overline{W}_k = \{(\bar{t}_0, \ldots, \bar{t}_{k-2}, \xi_{k-1}(t_{k-1}), \xi_k(t_k)) | \beta_k(\xi_k(t_k)) = c_k\xi_{k-1}(t_{k-1}),$

$\bar{\varrho}_{k-1}(\xi_{k-1}(t_{k-1})) = (\bar{t}_0, \ldots, \bar{t}_{k-2}, \xi_{k-1}(t_{k-1}))\} \cup$

$\cup \{(\bar{t}_0, \ldots, \bar{t}_{k-2}, \bar{t}_{k-1}) \in W_{k-1} |$

there are no $t_k \in S_{u_k}$ and $c_k \in A_k$ such that $\beta_k(\bar{t}_k) = c_k\bar{t}_{k-1}\} \cup$

$\cup \{(\bar{t}_0, \ldots, \bar{t}_{l-1}, \bar{t}_l) \in W_{k-1} | 1 \leqq l \leqq k-2\}$.

iv) $\bar{\varrho}_k\colon S_{u_k} \to \overline{W}_k$ satisfies the following requirement: for every

$t_k \in S_{u_k}$, if $\beta_k(\xi_k(t_k)) = c_k\xi_{k-1}(t_{k-1})$ and

$\bar{\varrho}_{k-1}(\xi_{k-1}(t_{k-1})) = (\bar{t}_0, \ldots, \bar{t}_{k-2}, \xi_{k-1}(t_{k-1}))$, then

$(\bar{t}_0, \ldots, \bar{t}_{k-2}, \xi_{k-1}(t_{k-1}), \xi_k(t_k)) \in W_k$ and

$\bar{\varrho}_k(\xi_k(t_k)) = (\bar{t}_0, \ldots, \bar{t}_{k-2}, \xi_{k-1}(t_{k-1}), \xi_k(t_k))$.

v) For

$\bar{\tau}_k\colon \overline{W}_k \to (A_k \ldots A_2A_1 \cup \ldots \cup A_2A_1 \cup A_1)\{1, \ldots, \nu(u_0)\}$, $\bar{\tau}_k|_{W_k \cap W_{k-1}} = \bar{\tau}_{k-1}|_{W_k \cap W_{k-1}}$

and if

$(\bar{t}_0, \ldots, \bar{t}_{k-2}, \xi_{k-1}(t_{k-1}), \xi_k(t_k)) \in \overline{W}_k$, $\beta_k(\xi_k(t_k)) = c_k\xi_{k-1}(t_{k-1})$, and

$\bar{\varrho}_{k-1}(\xi_{k-1}(t_{k-1})) = (\bar{t}_0, \ldots, \bar{t}_{k-2}, \xi_{k-1}(t_{k-1}))$ then

$\bar{\tau}_k((\bar{t}_0, \ldots, \bar{t}_{k-2}, \xi_{k-1}(t_{k-1}), \xi_k(t_k)) = c_k\tau_{k-1}((\bar{t}_0, \ldots, \bar{t}_{k-2}, \xi_{k-1}(t_{k-1})))$.

vi) $\varphi_k = \bar{\varrho}_k \circ \bar{\tau}_k$.

It can be seen that

$\overline{V}_k = \{(\gamma_0(\sigma_0), \ldots, \gamma_{k-1}(\sigma_{k-1}), \bar{\sigma}_k) |$ for $i = 0, \ldots, k-1$, $\sigma_i \in \Sigma_{\mathfrak{B}}(i)$,

$(\gamma_0(\sigma_0), \ldots, \gamma_{k-1}(\sigma_{k-1})) \in V_{k-1}$, σ_0 has the form $u_0\}$.

and σ_{k-1} has the form $(b_{k-1}\dots b_1, u_0, u_{k-1}[S_{u_{k-1}}, \varphi_{k-1}], \varrho_{k-1}, W_{k-1}, \tau_{k-1})$. There is an element $\sigma_k=(b_k\dots b_1, u_0, u_k[S_{u_k}, \varphi_k], \varrho_k, W_k, \tau_k)\in\Sigma_{\mathfrak{B}}(k)$ such that $(\sigma_0, \dots, \sigma_k)\in V_k$, and there is a mapping $\varepsilon_k\colon S_{u_k}\to A_k[W_{k-1}]_{k-1}$ such that conditions i)—iv) in part (5).b of Definition 3.1 hold, and $\bar\sigma_k=(b_k\dots b_1, u_0, u_k[S_{u_k}, \bar\varphi_k], \bar\varrho_k, \overline{W}_k, \bar\tau_k)$ satisfies the requirements ii)—vi) of $\Sigma_{\mathfrak{C}}(k)$.}

We define $\gamma_k\colon \Sigma_{\mathfrak{B}}(k)\to\Sigma_{\mathfrak{C}}(k)$ as follows: Let us consider the set

$$\Gamma_k=\{\gamma\colon \Sigma_{\mathfrak{B}}(k)\to\Sigma_{\mathfrak{C}}(k)\mid \text{for each } \sigma_k\in\Sigma_{\mathfrak{B}}(k),\ \gamma(\sigma_k)=\bar\sigma_k$$

has the form $(b_k\dots b_1, u_0, u_k[S_{u_k}, \varphi_k], \bar\varrho_k, \overline{W}_k, \bar\tau_k)$ and there is a vector $(\sigma_0, \dots, \sigma_{k-1}, \sigma_k)\in V_k$ such that σ_0 has the form u_0, σ_{k-1} has the form

$$(b_{k-1}\dots b_1, u_0, u_{k-1}[S_{u_{k-1}}, \varphi_{k-1}], \varrho_{k-1}, W_{k-1}, \tau_{k-1}),$$

σ_k has the form $(b_k\cdots b_1, u_0, u_k[S_{u_k}, \varphi_k], \varrho_k, W_k, \tau_k)$ and there is a mapping $\varepsilon_k\colon S_{u_k}\to A_k[W_{k-1}]_{k-1}$ such that conditions i)—iv) in part (5).b of Definition 3.1 hold, and $\bar\sigma_k$ satisfies the requirements ii)—vi) of $\Sigma_{\mathfrak{C}}(k)\}$.

One can see that if $\bar\gamma_k\in\Gamma_k$ then $\bar\gamma_k$ satisfies condition (2) because of the constructions of $\mathfrak{A}_k$ and $\Sigma_{\mathfrak{C}}(k)$. Using this fact one can see that $|\Gamma_k|=1$. Let γ_k be the only element of Γ_k. One can see that γ_k is surjective. Using the fact that γ_k satisfies condition (2), one can easily prove that the mappings $\gamma_0, \dots, \gamma_k$ satisfy condition (1).

Finally we shall prove, using the fact that for $j=0, \dots, k$ the mappings $\gamma_0, \dots, \gamma_j$ satisfy condition (1) and for $j=0, \dots, k$ the mapping γ_j satisfies condition (2), that $\tau_{\mathfrak{B}}=\tau_{\mathfrak{C}}$.

Assume that $K_0=(e[\{e\}, \psi_0\colon e\to bp], \Theta^0, Z^0, \Omega^0)$ is a starting configuration of $\mathfrak{B}$ and that for a configuration $K_1=(q_1[S_{q_1}, \psi^1], \Theta^1, Z^1, \Omega^1)$, $K_0\Rightarrow_{\mathfrak{B}}^* K_1$ holds. Then K_0 is a starting configuration of $\mathfrak{C}$ as well. We shall show that there is a configuration

$$\overline{K}_1=(q^1[S_{q^1}, \psi^1], \bar\Theta^1, \bar Z^1, \bar\Omega^1)$$

of $\mathfrak{C}$ with bijective correspondences

$$(*)\qquad \begin{aligned}&\alpha_0\colon [Z^1]_0\to[\bar Z^1]_0\\ &\quad\vdots\\ &\alpha_k\colon [Z^1]_k\to[\bar Z^1]_k\end{aligned}$$

such that α_0 and α_k is the identity function and $K_0\Rightarrow_{\mathfrak{C}}^*\overline{K}_1$ holds, moreover

i) for every $s_k\in S_{q^1}$, $\Theta^1(s_k)=(s_0, s_1, \dots, s_k)$ iff $\bar\Theta^1(\alpha_k(s_k))=(\alpha_0(s_0), \alpha_1(s_1), \dots, \alpha_k(s_k))$ and

ii) $(s_0, s_1, \dots, s_j)\in Z^1$ iff $(\alpha_0(s_0), \alpha_1(s_1), \dots, \alpha_j(s_j)\in\bar Z^1$ $(1\leqq j\leqq k, (s_0, s_1, \dots, s_j)\in(N^*)^j)$ and

iii) for every

$$(s_0, s_1, \dots, s_j)\in Z^1\ (1\leqq j\leqq k),\ \Omega^1((s_0, s_1, \dots, s_j))=\bar\Omega^1(\alpha_0(s_0), \alpha_1(s_1), \dots, \alpha_j(s_j))).$$

Conversely, if $K_0\Rightarrow_{\mathfrak{C}}^*\overline{K}_1$ holds then there is a configuration K_1 of $\mathfrak{B}$ and there are bijective functions $(*)$ such that α_0 and α_k are identity functions and i), ii), iii) hold. Hence if K_1 is final then $\overline{K}_1$ is final and vice versa, thus $\tau_{\mathfrak{B}}=\tau_{\mathfrak{C}}$ follows.

First we shall prove the first part of the statement, the second part can be proved similarly. We prove by induction on the length of the transition $K_0 \Rightarrow_{\mathfrak{B}}^* K_1$.

a) The length of $K_0 \Rightarrow_{\mathfrak{B}}^* K_1$ is zero, $(K_1 = K_0)$. Trivial.

b) Assume that the statement is true for $K_0 \Rightarrow_{\mathfrak{B}}^* K_1$, $K_0 \Rightarrow_{\mathfrak{C}}^* \bar{K}_1$ and for the functions $(*)$ and that $K_1 \Rightarrow_{\mathfrak{B}} K_2 = (q^2[S_{q^2}, \psi^2], \Theta^2, Z^2, \Omega^2)$ holds.

By the definition of the relation $\Rightarrow_{\mathfrak{B}}$, there are mappings $\varkappa_i$: $[Z^1]_i \to \Sigma_{\mathfrak{B}}(i)$ for $i = 0, 1, \ldots, k$ such that for every $(s_0, s_1, \ldots, s_j) \in Z^1$ $(1 \leqq j \leqq k)$ if

$$\Omega^1((s_0, s_1, \ldots, s_j)) = b_j \ldots b_1 u_0(1, \ldots, \nu(u_0))[\{1, \ldots, \nu(u_0)\}, \vartheta_0]$$

$$(b_j \in A_j, \ldots, b_1 \in A_1, u_0 \in G_0 \cup Y_0, \vartheta_0\colon \{1, \ldots, \nu(u_0)\} \to T_{G_0}(Y_0))$$

then $\varkappa_0(s_0) = u_0$, $\varkappa_i(s_i)$ has the form $(b_i \ldots b_1, u_0, u_i[S_{u_i}, \varphi_i], \varrho_i, W_i, \tau_i)$ for $i = 1, \ldots, j$, moreover $(\varkappa_0(s_0), \varkappa_1(s_1), \ldots, \varkappa_j(s_j)) \in V_j$. Take the mappings $\bar{\varkappa}_i$: $[\bar{Z}^1]_i \to \Sigma_{\mathfrak{C}}(i)$ for $i = 0, \ldots, k$ defined by $\bar{\varkappa}_i(\alpha_i(s_i)) = \gamma_i(\varkappa_i(s_i))$ for each $s_i \in [Z^1]_i$. Notice, that $\bar{\varkappa}_i$ is well defined, because α_i and γ_i are bijective. By the induction hypothesis for each $(s_0, s_1, \ldots, s_j) \in Z^1$ $(1 \leqq j \leqq k)$, $\Omega^1((s_0, s_1, \ldots, s_j)) = \bar{\Omega}_1((\alpha_0(s_0), \alpha_1(s_1), \ldots, \alpha_j(s_j)))$. Since $\varkappa_0 = \bar{\varkappa}_0$ and for each $s_i \in [Z^1]_i$ the first two components of $\varkappa_i(s_i)$ are equal to the first two components of $\bar{\varkappa}_i(\alpha_i(s_i))$ for $i = 1, \ldots, k$, moreover for every $(\sigma_0, \sigma_1, \ldots, \sigma_j) \in V_j$ $(1 \leqq j \leqq k)$, $(\gamma_0(\sigma_0), \gamma_1(\sigma_1), \ldots, \gamma_j(\sigma_j)) \in \bar{V}_j$ it follows that the mappings $\bar{\varkappa}_i$ $(i = 0, 1, \ldots, k)$ satisfy condition (1) in Definition 3.3. The mappings $\bar{\varkappa}_i$ $(i = 0, 1, \ldots, k)$ uniquely determine a configuration $\bar{K}_2 = (\bar{q}^2[S_{\bar{q}^2}, \bar{\psi}^2], \bar{\Theta}^2, \bar{Z}^2, \bar{\Omega}^2)$ of $\mathfrak{C}$ such that $\bar{K}_1 \Rightarrow_{\mathfrak{C}} \bar{K}_2$ holds. First we show that $q^2[S_{q^2}, \psi^2] = \bar{q}^2[S_{\bar{q}^2}, \bar{\psi}^2]$. By the transition $K_1 \Rightarrow_{\mathfrak{B}} K_2$ we know that $q^2 = q^1[S_{q^1}, \delta]$, where δ: $S_{q^1} \to T_{G_k}(Y_k \cup N^*)$ satisfies the following formula: for each $s_k \in S_{q^1}$ if $\varkappa_k(s_k) = (b_k \ldots b_1 u_0, u_k[S_{u_k}, \varphi_k], \varrho_k, W_k, \tau_k)$ then $\delta(s_k) = \omega_{s_k}(u_k)$. By the induction hypothesis and the transition $\bar{K}_1 \Rightarrow_{\mathfrak{C}} \bar{K}_2$ we obtain that $\bar{q}^2 = q^1[S_{q^1}, \bar{\delta}]$, where $\bar{\delta}$: $S_{q^1} \to T_{G_k}(Y_k \cup N^*)$ satisfies the following formula: for every $s_k \in S_{q^1}$ if $\varkappa_k(s_k) = (b_k \ldots b_1, u_0, u_k[S_{u_k}, \varphi_k], \varrho_k, W_k, \tau_k)$ and $\delta(s_k) = \omega_{s_k}(u_k)$ then $\bar{\varkappa}_k(\alpha_k(s_k)) = \bar{\varkappa}_k(s_k) = \gamma_k(\varkappa_k(s_k)) = (b_k \ldots b_1, u_0, u_k[S_{u_k}, \varphi_k], \bar{\varrho}_k, \bar{W}_k, \bar{\tau}_k)$ for some $\bar{\varrho}_k, \bar{W}_k$ and $\bar{\tau}_k$, moreover $\bar{\delta}(s_k) = \omega_{s_k}(u_k) = \delta(s_k)$, thus $q^2 = \bar{q}^2$.

Again by the transition $K_1 \Rightarrow_{\mathfrak{B}} K_2$ and $\bar{K}_1 \Rightarrow_{\mathfrak{C}} \bar{K}_2$ we have that ψ^2 and $\bar{\psi}^2$: $S_{q^2} \to A_k \ldots A_1 T_{G_0}(Y_0)$ satisfy the following conditions:

Let $\bar{s}_k \in S_{q^2}$ be arbitrary and consider its unique decomposition $\bar{s}_k = s_k t_k$, where $s_k \in S_{q^1}$, $\delta(s_k) = \omega_{s_k}(u_k)$ for some $u_k (\in P_{G_k}(Y_k))$, $t_k \in S_{u_k}$ and $\varkappa_k(s_k)$ has the form $\varkappa_k(s_k) = (b_k \ldots b_1, u_0, u_k[S_{u_k}, \varphi_k], \varrho_k, W_k, \tau_k)$. Then if $\varphi_k(t_k) = c_k \ldots c_1 t_0$, $(c_k \ldots c_1 \in A_k \ldots A_1$, $t_0 \in \{1, \ldots, \nu(u_0)\})$ and $\psi^1(s_k) = u_0(1, \ldots, \nu(u_0))[\{1, \ldots, \nu(u_0)\}, \vartheta_0]$, $(u_0 \in G_0 \cup Y_0, \vartheta_0\colon \{1, \ldots, \nu(u_0)\} \to T_{G_0}(Y_0))$ then $\psi^2(s_k t_k) = c_k \ldots c_1 \vartheta_0(t_0)$.

We know that $\bar{s}_k$ has the same decomposition using $\bar{\delta} = \delta$ and $\bar{\varkappa}_k$, because $\bar{\varkappa}_k(s_k)$ has the form $(b_k \ldots b_1, u_0, u_k[S_{u_k}, \varphi_k], \bar{\varrho}_k, \bar{W}_k, \bar{\tau}_k)$. Since $\varphi_k(t_k) = c_k \cdots c_1 t_0$ and $\psi^1(s_k) = u_0(1, \ldots, \nu(u_0))[\{1, \ldots, \nu(u_0)\}, \vartheta_0]$ thus $\bar{\psi}^2(s_k t_k) = c_k \ldots c_1 \vartheta_0(t_0)$. We have obtained that $\psi^2 = \bar{\psi}^2$.

$$Z^2 = \{(s_0 t_0, \ldots, s_j t_j) \mid (s_0, \ldots, s_l) \in Z^1, \quad j \leqq l \leqq k,$$

$$\varkappa_l(s_l) = (b_l \ldots b_1, u_0, u_l[S_{u_l}, \varphi_l], \varrho_l, W_l, \tau_l) \quad \text{and} \quad (t_0, t_1, \ldots, t_j) \in W_l\}.$$

$$\bar{Z}^2 = \{(\alpha_0(s_0) \bar{t}_0, \ldots, \alpha_j(s_j) \bar{t}_j) \mid (\alpha_0(s_0), \ldots, \alpha_l(s_l)) \in \bar{Z}^1, \quad j \leqq l \leqq k,$$

and $(\bar{t}_0, \ldots, \bar{t}_j)$ is a member of the fifth component of $\bar{\varkappa}_l(\alpha_l(s_l))$.$\}$

Now we define the mappings α_i^2: $[Z^2]_i \to [\bar{Z}^2]_i$, $(i=0, \ldots, k)$ as follows: Let α_0^2 and α_k^2 be identity mappings, and for $i=1, \ldots, k-1$ take an arbitrary element $s_i t_i \in [Z^2]_i$, where $\varkappa_i(s_i)=(b_i \ldots b_1, u_0, u_i[S_{u_i}, \varphi_i], \varrho_i, W_i, \tau_i)$, then we define $\alpha_i^2(s_i t_i)$ to be $\alpha_i(s_i)\xi_i(t_i)$, where ξ_i: $S_{u_i} \to \{1, \ldots, |S_{u_i}|\}$.

We have to show that for $i=1, \ldots, k-1$ α_i^2 is a bijective function. Let $s_i t_i = \bar{s}_i \bar{t}_i (\in Z_i^2)$ and assume that $s_i \neq \bar{s}_i$. Then one of s_i or $\bar{s}_i$ is a proper initial segment of the other one, which contradicts the definition of the configuration, thus α_i^2 is a well defined function.

Assume that $\alpha_i^2(s_i t_i)=\alpha_i^2(\bar{s}_i \bar{t}_i)$ such that $s_i \neq \bar{s}_i$ or $t_i \neq \bar{t}_i$. If $s_i \neq \bar{s}_i$ then $\alpha_i(s_i) \neq \alpha_i(\bar{s}_i)$ and ξ_i is a function whose range is N thus $\alpha_i(s_i)\xi_i(t_i) \neq \alpha_i(\bar{s}_i)\xi_i(\bar{t}_i)$. If $s_i=\bar{s}_i$ and $t_i \neq \bar{t}_i$ then $\alpha_i^2(s_i t_i)=\alpha_i(s_i)\xi_i(t_i) \neq \alpha_i(s_i)\xi_i(\bar{t}_i)=\alpha^2(\bar{s}_i \bar{t}_i)$ since $\xi_i(t_i) \neq \xi_i(\bar{t}_i)$ thus we obtained that α_i^2 is injective.

Let $\bar{s}_i \bar{t}_i \in [Z^2]_i$, then there is an element $(\bar{s}_0 \bar{t}_0, \ldots, \bar{s}_i \bar{t}_i, \ldots, \bar{s}_j \bar{t}_j) \in Z^2$, where $j \geqq i$. By the construction of Z^2, $(\bar{s}_0, \ldots, \bar{s}_i, \ldots, \bar{s}_j, \ldots, \bar{s}_l) \in Z^1$ for some $\bar{s}_{j+1}, \ldots, \bar{s}_l (\in N^*)$, $1 \leqq j \leqq l \leqq k$, and

$$\bar{\varkappa}_l(\bar{s}_l) = \begin{cases} (b_l \ldots b_1, \sigma_0, \sigma_l(1, \ldots, v^l(\sigma_l))[\{1, \ldots, v^l(\sigma_l)\}, \bar{\varphi}_l], \bar{\varrho}_l, \overline{W}_l, \bar{\tau}_l) & \text{if } l \leqq k, \\ (b_k \ldots b_1, \sigma_0, u_k[S_{u_k}, \varphi_k], \bar{\varrho}_k, \overline{W}_k, \bar{\tau}_k) & \text{if } l = k, \end{cases}$$

and $(\bar{t}_0, \ldots, \bar{t}_i, \ldots, \bar{t}_j) \in W_l$. By the induction hypothesis there is an element $(s_0, \ldots, s_i, \ldots, s_j, \ldots, s_l)$ of Z^1 such that

$$(\alpha_0(s_0), \ldots, \alpha_i(s_i), \ldots, \alpha_j(s_j), \ldots, \alpha_l(s_l)) = (\bar{s}_0, \ldots, \bar{s}_i, \ldots, \bar{s}_j, \ldots, \bar{s}_l).$$

Since $\bar{\varkappa}_l(s_l)=\gamma_l(\varkappa_l(s_l))$ by definition, we can apply condition (2) (ii) stated for γ_l, which tells us that $(\bar{t}_0, \ldots, \bar{t}_i, \ldots, \bar{t}_j) \in \overline{W}_l$ iff there is a $(\xi_0^{-1}(\bar{t}_0), \ldots, \xi_i^{-1}(\bar{t}_i), \ldots, \xi_j^{-1}(\bar{t}_j)) \in W_l$ for $\xi_0, \ldots, \xi_j$ defined in the condition. Thus

$$(s_0 \xi_0^{-1}(\bar{t}_0), \ldots, s_i \xi_i^{-1}(\bar{t}_i), \ldots, s_j \xi_j^{-1}(\bar{t}_j)) \in Z^2,$$

moreover

$$\alpha_i^2(s_i \xi_i^{-1}(\bar{t}_i)) = \alpha_i(s_i)\xi_i(\xi_i^{-1}(\bar{t}_i)) = \alpha_i(s_i)\bar{t}_i = \bar{s}_i \bar{t}_i,$$

hence α_i^2 is surjective $(i=1, \ldots, k-1)$. Thus we have proved that α_i^2 is bijective $(i=0, \ldots, k)$.

Let $\bar{s}_k \in S_{q^2}$ be arbitrary and consider its unique decomposition $\bar{s}_k = s_k t_k$ where $s_k \in S_{q^1}$, $\varkappa_k(s_k)$ has the form $(b_k \ldots b_1, u_0, u_k[S_{u_k}, \varphi_k], \varrho_k, W_k, \tau_k)$, $\delta(s_k)=\omega_{s_k}(u_k)$, $t_k \in S_{u_k}$. In this case $\bar{\varkappa}_k(s_k)(=\gamma_k(\varkappa_k(s_k)))$ has the form $(b_k \ldots b_1, u_0, u_k[S_{u_k}, \varphi_k], \bar{\varrho}_k, \overline{W}_k, \bar{\tau}_k)$. Using condition (2) (iv) stated for γ_k, $\varrho_k(t_k)=(t_0, t_1, \ldots, t_k)$ iff $\bar{\varrho}_k(t_k) = (\xi_0(t_0), \xi_1(t_1), \ldots, \xi_k(t_k))$ for $\xi_0, \xi_1, \ldots, \xi_k$ defined in the condition.

Using the induction hypothesis $\Theta^1(s_k)=(s_0, s_1, \ldots, s_k)$ iff $\bar{\Theta}^1(s_k)=(\alpha_0(s_0), \alpha_1(s_1), \ldots, \alpha_k(s_k))$. By the definition of Θ^2 and $\bar{\Theta}^2$ we obtain that

$$\Theta^2(\bar{s}_k) = (s_0 t_0, s_1 t_1, \ldots, s_k t_k) \quad \text{iff}$$

$$\bar{\Theta}^2(\bar{s}_k) = (\alpha_0(s_0)\xi_0(t_0), \alpha_1(s_1)\xi_1(t_1), \ldots, \alpha_k(s_k)\xi_k(t_k)) =$$

$$= (\alpha_0^2(s_0 t_0), \alpha_1^2(s_1 t_1), \ldots, \alpha_k^2(s_k t_k)).$$

Thus we have proved that condition i) holds for the mappings $\alpha_0^2, \ldots, \alpha_k^2$.

Let $(s_0 t_0, \ldots, s_j t_j) \in Z^2$ be arbitrary, where $1 \leqq j \leqq k$ and $(s_0, \ldots, s_j, \ldots, s_l) \in Z^1$ for some $s_{j+1}, \ldots, s_l (\in N^*)$, $(j \leqq l \leqq k)$, moreover $\varkappa_l(s_l)=(b_l \ldots b_1, u_0, u_l[S_{u_l}, \varphi_l],$

$\varrho_l, W_l, \tau_l)$ and $(t_0, \ldots, t_j) \in W_l$. By the induction hypothesis, $(\alpha_0(s_0), \ldots, \alpha_l(s_l)) \in \bar{Z}^1$. By the definition of $\bar{\varkappa}_l$, $\bar{\varkappa}_l(\alpha_l(s_l)) = \gamma_l(\varkappa_l(s_l))$, i.e,

$$\bar{\varkappa}_l(\alpha_l(s_l)) = \begin{cases} (b_l \ldots b_1, u_0, \sigma_l(1, \ldots, \nu^l(\sigma_l))[\{1, \ldots, \nu^l(\sigma_l)\}, \bar{\varphi}_l], \bar{\varrho}_l, \overline{W}_l, \bar{\tau}_l) & \text{if } l < k, \\ (b_k \ldots b_1, u_0, u_k[S_{u_k}, \varphi_k], \varrho_k, W_k, \tau_k) & \text{if } l = k. \end{cases}$$

We can apply condition (2) (ii) stated for γ_l which tells us that $(\xi_0(t_0), \ldots, \xi_j(t_j)) \in \overline{W}_l$ iff $(t_0, \ldots, t_j) \in W_l$ for the mappings $\xi_0, \ldots, \xi_j$ defined in the condition. Thus

$$(\alpha_0^2(s_0 t_0), \ldots, \alpha_j^2(s_j t_j)) = (\alpha_0(s_0)\xi_0(t_0), \ldots, \alpha_j(s_j)\xi_j(t_j)) \in Z^2.$$

Conversely, let $(v_0, \ldots, v_j) \in Z^2$ be arbitrary. By the construction of the set Z^2 there are two vectors $(\bar{s}_0, \ldots, \bar{s}_j, \ldots, \bar{s}_l) \in Z^1$ $(1 \leqq j \leqq l \leqq k)$ and $(\bar{t}_0, \ldots, \bar{t}_j) \in ((N^*)^j)$ such that $v_i = \bar{s}_i \bar{t}_i$ for $i = 0, \ldots, j$, and $(\bar{t}_0, \ldots, \bar{t}_j)$ is in the fifth component of $\bar{\varkappa}_l(\bar{s}_l)$. By the induction hypothesis, $(\alpha_0^{-1}(\bar{s}_0), \ldots, \alpha_j^{-1}(\bar{s}_j), \ldots, \alpha_l^{-1}(\bar{s}_l)) \in Z^1$. We know that $\bar{\varkappa}_l(\bar{s}_l) = \gamma_l(\varkappa_l(\alpha_l^{-1}(\bar{s}_l)))$. According to condition (2)(ii) stated for γ_l we obtain that $(\xi_0^{-1}(\bar{t}_0), \ldots, \xi_j^{-1}(\bar{t}_j))$ is in the fifth component of $\varkappa_l(\bar{s}_l)$ for the mappings $\xi_0, \ldots, \xi_j$ defined in the condition. Thus $(\alpha_0^{-1}(\bar{s}_0)\xi_0^{-1}(\bar{t}_0), \ldots, \alpha_j^{-1}(\bar{s}_j)\xi_j^{-1}(\bar{t}_j)) \in Z^2$, moreover

$$\alpha_i^2(\alpha_i^{-1}(\bar{s}_i)\xi_i^{-1}(\bar{t}_i)) = \bar{s}_i \bar{t}_i \quad \text{for} \quad i = 0, \ldots, j.$$

We have proved that condition ii) holds for the mappings $\alpha_0^2, \ldots, \alpha_k^2$.

It has remained to prove that condition iii) holds for $\alpha_0^2, \ldots, \alpha_2^k$. Let $(s_0 t_0, \ldots, s_j t_j) \in Z^2$ be arbitrary, where $1 \leqq j \leqq k$, $(s_0, \ldots, s_j, \ldots, s_l) \in Z^1$ for some $s_{j+1}, \ldots, s_l (\in N^*)$, $j \leqq l \leqq k$, and $\varkappa_l(s_l) = (b_l \ldots b_1, u_0, u_l[S_{u_l}, \varphi_l], \varrho_l, W_l, \tau_l)$ and $(t_0, \ldots, t_j) \in W_l$. We know that $\Omega^1((s_0, \ldots, s_j, \ldots, s_l)) = b_l \ldots b_1 u_0(1, \ldots, \nu(u_0))[\{1, \ldots, \nu(u_0)\}, \vartheta_0]$ for some $\vartheta_0 : \{1, \ldots, \nu(u_0)\} \to T_{G_0}(Y_0)$, and $\tau_l((t_0, \ldots, t_j)) = c_j \ldots c_1 t_0$ for some $c_j \in A_j, \ldots, c_1 \in A_1$, $t_0 \in \{1, \ldots, \nu(u_0)\}$, thus $\Omega^2((s_0 t_0, \ldots, s_j t_j)) = c_j \ldots c_1 \vartheta_0(t_0)$. By the induction hypothesis, $(\alpha_0(s_0), \ldots, \alpha_j(s_j), \ldots, \alpha_l(s_l)) \in Z^1$ and

$$\bar{\varkappa}_l(\alpha_l(s_l)) = \gamma_l(\varkappa_l(s_l)) = \begin{cases} (b_l \ldots b_1, u_0, u_l[S_{u_l}, \varphi_l], \varrho_l, W_l, \tau_l) & \text{if} \quad l = k, \\ (b_l \ldots b_1, u_0, \varkappa_l(s_l)(1, \ldots, \nu^l(\varkappa_l(s_l)))[\{1, \ldots, \nu^l(\varkappa_l(s_l))\}, \bar{\varphi}_l], & \\ \bar{\varrho}_l, \overline{W}_l, \bar{\tau}_l) & \text{if} \quad l < k. \end{cases}$$

We can apply condition 2(v) stated for γ_l which tells us that $\tau_l((t_0, \ldots, t_j)) = c_j \ldots c_1 t_0$ iff $\bar{\tau}_l((\xi_0(t_0), \ldots, \xi_j(t_j))) = c_j \ldots c_1 \xi_0(t_0)$ for the mappings $\xi_0, \ldots, \xi_j$ defined in the condition.

Thus $\bar{\tau}_l((\xi_0(t_0), \ldots, \xi_j(t_j))) = c_j \ldots c_1 \xi_0(t_0)$ holds. By the induction hypothesis, $\Omega^1((\alpha_0(s_0), \ldots, \alpha_j(s_j), \ldots, \alpha_l(s_l))) = b_l \ldots b_1 u_0(1, \ldots, \nu(u_0))[\{1, \ldots, \nu(u_0)\}, \vartheta_0]$. By the definition of $\bar{\Omega}^2$ and α_i^2 $(i = 0, \ldots, k)$,

$$\bar{\Omega}^2((\alpha_0(s_0)\xi_0(t_0), \ldots, \alpha_j(s_j)\xi_j(t_j))) =$$

$$= \bar{\Omega}^2((\alpha_0^2(s_0 t_0), \ldots, \alpha_j^2(s_j t_j))) = c_j \ldots c_1 \vartheta_0(t_0).$$

Thus $\Omega^2((s_0 t_0, \ldots, s_j t_j)) = \bar{\Omega}^2((\alpha_0^2(s_0 t_0), \ldots, \alpha_j^2(s_j t_j)))$ holds. The proof of the first part of the statement is complete. The second part of the statement can be proved by induction on the length of the transition $K_0 \Rightarrow_{\mathfrak{C}}^* \bar{K}_1$.

a) The length of $K_0 \Rightarrow_{\mathfrak{C}}^* \bar{K}_1$ is zero, $(\bar{K}_1 = K_0)$. Trivial.

b) Assume that the statement is true for $K_0 \Rightarrow_{\mathfrak{C}}^* \bar{K}_1$, $K_0 \Rightarrow_{\mathfrak{B}}^* K_1$ and for the functions (*) and that $\bar{K}_1 \Rightarrow_{\mathfrak{C}}^* \bar{K}_2$ holds. By the definition of the relation $\Rightarrow_{\mathfrak{C}}$ there are mappings

$\bar{\varkappa}_i$: $[\bar{Z}^1]_i \to \Sigma_{\mathfrak{B}}(i)$ for $i=0, 1, \ldots, k$ such that for every $(\bar{s}_0, \bar{s}_1, \ldots, \bar{s}_j) \in \bar{Z}^1$ $(1 \leqq j \leqq k)$ if $\bar{\Omega}^1((\bar{s}_0, \bar{s}_1, \ldots, \bar{s}_j)) = b_j \ldots b_1 u_0(1, \ldots, v(u_0))[\{1, \ldots, v(u_0)\}, \vartheta_0]$ $(b_j \in A_j, \ldots, b_1 \in A_1, u_0 \in G_0 \cup Y_0, \vartheta_0$: $\{1, \ldots, v(u_0)\} \to T_{G_0}(Y_0))$ then $\bar{\varkappa}_0(s_0) = u_0$, for $i=1, \ldots, j$, $\varkappa_i(s_i)$ has the form

$$\begin{cases} (b_i \ldots b_1, u_0, \sigma_i(1, \ldots, v^i(\sigma^i))[\{1, \ldots, v^i(\sigma_i)\}, \bar{\varphi}_i], \bar{\varrho}_i, \bar{W}_i, \bar{\tau}_i) & \text{if } 1 \leqq i \leqq k-1, \\ (b_k \ldots b_1, u_0, u_k[S_{u_k}, \varphi_k], \varrho_k, W_k, \tau_k) & \text{if } i = k, \end{cases}$$

moreover $(\bar{\varkappa}_0(\bar{s}_0), \ldots, \bar{\varkappa}_j(\bar{s}_j)) \in \bar{V}_j$.

Take the mappings $\varkappa_i$: $[Z^1]_i \to \Sigma_{\mathfrak{B}}(i)$ for $i=0, \ldots, k-1$ defined by $\varkappa_i(s_i) = \gamma_i^{-1}(\bar{\varkappa}_i(\alpha_i(s_i)))$ for each $s_i \in [Z^1]_i$. Notice that $\varkappa_i$ is well defined, because α_i and γ_i are bijective. According to Definition 3.2 for each $\bar{s}_k \in [Z^1]_k$, $\Theta^1(s_k)$ is the only element of $\bar{Z}^1$ which has the form $(\bar{s}_0, \ldots, \bar{s}_{k-1}, \bar{s}_k)$ for some $\bar{s}_0, \ldots, \bar{s}_{k-1} \in N^*$. We know that $(\bar{\varkappa}_0(\bar{s}_0), \ldots, \bar{\varkappa}_{k-1}(\bar{s}_{k-1}), \bar{\varkappa}_k(\bar{s}_k)) \in \bar{V}_k$. We can apply condition (1) stated for $\gamma_0, \ldots, \gamma_{k-1}, \gamma_k$, which tells us that there is a unique $\sigma_k \in \Sigma_{\mathfrak{B}}(k)$ such that $\gamma_k(\sigma_k) = \bar{\varkappa}_k(\bar{s}_k)$ and $(\gamma_0^{-1}(\varkappa_0(s_0)), \ldots, \gamma_{k-1}^{-1}(\varkappa_{k-1}(s_{k-1})), \sigma_k) \in V_k$. Let $\varkappa_k(\bar{s}_k)$ be σ_k. By the induction hypothesis for each $(s_0, s_1, \ldots, s_j) \in Z^1$ $(1 \leqq j \leqq k)$, $\Omega^1((s_0, s_1, \ldots, s_j)) = \bar{\Omega}^1((\alpha_0(s_0) \alpha_1(s_1), \ldots, \alpha_j(s_j)))$. Since $\varkappa_0 = \bar{\varkappa}_0$ and for each $s_i \in [Z^1]_i$ the first two components of $\varkappa_i(s_i)$ are equal to the first two components of $\bar{\varkappa}_i(\alpha_i(s_i))$ for $i=1, \ldots, k$, moreover for every $(s_0, s_1, \ldots, s_j) \in Z^1$ $(1 \leqq j \leqq k)$, $(\varkappa_0(s_0), \varkappa_1(s_1), \ldots, \varkappa_j(s_j)) \in V_j$ it follows that the mappings $\varkappa_i$ $(i=0, 1, \ldots, k)$ satisfy condition (1) in the Definition 3.3.

The mappings $\varkappa_i$ $(i=0, 1, \ldots, k)$ uniquely determine a configuration $K_2 = (q^2[S_{q^2}, \psi^2], \Theta^2, Z^2, \Omega^2)$ such that $K_1 \Rightarrow_{\mathfrak{C}} K_2$ holds.

From now on the proof of the second part of the statement is similar to the proof of the first part.

The proof of the theorem is complete.

4. Example

Let us consider the folloving two R-transducers:

$$\begin{aligned}
\mathfrak{A}_1 &= (G_0, Y_0, A_1, G_1, Y_1, A_1', \Sigma_{\mathfrak{A}_1}), \text{ where} \\
G_0 &= G_0^2 = \{g_0\}, \; Y_0 = \{x_0\}, \\
G_1 &= G_1^2 = \{g_1\}, \; Y_1 = \{x_1, y_1\}, \\
A_1 &= \{a_1, b_1, c_1\}, \; A_1' = \{a_1\}, \\
\Sigma_{\mathfrak{A}_1} &= \{b_1 x_0 \to y_1, \; b_1 x_0 \to x_1, \\
&\quad a_1 g_0 \to g_1(1, 2)[\{1, 2\}, \varphi_{11}\colon 1 \mapsto b_1 1; \; \varphi_{11}\colon 2 \mapsto b_1 2], \\
&\quad a_1 g_0 \to g_1(1, 2)[\{1, 2\}, \varphi_{12}\colon 1 \mapsto b_1 1; \; \varphi_{12}\colon 2 \mapsto c_1 2]\}. \\
\tau_{\mathfrak{A}_1} &= \{(g_0(x_0, x_0), g_1(y_1, y_1)), (g_0(x_0, x_0), g_1(x_1, x_1)), \\
&\quad (g_0(x_0, x_0), g_1(x_1, y_1)), (g_0(x_0, x_0), g_1(y_1, x_1))\}. \\
\mathfrak{A}_2 &= (G_1, Y_1, A_2, G_2, Y_2, A_2', \Sigma_{\mathfrak{A}_2}), \text{ where}
\end{aligned}$$

$G_2 = G_2^2 = \{g_2\}, Y_2 = \{x_2, y_2, z_2\},$

$A_2 = \{a_2, b_2\}, A_2' = \{a_2\}.$

$\Sigma_{\mathfrak{A}_2} = \{a_2 g_1 \to g_2(1,2)[\{1,2\}, \varphi_{21}\colon 1 \mapsto b_2 1;\ \varphi_{22}\colon 2 \mapsto b_1 1],$
$b_2 x_1 \to y_2,\ b_2 x_1 \to z_2,\ b_2 y_1 \to x_2\}.$

One can see that

$\tau_{\mathfrak{A}_1} \circ \tau_{\mathfrak{A}_2} = \{(g_0(x_0, x_0), g_2(x_2, x_2)), (g_0(x_0, x_0), g_2(y_2, y_2)),$
$(g_0(x_0, x_0), g_2(y_2, z_2)), (g_0(x_0, x_0), g_2(z_2, y_2)),$
$(g_0(x_0, x_0), g_2(z_2, z_2))\}.$

We construct the 2-synchronized R-transducer $\mathfrak{B}$ according to the Theorem 3.5:

$\mathfrak{B} = (G_0, G_1, Y_0, Y_1, A_1, A_2, A_1', A_2', \Sigma_{\mathfrak{B}}, V)$, where

$\Sigma_{\mathfrak{B}}(0) = V_0 = G_0 \cup Y_0,$

$\Sigma_{\mathfrak{B}}(1) = \{\sigma_1, \sigma_2, \sigma_3, \sigma_4\}$, where $\sigma_1 = (b_1, x_0, y_1, \emptyset, \emptyset, \emptyset),$

$\sigma_2 = (b_1, x_0, x_1, \emptyset, \emptyset, \emptyset),$

$\sigma_3 = (a_1, g_0, g_1(1,2)[\{1,2\}, \varphi_3\colon 1 \mapsto b_1 1;\ \varphi_3\colon 2 \mapsto b_1 2],$
$\varrho_3\colon 1 \mapsto (1,1);\ \varrho_3\colon 2 \mapsto (2,2), \{(1,1), (2,2)\},$
$\tau_3\colon (1,1) \mapsto b_1 1;\ \tau_3\colon (2,2) \mapsto b_1 2),$

$\sigma_4 = (a_1, g_0, g_1(1,2)[\{1,2\}, [\varphi_4\colon 1 \mapsto b_1 1;\ \varphi_4\colon 2 \mapsto c_1 2],$
$\varrho_4\colon 1 \mapsto (1,1);\ \varrho_4\colon 2 \mapsto (2,2), \{(1,1), (2,2)\},$
$\tau_4\colon (1,1) \mapsto b_1 1;\ \tau_4\colon (2,2) \mapsto c_1 2).$

$V_1 = \{(x_0, \sigma_1), (x_0, \sigma_2), (g_0, \sigma_3), (g_0, \sigma_4)\}.$

$\Sigma_{\mathfrak{B}}(2) = \{\sigma_5, \sigma_6, \sigma_7, \sigma_8, \sigma_9\}$, where $\sigma_5 = (b_2 b_1, x_0, x_2, \emptyset, \emptyset, \emptyset),$

$\sigma_6 = (b_2 b_1, x_0, y_2, \emptyset, \emptyset, \emptyset),\ \sigma_7 = (b_2 b_1, x_0, z_2, \emptyset, \emptyset, \emptyset),$

$\sigma_8 = (a_2 a_1, g_0, g_2(1,2)[\{1,2\}, \varphi_8\colon 1 \mapsto b_2 b_1 1;\ \varphi_8\colon 2 \mapsto b_2 b_1 1],$
$\varrho_8\colon 1 \mapsto (1,1,1);\ \varrho_8\colon 2 \mapsto (1,1,2), \{(1,1,1), (1,1,2), (2,2)\},$
$\tau_8\colon (1,1,1) \mapsto b_2 b_1 1;\ \tau_8\colon (1,1,2) \mapsto b_2 b_1 1;\ \tau_8\colon (2,2) \mapsto b_2 1),$

$\sigma_9 = (a_2 a_1, g_0, g_2(1,2)[\{1,2\}, \varphi_9\colon 1 \mapsto b_2 b_1 1;\ \varphi_9\colon 2 \mapsto b_2 b_1 1],$
$\varrho_9\colon 1 \mapsto (1,1,1);\ \varrho_9\colon 2 \mapsto (1,1,2), \{(1,1,1), (1,1,2), (2,2)\},$
$\tau_9\colon (1,1,1) \mapsto b_2 b_1 1;\ \tau_9\colon (1,1,2) \mapsto b_2 b_1 1;\ \tau_9\colon (2,2) \mapsto c_1 1).$

$V_2 = \{(x_0, \sigma_1, \sigma_5), (x_0, \sigma_2, \sigma_6), (x_0, \sigma_2, \sigma_7), (g_0, \sigma_3, \sigma_8), (g_0, \sigma_4, \sigma_9)\}.$

Let us consider configurations $K_0, K_1, K_2, K_3, K_4, K_5, K_6$ of $\mathfrak{B}$, where K_0 is a starting configuration, K_2, K_3, K_4, K_5, K_6 are final configurations.

$K_0 = (a_2 a_1 g_0(x_0, x_0), \Theta_0\colon e \mapsto (e, e, e), \{(e, e, e)\}, \quad \Omega_0\colon (e, e, e) \mapsto a_2 a_1 g_0(x_0, x_0)),$

$K_1 = (g_2(b_2 b_1 x_0, b_2 b_1 x_0), \Theta_1\colon 1 \mapsto (1, 1, 1); \Theta_1\colon 2 \mapsto (1, 1, 2),$

$\{(1, 1, 1), (1, 1, 2), (2, 2)\}, \Omega_1\colon (1, 1, 1) \mapsto b_2 b_1 x_0;$

$\Omega_1\colon (1, 1, 2) \mapsto b_2 b_1 x_0; \Omega_1\colon (2, 2) \mapsto b_1 x_0),$

$K_2 = (g_2(x_2, x_2), \emptyset, \emptyset, \emptyset),$

$K_3 = (g_2(y_2, y_2), \emptyset, \emptyset, \emptyset),$

$K_4 = (g_2(y_2, z_2), \emptyset, \emptyset, \emptyset),$

$K_5 = (g_2(z_2, y_2), \emptyset, \emptyset, \emptyset),$

$K_6 = (g_2(z_2, z_2), \emptyset, \emptyset, \emptyset).$

All the transitions from configuration K_0 in $\mathfrak{B}$ which are ended by final configuration are the following:

$K_0 \Rightarrow_{\mathfrak{B}} K_1 \Rightarrow_{\mathfrak{B}} K_2,$

$K_0 \Rightarrow_{\mathfrak{B}} K_1 \Rightarrow_{\mathfrak{B}} K_3,$

$K_0 \Rightarrow_{\mathfrak{B}} K_1 \Rightarrow_{\mathfrak{B}} K_4,$

$K_0 \Rightarrow_{\mathfrak{B}} K_1 \Rightarrow_{\mathfrak{B}} K_5,$

$K_0 \Rightarrow_{\mathfrak{B}} K_1 \Rightarrow_{\mathfrak{B}} K_6.$

The transition $K_0 \Rightarrow_{\mathfrak{B}} K_1$ is determined by the mappings:

$\varkappa_0\colon \{e\} \to \Sigma_{\mathfrak{B}}(0); \varkappa_0(e) = g_0,$

$\varkappa_1\colon \{e\} \to \Sigma_{\mathfrak{B}}(1); \varkappa_1(e) = \sigma_3,$

$\varkappa_2\colon \{e\} \to \Sigma_{\mathfrak{B}}(2); \varkappa_2(e) = \sigma_8.$

The transition $K_1 \Rightarrow_{\mathfrak{B}} K_2$ is determined by the mappings:

$\varkappa_0\colon \{1, 2\} \to \Sigma_{\mathfrak{B}}(0); \varkappa_0(1) = x_0; \varkappa_0(2) = x_0,$

$\varkappa_1\colon \{1, 2\} \to \Sigma_{\mathfrak{B}}(1); \varkappa_1(1) = \sigma_1; \varkappa_1(2) = \sigma_2,$

$\varkappa_2\colon \{1, 2\} \to \Sigma_{\mathfrak{B}}(2); \varkappa_2(1) = \sigma_5; \varkappa_2(2) = \sigma_5.$

The transition $K_1 \Rightarrow_{\mathfrak{B}} K_3$ is determined by the mappings:

$\varkappa_0\colon \{1, 2\} \to \Sigma_{\mathfrak{B}}(0); \varkappa_0(1) = x_0; \varkappa_0(2) = x_0,$

$\varkappa_1\colon \{1, 2\} \to \Sigma_{\mathfrak{B}}(1); \varkappa_1(1) = \sigma_2; \varkappa_1(2) = \sigma_1,$

$\varkappa_2\colon \{1, 2\} \to \Sigma_{\mathfrak{B}}(2); \varkappa_2(1) = \sigma_6; \varkappa_2(2) = \sigma_6.$

The transition $K_1 \Rightarrow_{\mathfrak{B}} K_4$ is determined by the mappings:

$\varkappa_0$: $\{1, 2\} \to \Sigma_{\mathfrak{B}}(0)$; $\varkappa_0(1) = x_0$; $\varkappa_0(2) = x_0$,

$\varkappa_1$: $\{1, 2\} \to \Sigma_{\mathfrak{B}}(1)$; $\varkappa_1(1) = \sigma_2$; $\varkappa_1(2) = \sigma_2$,

$\varkappa_2$: $\{1, 2\} \to \Sigma_{\mathfrak{B}}(2)$; $\varkappa_2(1) = \sigma_6$; $\varkappa_2(2) = \sigma_7$.

The transition $K_1 \Rightarrow_{\mathfrak{B}} K_5$ is determined by the mappings:

$\varkappa_0$: $\{1, 2\} \to \Sigma_{\mathfrak{B}}(0)$; $\varkappa_0(1) = x_0$; $\varkappa_0(2) = x_0$,

$\varkappa_1$: $\{1, 2\} \to \Sigma_{\mathfrak{B}}(1)$; $\varkappa_1(1) = \sigma_2$; $\varkappa_1(2) = \sigma_2$,

$\varkappa_2$: $\{1, 2\} \to \Sigma_{\mathfrak{B}}(2)$; $\varkappa_2(1) = \sigma_7$; $\varkappa_2(2) = \sigma_6$.

The transition $K_1 \Rightarrow_{\mathfrak{B}} K_6$ is determined by the mappings:

$\varkappa_0$: $\{1, 2\} \to \Sigma_{\mathfrak{B}}(0)$; $\varkappa_0(1) = x_0$; $\varkappa_0(2) = x_0$,

$\varkappa_1$: $\{1, 2\} \to \Sigma_{\mathfrak{B}}(1)$; $\varkappa_1(1) = \sigma_2$; $\varkappa_1(2) = \sigma_2$,

$\varkappa_2$: $\{1, 2\} \to \Sigma_{\mathfrak{B}}(2)$; $\varkappa_2(1) = \sigma_7$; $\varkappa_2(2) = \sigma_7$.

One can see that $\tau_{\mathfrak{B}} = \tau_{\mathfrak{A}_1} \circ \tau_{\mathfrak{A}_2}$.

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

References

[1] DAUCHET, M., Transduction de forêts, bimorphismes de magmoides. — Thèse de doctorat, Universite de Lille I (1977).
[2] GÉCSEG, F., STEINBY, M., Tree automata, Akadémiai Kiadó, Budapest, 1984.

(Received Sept. 20, 1985.)

Bibliographie

Franco P. Preparata, Michael Ian Shamos, Computational Geometry, An introduction (Texts an Monographs in Computer Science) XIII+390, Springer-Verlag, 1985

"The objective of this book is a unified exposition of the wealth of results that have appeared-mostly in the past decade — in Computational Geometry. This young discipline — so christened in its current connotation by one of us, M. I. Shamos — has attracted enormous research interest, and has grown from a collection of scattered results to a mature body of knowledge. This achieved maturity, however, does not prevent computational geometry from being a continuing source of problems and scientific interest."

The book is divided into eight chapters.

Chapter 1 starts with a short historical survey and contains the necessary concepts of the geometry of convex sets, metric and combinatorial geometry, the theory of algorithms, complexity theory and data structures. Some special data structures are introduced, such as the segment tree and the doubly-connected-edge-list. Here can be found the famous Ben-Or theorem about the depth of an algebraic decision tree that solves the membership problem in a subset of E^n. This theorem will be the basic tool for proving lower bound results.

Chapter 2 develops the basic methods of geometric searching that will be used in the succeeding chapters to solve rather formidable problems. Two types of questions are considered. The first one is to determine whether a given point is internal to a simple or a convex polygon. These questions can be answered in $O(N)$ time (for an N-gon) without preprocessing, but in $O(\log N)$ time if given $O(N)$ space and $O(N)$ preprocessing time. The generalization of this problem is to locate a point in a planar subdivision generated by a planar straight-line graph. There are several efficient algorithms for this question, such as the planar-separator method and the triangulation method.

The second class of problems is the range searching problems, which may be viewed as dual, in some sense, of the previously discussed point-location problems. Some quite interesting and clever methods are illustrated for these problems.

Chapter 3 deals with one of the central questions of computational geometry: the determination of convex hull. Ben-Or's result is applied to give a lower bound $\Omega(N \log N)$ and then optimal algorithms are considered in two dimension, such as Graham's scan, Jarvis's march, divide-and-conquer and dynamic algorithms. For the more complicated higher-dimensional cases the gift-wrapping and beneath-beyond methods are presented with complexity $O(N^{\lfloor d/2 \rfloor + 1})$. However, very surprisingly, in the most important three-dimensional case the problem can be solved in optimal time $O(N \log N)$.

Chapter 4 is devoted to the discussion of extensions and applications of the convex hull algorithms. The average case analysis of Jarvis's algorithm gives the $O(N)$ expected time, and an approximation algorithm for convex hull is presented which is quite efficient for statistical problems. The remaining part of the chapter deals with the applications. Their variety should convince the reader that the hull problem is important both in practice and as a fundamental tool in computational geometry.

Chapter 5 is concerned with proximaty problems: closest pair, all nearest neighbors, euclidean spanning tree, triangulation. After proving the $\Omega(N \log N)$ lower bound for these problems a divide-and-conquer scheme is presented to solve the closest pair problem in $O(N \log N)$ time. The main

objective of this chapter is to develop the quite fruitful concept of the Voronoi diagram, which contains all of the proximity information defined by the given set. The algorithmic construction of the Voronoi diagram is given and is then applied to obtain optimal algorithms for the first two of the above — mentioned problemes.

In Chapter 6, continuing the discusson of the Voronoi diagram, an afficient triangulation method is presented and this gives optimal algorithms for the euclidean spanning tree problem and approximate solutions for the euclidean travelling salesmen and euclidean matching problems. In the remaining part several generalizations of the Voronoi diagram are obtained and further applications can be found.

Chapter 7 starts the study of intersection problems by selecting some applications from various fields to motivate these questions. These are the hidden-line and hidden-surface problems, pattern recognition, wire and component layout and linear programming. Efficient algorithms are given for the intersection of convex polygons, star-shaped polygons and line segments in the planar case, while the intersection of convex polyhedra in three dimensions can be determined by a good algorithm, although it is not known whether it is optimal or not.

Chapter 8 is denoted to the study of the geometry of rectangles, which has not only theorethical interest but is the fundamental ingredient of a number of applications, such as Very—Large—Scale—Integration and concurrency controls in data-bases. Using the results of the previous chapters, efficient algorithms are given to determine the measure perimeter, contour, closure and external contour of a union of rectangles and intersections of rectangles.

The book is written in a nice style. Each section is followed by additional notes and comments and a collection of interesting exercises. At the end of the book, very good up-to-date references can be found.

This excellent book is recommended to mathematicians intending to specialize in computationa geometry, and also to non-specialists who are interested in the recent advances in computational geometry.

J. KINCSES

J. P. Tremblay, P. G. Sorenson: The theory and practice of compiler writing XIX + 796 pages, McGraw-Hill Book Company, 1985.

The book deals with all aspects of compiler writing, mainly from a practical point of view. The reader familiar with basic notions of programming languages and grammars can use the chapters independently as a reference book in designing compiler modules. In the discussion of the different technics, after a short overview of motivation and the theoretical background, the algorithms are given in full detail textually. The language used to formulate the algorithms is easy to read. Numerous exercises serve the self-study in compiler design. Each chapter contains an appropriate bibliography.

The main chapters are: programming language design (with an overview of ADA as an example); scanners (regular grammars and finite-state acceptors); top-down parsing (SLL(1), LL(1) parsers); bottom-up parsing (operator precedence, simple precedence grammars, LR(0), SLR(1), LALR(1), LR(1) parsers); compile-time error handling; symbol-table handling; sum-time storage organization; semantic analysis; code generation and optimization; compiler-compilers.

Á. MAKAY

William A. Foley, Robert D. Van Valin, Jr.: Functional Syntax and Universal Grammar. Cambridge University Press, 1984. 416 p.

This book is the result of an effort to develop a grammar which is based on the function of language. As the title suggests, the authors have made an attempt to construct a theory of syntax which is wide enough to cover linguistic phenomena in a great number of languages.

The approach represents a combination of analysis from different levels: the authors view language in function not as a *set of isolated simple sentences*, but rather as a piece of discourse constituted by *complex expressions*. These expressions are made up of a number of clauses linked together in various ways. This linkage is chosen as the starting-point for the integration of linguistic phenomena from different levels. Thus, the main concern of this book is the investigation of the relationship of "clause-internal morhosyntax to clause linkage and cross-caluse reference-tracking mechanisms". The authors argue that the morphosyntactical analysis of the clause must proceed from an interclausal and ultimately discourse perspective.

The investigation of discourse function in carried out within the theory of Role and Reference Grammar (RRG), which W. Foley and R. Van Valin have been developing since the publication of its preliminary sketch in 1980.

If the evolution of linguistics is marked by the antagonism of formalism and functionalism, then it is the latter school of thought to which this monograph adheres. And if we view formalism as an orientation based on the assumption that language is a potentially infinite set of structural descriptions of sentences, then functionalism must be assumed to deal with language in relation to its role in human communication. This is the theoretical distinction which is made clear in Chapter 1.

The remaining six chapters are devoted to an exploration of the means which languages use to code participants and situations in (narrative) discourse, particularly the tracking of participants across clause sequences.

Chapter 2 is concerned with predicate semantics, and Chapter 3 deals with case marking following Silverstein's basic assumptions. Chapter 4 is a presentation of passivization and antipassivization, whereas Chapter 5 goes beyond the confines of clause-internal syntax, discussing a number of issues pertaining to complex sentences. Chapter 6 is an extension of the investigation into clause linkage with a discussion of *nexus*, a term referring to the relations that hold between clauses in complex sentences. Chapter 7 is an analysis of reference-tracking in discourse.

It should be noted that functionalism as an orientation alongside and, to some extent, against linguistic formalism represented by such outstanding theoreticians as N. Chomsky or R. Montague, has attracted much attention for the past two or three years. This is probably due to an ever-growing interest in discourse analysis, which has now become an integral part of present day functional grammars, as evidenced both by the book at hand and by M. A. K. Halliday's recent monograph. If the latter is an introduction to functional grammar based on English, then Functional Syntax and Universal Grammar is a book with a rich illustrative material from a wide variety of languages.

It is the reviewer's contention that all those interested in the theory and use of language will find this book valuable and stimulating.

I. BÉKÉSI

M. Berger: Computer graphics with Pascal. XVII+347 pages, The Benjamin/Cummings Pubishing Company, Inc. 1986.

The book is published in the Benjamin/Cummings Series in Computing and Information Sciences.

"The text begins with a description of the history and applications of computer graphics which is followed by an introduction to the hardware and software components of a graphics system. Included are hardcopy output and input devices, CRT technology, raster-scan and random-vector systems, the display processor, and scan conversion. In Chapter 2 the student begins to draw images using the screen coordinates. The difficulties in drawing basic figures such as lines and circles are explored. The next chapter introduces the reader to the worlds coordinate system and the viewing transformation. Chapter 4 uses the concepts presented in the previous chapter to create business and artistic graphics.

Chapter 5 describes the fundamentals of two-dimensional geometric transformations. Chapter 6 implements the concept of display file segmentation.

Chapter 7 examines the requirements of a user-friendly graphics program. After an initial discussion of the problems inherent in running graphics programs on a minicomputer, the reader is led through a detailed decription of error-handling and menu-generating routines. Chapter 8 treats interactive techniques, while Chapter 9 extends these concepts to animation.

Chapter 10 provides frame buffer and scan conversion algorithms for polygon and area filling. Chapter 11 introduces the coordinate systems and transformations needed for three-dimensional viewing. Chapter 12 describes the generation of realistic images using curves and surfaces, and Chapter 13 extends this realism by implementing hidden-surface removal.

The appendix describes the fundamental features of the two-dimensional graphics standard GKS."

This very clearly written book can be recommended as a text for an introductory course in computer graphics.

J. CSIRIK

G. Reinelt: The Linear Ordering Problem: Algorithms and Applications. XI + 158 pages, Heldermann Verlag Berlin, 1985. (Research and exposition in mathematics; Vol. 8)

The book gives a new algorithm for the linear ordering problem. This problem may be formulated as follows:

"We are given the complete digraph $D_n=(V_n, A_n)$ on n nodes and arc weights C_{ij} for each arc $(i,j)\in A_n$. The linear ordering problem now consists in finding a spanning acyclic tournament in D_n such that the sum of the weights of its arcs is as large as possible. This problem is interesting from a theoretical point of view, and moreover, it has several practical applications in economics, scheduling, sports, and social sciences. In this treatise we solve a number of real-world problems of this type.

In Chapter 1 some basic mathematical definitions and results from graph theory, polyhedral theory and computational complexity theory are surveyed. This introduction is not meant to be comprehensive but is intended to provide the reader with the basic concepts and notations. In Chapter 2 the linear ordering polytope P^n_{LO} is defined, various classes of facet defining inequalities for this polytope are derived, and in addition some remarks concerning adjacency and diameter are made. The chapter ends with the partial description of P^n_{LO} by a set of nonredundant inequalities and equations. This theoretical investigation is a central part of this monograph and lays the foundation of an algorithm for the solution of linear ordering problems whichis discussed in Chapter 3. The computational results of the algorithm when applied to the so-called triangulation problem for input-output tables are reported in Chapter 4. Statistical data such as computing times, number of generated cutting planes or sizes of the linear programs involved are given, the optimization process is illustrated and several cutting plane generation strategies are compared. Since the triangulation problem for input-output tables is an important one in economics (there is a great variety of publications dealing with this problem) we discuss several aspects of it in Chapter 5. We focus attention on how knowledge of "true" optimum solutions can influence or refine previous interpretations and applications made in the literature which were often based on suboptimal solutions. A review of previous algorithms and approaches to the solution of the linear ordering problem is given in Chapter 6. Some more examples of applications of the linear ordering problem are considered in Chapter 7 and complete this tract."

The book is self-contained, and the material is well-arranged. The book can be recommended to everyone interested in combinatorial optimization problems.

J. CSIRIK

The Carnegie-Mellon Curriculum for Undergraduate Computer Science (Edited by Mary Shaw) X + 198 pages, Springer-Verlag, New York Berlin Heidelberg Tokyo, 1985

"This book is a result of a three-year effort by the Carnegie-Mellon Computer Science Department to develop a unified undergraduate computer science curriculum. The study, conducted by an eight-member Curriculum Design Group, responds to this rapidly changing field by emphasizing a balanced blend of fundamental conceptual material which the student can adapt to new situations, with examples drawn from the current practice. This integration of theory and practice is a theme of virtually every course described, recognizing that students must be able to use their theoretical knowledge to generate cost-effective solutions to real problems. This comprehensive redesign of the traditional curriculum reflects the structure of modern computer science. As a result, concepts traditionally distributed over several courses often form the basis for new courses. The book outlines 30 computer science courses along with requirements for an udergraduate major based on this curriculum."

The book is warmly recommended to people dealing with computer science education.

GY. HORVÁTH

A SZERKESZTŐ BIZOTTSÁG CÍME:

6720 SZEGED
SOMOGYI U. 7.

EDITORIAL OFFICE:

6720 SZEGED
SOMOGYI U. 7.
HUNGARY

Information for authors

Acta Cybernetica publishes only original papers in the field of computer sciences mainly in English, but also in French, German or Russian. Authors should submit two copies of manuscripts to the Editorial Board. The manuscript must be typed double-spaced on one side of the paper only. Footnotes should be avoided and the number of figures should be as small as possible. For the form of references, see one of the articles previously published in the journal. A list of special symbols used in the manuscript should be supplied by the authors.

A galley proof will be sent to the authors. The first-named author will receive 50 reprints free of charge.

8 290010 012008 290

INDEX—TARTALOM

ISSN 0324—721 X

Felelős szerkesztő és kiadó: Gécseg Ferenc
A kézirat a nyomdába érkezett: 1985.
Megjelenés: 1986 augusztus
Terjedelem: 11,37 (A/5) iv
Készült monószedéssel, ives magasnyomással
az MSZ 5601 és az MSZ 5602—55 szabvány szerint
86-286 — Szegedi Nyomda — F. v.: Surányi Tibor igazgató