ՀԱՅԿԱԿԱՆ ՍՍՀ ԳԻՏՈՒԹՅՈՒՆՆԵՐԻ ԱԿԱԴԵՄԻԱՅԻ ՏԵՂԵԿԱԳԻՐ
ИЗВЕСТИЯ АКАДЕМИИ НАУК АРМЯНСКОЙ ССР
Մաթեմատիկա X, № 5, 1975 Математика

D. A. EDINJIKLIAN

ON SOME FORMAL REPRESENTATIONS OF ALGORITHMS DEFINED BY GRAPH SCHEMES WITH MEMORY*

§ 0. Introduction. In this paper we shall consider some problems relating to the transformation of graph schemes with memory. The general concept of algorithmic schemes used here is based on the notions of the logical scheme (A. A. Ljapounov [8, 9], Ju. I. Ianov [5], L. A. Kaluznin [7], R. I. Podlovchenko [11, 12]; the concepts concerning the use of memory locations and of the classes of basic algorithms are founded on the notions introduced by A. P. Ershov [2, 3]; the definition of graph scheme with memory and the set of parallel definitions are same as in [15]. The general direction of investigations is in some aspects similar to that of the papers [1], [4], [6], [10], [13], [14].

In [15] definitions were given for the notion of normal closure for a given set of algorithms, and for graph schematic closure. The normal closure of a given set is the set of algorithms which can be obtained rom algorithms of a given set, plus some standard algorithms and the operations of Composition, Branching and Repetition of algorithms, including the adjoining and dropping of fictitious veriables. The graph schematic closure of a given set of algorithms is the set of those algorithms which are specified by graph schemes with memory constructed on the base of the algorithms belonging to the given set. It was proved that graph schematic closure and normal closure are equal for every given set of algorithms.

However, the method of obtaining the given graph schemes in the corresponding normal expression in [15] is essentially dependent on the interpretation of the schemes under consideration.

In the present paper we shall show that the analogous theorems can be obtained on a purely formal level, independently of the interpretations of graph schemes with memory defined in formal terms. In order to do this, the notion of normal closure has been slightly changed, for example, by rejecting the operation of the dropping of fictiti ous variables which cannot be given in formal terms. However, the whole concept of normal closure remains unchanged.

We shall prove that for every graph scheme with memory given in formal terms these exists an algebraical expression using the operations of normal closure and such that for every interpretation of elementary symbols in the given graph scheme, the same algorithm as described by our graph scheme will be expressed. For this purpose we shall

I wish to thank my advisor, I. D. Zaslavskiĭ, for his help and encouragement in the preparation of this paper.

consider the notion of generalized graph scheme having some interest in itself, and we shall prove some theorems about standard forms of these schemes.

The transformations of graph schemes considered here may be useful in studying practical algorithmic languages.

§ 1. *Basic definitions.* In the interest of economy, we shall omit some of the definitions, and the proofs of some of the lemmas, which are similar to those of [15]. Such references will be noted by the symbol [15], followed by the page number. Eg. [15: 139].

We shall consider the following types of variables:

1) Algorithmic functor variables f_1, f_2, $f_3, \cdots$,

2) Algorithmic predicate veriables p_1, p_2, $p_3, \cdots$,

3) Object variables x_1, x_2, $x_3, \cdots$.

The dimensions of an algorithmic functor variable $f_{2^k(2l+1)}$ and of an algorithmic predicate variable $p_{2^k(2l+1)}$ is k.

Def. 1. Let us define F-terms and P-terms inductively as follows:

1) Every f_l is an F-term of corresponding dimension.

2) Every p_l is a P-term of corresponding dimension.

3) I is an F-term of dimension one.

4) D is an F-term of dimension zero.

5) T is a P-term of dimension zero.

6) L is a P-term of dimension zero.

7) If $T_1, \cdots, T_{m+1}$ are F-terms of dimensions $m, n, \cdots, n$ respectively, then Comp $(T_1, \cdots, T_{m+1})$ is an F-term of dimension n.

8) If T_1 is a P-term of dimension m and $T_2, \cdots, T_{m+1}$ are F-terms of dimension n, then Comp $(T_1, \cdots, T_{m+1})$ is a P-term of dimension n.

9) If T_1 is a P-term of dimension n and T_2 and T_3 are F-terms of dimension n, then Br (T_1, T_2, T_3) is an F-term of dimension n.

10) If T_1, T_2 and T_3 are P-terms of dimension n, then Br (T_1, T_2, T_3) is a P-term of dimension n.

11) If T_1 is a P-term of dimension n, and $T_2, \cdots, T_{n+1}$ are F-terms of dimension n, then Rep $(i, T_1, \cdots, T_{n+1})$, where $(1 \leqslant i \leqslant n)$ is an F-term of dimension n.

12) If T is an F-term or a P-term of dimension n, then Adj (i, T) $(0 \leqslant i \leqslant n)$ is correspondingly an F-term or a P-term of dimension $n+1$.

Def. 2. An expression of the form $[T_1, \cdots, T_n]$ is called an A-term of dimension n, iff every $T_i (1 \leqslant i \leqslant r)$ is an F-term of dimension n.

Def. 3. An expression of the form $\alpha \leftarrow C$ is called a generalized memory transformation row iff either α is a functor skeleton variable ([15: 173]) and C is an A-term or α is a predicate skeleton variable ([15: 173]) and C is a P-term.

Def. 4. A system (possibly empty) of generalized memory transformation rows is called a generalized memory transformation table.

Def. 5. A quadruple of objects consisting of a skeleton scheme Σ, ([15: 174]), a generalized memory transformation table Ξ, a system of

input variables W_1 and a system of output variable W_2 will be called a generalized memory transformation scheme of dimension n, denoted by (Σ, Ξ, W_1, W_2) iff the table Ξ is matched ([15: 178]) with the scheme Σ and every A-term or P-term in the right side of the rows of Ξ has a dimension n and all the object variables in each of the systems W_1, W_2 are distinct pairwise and their index set is a subset of $[1, \cdots, n]$.

Def. 6. A pair of objects is called a generalized graph scheme of dimension n in M, denoted by (Ψ, L), iff the first member of the ordered pair is a generalized memory transformation scheme Ψ of dimension n and the second is a matched table of algorithms L ([15: 182]) in the constructive set M ([15: 179]).

Def. 7. The value of an F-term or a P-term R in the state P ([15: 185]) of a generalized graph scheme by $P(R)$ is defined inductively as follows:

1) If the F-term R has the form $f_{2^n(2l+1)}$, then $P(R)$ is defined iff! $R(P(x_1), \cdots, P(x_n))$ and its value is $R(P(x_1), \cdots, P(x_n))$.

2) If the P-term R has the form $P_{2^n(2l+1)}$, then $P(R)$ is defined iff! $R(P(x_1), \cdots, P(x_n))$ and its value is $R(P(x_1), \cdots, P(x_n))$.

3) If the F-term R has the form I, then $P(R)$ is defined and its value is $P(x_1)$.

4) If the F-term R has the form D then $P(R)$ is undefined and has no value.

5) If the P-term R has the form T, then $P(R)$ is defined and its value is true.

6) If the P-term R has the form L, then $P(R)$ is defined and its value is false.

7) If the F-term or the P-term R has the form Comp $(T_1, \cdots, T_{m+1})$ of dimension n, then $P(R)$ is defined iff every $! T_i(P(x_1), \cdots, P(x_n))$ $(1 \leqslant i \leqslant m+1)$ and if $T_i(P(x_1), \cdots, P(x_n)) = a_i$ then $! T_1(a_1, \cdots, a_m)$ and its value is $T_1(a_1, \cdots, a_m)$.

8) If the F-term or the P-term R has the form Br (T_1, T_2, T_3) of dimension n, then $P(R)$ is defined iff either (i) $! T_1(P(x_1), \cdots, P(x_n))$ and has the value true with $! T_2(P(x_1), \cdots, P(x_n))$ or (ii) $! T_1(P(x_1), \cdots$ $\cdots, P(x_n))$ and has the value false with $! T_3(P(x_1), \cdots, P(x_n)) \cdot P(R)$ in the first case is $T_2(P(x_1), \cdots, P(x_n))$ and in the second case $T_3(P(x_1), \cdots, P(x_n))$.

9) If the P-term R has the form Rep $(k, T_1, \cdots, T_{n+1})$, then $P(R)$ is defined iff it is possible to construct a natural number $t \geqslant 0$ and a system of objects $a_i^{(j)}$ $(1 \leqslant i \leqslant n)$ and $(0 \leqslant j \leqslant t)$. Further for $t = 0$ we have $a_i^{(0)} = P(x_i)$ $(1 \leqslant i \leqslant n)$, $! T_1(a_1^{(0)}, \cdots, a_n^{(0)})$ and $T_1(a_1^{(0)}, \cdots, a_n^{(0)})$ is true, whereas for $t > 0$ we have $a_i^{(0)} = P(x_i)(1 \leqslant i \leqslant n)$, $! T_{j+1}(a_1^{(j)}, \cdots$ $\cdots, a_n^{(j)})$ $((1 \leqslant i \leqslant n), \quad (0 \leqslant j < t))$, $a_i^{(j+1)} = T_{i+1}, (a_1^{(j)}, \cdots, a_n^{(j)})$ $((1 \leqslant i \leqslant n), (0 \leqslant j < t))$; $! T_1(a_1^{(j)}, \cdots, a_n^{(j)})$ $(0 \leqslant j \leqslant t)$; $T_1(a_1^{(j)}, \cdots, a_n^{(j)})$ is false $(0 \leqslant j \leqslant t-1)$; and $T_1(a_1^{(t)}, \cdots, a_n^{(t)})$ is true. The value of R in this case will be $a_k^{(t)}$.

10) If the F-term or the P-term R has the form Adj (i, T), then $P(R)$ will be defined iff $P^*(T)$ is defined, where P^* is a memory state such that $\overline{P^*} = \overline{P}$, $P^*(x_j) = P(x_j)$, when $(1 \leqslant j \leqslant i)$; and $P^*(x_j) = P(x_{j+1})$ when $j > i$. If $P(R)$ is defined, then the value $P(R)$ is $P^*(T)$.

The assertion "$P(R)$ is defined" will be denoted by $!\,P(R)$.

Note: In what follows, we shall use the F-terms I_m^j, where $(1 \leqslant j \leqslant m)$, which are defined recursively as (i) $I_1^1 = I$, (ii) $I_{n+1}^{k+1} = \text{Adj}\,(1, I_n^k)$ $(0 \leqslant 1 < k)$, (iii) $I_{n+1}^k = \text{Adj}\,(1, I_n^k)$ $(k \leqslant 1 \leqslant n)$. It is easy to see that if $P = (\alpha, (a_1, \cdots, a_m))$ then $P(I_m^j) = a_j$.

Def. 8. A memory state P of a generalized graph scheme G of dimension n is said to be amenable to G and denoted by $!!\,G(P)$ if one of the following conditions holds:

(i) $\overline{P}$ is a starter.

(ii) $\overline{P}$ is a skeleton functor variable such that if $\overline{P}$ is $[T_1, \cdots, T_n]$ then $!\,P(T_1), !\,P(T_2), \cdots, !\,P(T_n)$.

(iii) $\overline{P}$ is a predicate skeleton variable such that if $\overline{P}$ is Q, where Q is a P-term, then $!\,P(Q)$.

Def. 9. For any memory state P of a generalized graph scheme G of dimension n amenable to G, a direct successor memory state Q of G, denoted by $P \underset{G}{\vdash} Q$, is assigned as follows:

(i) If $\overline{P}$ is a starter, with $|\overline{P}| = (\overline{P}, \varsigma)$, we have $\overline{Q} = \varsigma$, and $Q(x_i) = P(x_i)\,(1 \leqslant i \leqslant n)$.

(ii) If $\overline{P}$ is a functor skeleton variable with $|\overline{P}| = (\overline{P}, \varsigma)$ and $\overline{P} \leftrightarrow [T_1, \cdots, T_n]$ then $\overline{Q} = \varsigma$, and for every $i\,(1 \leqslant i \leqslant n)$ $Q(x_i) = P(T_i)$.

(iii) If $\overline{P}$ is a predicate skeleton variable with $|\overline{P}| = (\overline{P}, \varsigma, \eta)$ and $\overline{P} \leftrightarrow R$, where R is a P-term, then $\overline{Q} = \begin{cases} \varsigma \text{ if } P(R) \text{ is true} \\ \eta \text{ if } P(R) \text{ is false} \end{cases}$ and $Q(x_i) = P(x_i)\,(1 \leqslant i \leqslant n)$.

Def. 10. A generalized graph scheme G of dimension n is called regular iff G is equivalent ([15 : 188]) to itself.

Def. 11. A generalized memory transformation scheme (Σ, Ξ, W_1, W_2) of dimension n is called strongly regular iff for every algorithmic table L in the constructive set M, if $((\Sigma, \Xi, W_1, W_2), L)$ is a generalized graph scheme of dimension n, then it is regular.

Def. 12. Two generalized memory transformation schemes Ψ_1 and Ψ_2 of dimensions n and m respectively are said to be D-equivalent iff for every algorithmic table L, in the constructive set M, if (Ψ_1, L) and (Ψ_2, L) are generalized graph schemes, then they are equivalent.

Def. 13. A generalized functor graph scheme G ([15 : 179]) in M of dimension n with input variables $x_1, \cdots, x_l$ and output variable r is

said to specify a functor U in M of dimension t iff for every $a_1, \cdots, a_t$ in M, the value of $U(a_1, \cdots, a_t)$ is obtained as follows:

(*i*) An initial memory state P of G is constructed such that $\overline{P} = H$ and $P(x_i) = a_i$ $(1 \leqslant i \leqslant t)$ (For object variables other than $x_1, \cdots, x_t$, $P(x_i)$ is taken arbitrary).

(*ii*) The generalized graph scheme is applied ([15:188]) to the state P, and if a terminal state Q is arrived at, then $Q(r)$ is the value of $U(a_1, \cdots, a_t)$.

Def. 14. A generalized predicate graph scheme ([15:179]) G in M of dimension n with input variables $x_1, \cdots, x_t$ is said to specify a predicate P in M of dimension t iff for every $a_1, \cdots, a_t$ in M, the value $P(a_1, \cdots, a_t)$ is obtained as follows:

(*i*) An initial memory state P of G is constructed such that $\overline{P} = H$ and $P(x_i) = a_i$ $(1 \leqslant i \leqslant t)$.

(*ii*) The generalized graph scheme is applied to the state P, and if a terminal state Q is arrived at, then the value of $P(a_1, \cdots, a_t)$ is either true or false depending on whether Q is O_T or O_F respectively.

Def. 15. An F-term R of dimension n is said to realize an algorithm U with respect to L iff U is specified by the generalized graph scheme $((((H, \varphi_1), (\varphi_1, O_k)), \varphi_1 \leftarrow\!\!\rightarrow [R, I_n^2, \cdots, I_n^n]), (x_1, \cdots, x_n), (x_1)), L)$.

Def. 16. A P-term R of dimension n is said to realize an algorithm P with respect to L iff P is specified by the generalized predicate graph scheme $((((H, \Pi_1), (\Pi_1, \omega_1, \omega_2)), (\Pi_1 \leftarrow\!\!\rightarrow R), (x_1, \cdots, x_n), ( )), L)$.

2. *Substitution*: Let $\Psi = (\Sigma, \Xi, W_1, W_2)$ be a generalized memory transformation scheme of dimension n, with the skeleton scheme $\Sigma = (S_1, \cdots, S_k)$. Let Γ be a system consisting of some of its skeleton functor and predicate variables, and let E be a subscheme of Ψ generalized by Γ ([15: 194]).

Let E^* be a generalized memory transformation scheme of dimension m D-equivalent to E in which the systems of input and output variables coincide with the corresponding systems of the generalized memory transformation scheme E. The operation of substitution of a generalized memory transformation scheme E^* into a generalized transformation scheme Ψ instead of subscheme E will be defined as follows:

(*i*) By appropriate renaming ([15:195]) of skeleton and algorithmic variables of the transformation scheme E we obtain a transformation scheme $\tilde{E} = (\tilde{\Sigma}, \tilde{\Xi}, W_1, W_2)$ that differs from E^* only by the name of the variables, and such that all its skeleton functor and predicate variables and algorithmic variables are distinct from the skeleton and algorithmic variables of Ψ.

(*ii*) From the skeleton scheme Σ of the generalized memory transformation scheme Ψ we eliminate all the terms whose first member is contained in Γ, in each of the remaining skeleton Terms ([15:173]) S we perform the following replacements: If the second mem-

ber of the term S occurs in Γ, we replace it by the second member of the (unique) skeleton term of the scheme $\bar{\Sigma}$ whose first member is v_i'; if the third member of S occurs in Γ, we shall replace it by the second member of the skeleton term of the scheme $\bar{\Sigma}$ whose first member is v_i''.

(*iii*) From the skeleton scheme $\tilde{\Sigma}$ of the generalized memory transformation scheme $\tilde{E}$ we eliminate all the initial terms; in each of the remaining skeleton terms we replace each stop ω_i' by the second member of the term S_i, and each stop ω_i'' by the third member of s_i.

(*iv*) We construct a skeleton scheme $\tilde{\Sigma}^*$ which is a union of the skeleton scheme obtained from Σ by the transformations indicated in step 2 and the skeleton scheme obtained from $\tilde{\Sigma}$ by the transformation indicated in step 3.

(*v*) The generalized memory transformation table $\tilde{\Xi}^*$ is constructed as follows:

a) If $m = n$, then $\tilde{\Xi}^*$ is the union of the generalized memory transformation tables Ξ and $\bar{\Xi}$ of the generalized memory transformation schemes Ψ a and E^*.

b) If $m > n$, then a generalized memory transformation table Ξ' of dimension n is obtained by cancelling in Ξ all the rows whose left sides are not contained in Γ. Let us construct a generalited memory transformation table $\bar{\Xi}$ of dimension m, such that each row $\varphi_i \leftarrow [T_1, \cdots \cdots, T_n]$ in Ξ' is replaced by $\varphi_i \leftarrow [T_1', \cdots, T_m']$ where every F-term $T_j'(n+1 < j \leqslant m)$ is I_m^j, and every F-term T_k' $(1 \leqslant k \leqslant n)$ is Adj $(m-1,$ Adj $(m-2, \cdots,$ Adj $(n, T_k)) \cdots)$. And thus $\tilde{\Xi}^*$ is the union of the generalized transformation tables $\bar{\Xi}$ end $\tilde{\Xi}$.

c) Similarly for $n > m$.

(*vi*) We construct a system of objects $\tilde{\Psi}^* = (\tilde{\Sigma}^*, \tilde{\Xi}^*, W_1, W_2)$ where W_1 and W_2 are the systems of input and output variables of the generalized memory transformation scheme Ψ.

It is easy to see that $\tilde{\Psi}^*$ is a generalized memory transformation scheme of dimension max (m, n)

Theorem 1. (*On substitution*). *Let Ψ be a strongly regular generalized memory transformation scheme of dimension n, Γ a system consisting of some of its skeleton functor and predicate variables, λ a subscheme of Ψ generalized by Γ, λ^* a generalized transformation scheme of dimension m, D-equivalent to λ with the system of input and output variables of λ^* coinciding with the systems of input and output variables of λ. Let Ψ^* be the result of substituting the generalized memory*

transformation scheme λ^* for the subscheme of λ into the generalized memory transformation scheme Ψ. Then Ψ^* will be D-equivalent to Ψ.

Proof: Let L be any algorithmic table in the constructive set M such that (Ψ, L) and (Ψ^*, L) are generalized graph schemes of corresponding dimensions. The proof that the generalized graph schemes (Ψ, L) and (Ψ^*, L) are equivalent, is carried in a similar way, as in [15:198—200]. Thus Ψ and Ψ^* are D-equivalent.

§ 3. In this last section, with the aid of a series of lemmas we shall prove the main theorem of this paper.

Lemma 1. For any strongly regular generalized memory transformation scheme Ψ it is possible to construct a D-equivalent one-sided ([15:179]) generalized memory transformation scheme $\bar{\Psi}$, of the same dimension. Proof: The proof is carried in a way similar to the proof of lemma 6.3 of [15:232—233].

Lemma 2. For every one-sided pseudopredicate ([15:233]) strongly regular generalized memory transformation scheme $\Psi = (\Sigma, \Xi, W_1, W_2)$ which does not contain any predicate skeleton variable in Σ it is possible to construct a D-eguivalent generalized memory transformation scheme, $\bar{\Psi} = (\Sigma', \Xi', W_1, W_2)$ of the same dimension as Ψ such that Σ' has the form $((v_1, \Phi_1), (\Phi_1, \omega_1))$.

Proof: The proof is carried in a way similar to the proof of lemma 6.4 of [15:234—235].

Lemma 3. For any generalized memory transformation scheme Ψ of dimension n that satisfies the conditions of lemma 2, it is possible to construct a primitive ([15:233]) generalized memory transformation scheme $\bar{\Psi}$ of dimension n which is D-equivalent to Ψ and has the same systems of input and output variables as Ψ.

Proof: The proof is carried in a way similar to the proof of lemma 6,5 of [15:235—236].

Lemma 4. For any one-sided pseudopredicate strongly regular generalized memory transformation scheme Ψ without cycles [15:234] it is possible to construct a primitive generalized memory transformation scheme $\bar{\Psi}$ D-equivalent to Ψ that has the same dimension and the same systems of input and output variables as Ψ.

Proof: The proof is carried in a way similar to the proof of lemma 6.6 of [15:236—241].

Lemma 5. For any primitive strongly regular generalized functor memory transformation scheme $\Psi = (\Sigma, \Xi, W_1, W_2)$ of dimension n it is possible to construct a generalized functor memory transformation scheme Ψ' of dimension s, where s is the maximum index of the object variables in $W_1 \cup W_2$, which is D-equivalent to Ψ and has a skeleton scheme of the form $((H, \Phi_1), (\Phi_1, \omega_1))$.

Proof: Let Ψ be a primitive strongly regular generalized functor memory transformation scheme of dimension n, which has the form

$(((\omega_1, \Pi_1), (\Pi_1, \Phi_1, \Phi_2), (\Phi_1, \omega_1), (\Phi_2, \omega_2)), (\Pi_1 \dashrightarrow B, \Phi_1 \dashrightarrow [T_1, \cdots, T_n), \Phi_2 \dashrightarrow [T_1, \cdots, T_n]), W_1, W_2)$, where W_1 and W_2 are systems of pairwise distinct variables. Since Ψ is generalised functor memory transformation scheme, we have $\omega_1 = H$, $\omega_2 = O_k$ and $\omega_2 = O_k$, the system W_2 will consist of one variable, denoted by x_l, and W_1 will be either empty or have variables which are denoted by $x_{k_1}, \cdots, x_{k_t} (t \leqslant n)$.

Case 1: Suppose W_1 is empty and there exists at least a functor algorithmic variable (F-term), call it T_0, of dimension 0. Then the required generalized memory transformation scheme Ψ' is $(((H, \Phi_1), (\Phi_1, \omega_1)), (\Phi_1 \dashrightarrow [T_1^*, \cdots, T_l^*]), W_1, W_2)$, where $T_i^* = \mathrm{Adj}(0, \mathrm{Comp}(Br(B, T_i, T_i), \overbrace{T_0, \cdots, T_0}^{n}) \cdots)(1 \leqslant i \leqslant l)$, which has dimension l.

Case 2: Suppose W_1 is empty and there is no functor algorithmic variable of dimension 0 in Ψ. Then Ψ' of dimension l is $(((H, \Phi_1), (\Phi_1, \omega_1)), (\Phi_1 \dashrightarrow [I_l, \cdots, I_1^{l-1}, \mathrm{Adj}(l-1, \cdots, \mathrm{Adj}(0, D) \cdots)], W_1, W_2)$.

Let us show that Ψ' is D-equivalent to Ψ. From the construction of Ψ', it is obvious that for any algorithmic table L in a constructive set M and for any initial memory state P of the generalized graph scheme (Ψ', L), (Ψ', L) is inapplicable to P. Hence Ψ' is strongly regular.

Subcase 1: Suppose Ψ is a generalized memory transformation scheme such that for every algorithmic table L in a constructive set M (Ψ, L) is a generalized graph scheme, and for every initial memory state P in M (Ψ, L) is inapplicable to P. Hence Ψ is strongly regular and D-equivalent to Ψ'.

Subcase 2: Suppose L is an algorithmic table in a constructive set M such that (Ψ, L) is a generalized graph scheme and there exists an initial memory state P in M such that (Ψ, L) is applicable to P. It is obvious that Ψ and Ψ' are not D-equivalent. In this case let us show that Ψ is not strongly regular.

(*i*) If L is empty, then Ψ is not strongly regular, since W_1 is empty and W_2 is non-empty.

(*ii*) If L is non-empty, let us construct a constructive set M^* from M such that $M^* = M \cup \{a^*\}$, where $a^* \& M$. Let us construct an algorithmic table L^* in M^* from L by replacing every algorithm U of dimension $k (k \leqslant n)$ in L by an algorithm U^* of the same dimension in M^* as follows:

$$U^*(r_1, \cdots, r_k) = \begin{cases} U(r_1, \cdots, r_k), & \text{if there exists no } i\ (1 \leqslant i \leqslant k) \\ & \text{such that } r_i = a^*; \\ \text{undefined, otherwise.} \end{cases}$$

Hence (Ψ, L^*) is a generalized graph scheme.

The initial memory state P is also applicable to (Ψ, L^*) in M. Let P be an initial memory state such $P^*(x_i) = a^* (1 \leqslant i \leqslant n)$, then P^* is inapplicable to (Ψ, L^*), since L is non-empty. Hence (Ψ, L^*) is not regular since W_1 is empty, and thus Ψ is not strongly regular.

Hence the only case that Ψ is strongly regular is in subcase 1 which is D-equivalent to Ψ'.

Case 3: Suppose W_1 is not empty, and let x_s is an object variable in $W_1 \cup W_2$ with maximum index. Then the required generalized memory transformation scheme Ψ' of dimension s, D-equivalent to Ψ is constructed as follows: $(((H, \Phi_1), (\Phi_1\ \omega_1)))$, $\Phi_1 \leftarrow [T_1^*, \cdots, T_s^*])$, W_1, W_2), where for every $i\ (1 \leqslant i \leqslant s)$ $T_i^* = \mathrm{Comp}\,(Br\,(B, T_i, T_i), I_s^*, \cdots, I_s^*)$.

It is easily seen that Ψ and Ψ' are D-equivalent.

This completes the proof of the lemma.

Lemma 6. For any premitive strongly regular generalized predicate memory transformation scheme $\Psi = (\Sigma, \Xi, W_1, W_2)$ of dimension n, it is possible to construct a generalized predicate memory transformation scheme Ψ' of dimension s, where s is the maximum index of the object variable in $W_1 \cup W_2$, which is D-equivalent to Ψ and has a skeleton scheme of the form $((H, \Pi_1), (\Pi_1, \omega_1, \omega_2))$.

Proof: Let Ψ be a primitive strongly regular generalized predicate memory transformation scheme of dimension n, which has the form $(((\upsilon_1, \Pi_1), (\Pi_1, \varphi_1, \varphi_2), (\varphi_1, \omega_1), (\varphi_2, \omega_2))$, $(\Pi_1 \leftarrow B, \varphi_1 \leftarrow [T, \cdots, T_n], \varphi_2 \leftarrow [T_1, \cdots \cdots, T_n])$, W_1, $W_2)$, where W_1 and W_2 are systems of pairwise distinct variables. Since Ψ is a generalized predicate memory transformation scheme, we have $\upsilon_1 = H$, and each of the stops ω_1 and ω_2 is equal either to O_T or O_F and W_2 is empty.

Case 1: Suppose W_1 is empty and there exists at least a functor algorithmic variable (F-term), call it T_0, of dimension 0 in Ψ. Then the required generalized transformation scheme Ψ' of dimension 0 is $(((H, \Pi_1), (\Pi_1, \omega_1, \omega_2))$, $(\Pi_1 \leftarrow \mathrm{Comp}\,(Br\,(B, \mathrm{Comp}\,(R_1, T_1 \cdots, T_n), \mathrm{Comp}\,(R_2, T_1, \cdots, T_n)), \overbrace{T_0, \cdots, T_0}^{n}))$, W_1, $W_2)$ where $R_i\ (i = 1, 2)$ is either $\mathrm{Adj}\,(n-1, \cdots, \mathrm{Adj}\,(0, \mathrm{T}) \cdots)$ or $\mathrm{Adj}\,(n-1, \mathrm{Adj}\,(n-2, \cdots \cdots, \mathrm{Adj}\,(0, \mathrm{L})) \cdots)$ in the cases $\omega_i = O_T$ or $\omega_i = O_F$ respectively. It is clear that Ψ' is D-equivalent to Ψ.

Case 2: Suppose W_1 is empty and there is no functor algorithmic variable with dimension 0 in Ψ. Then two subcases arise:

Subcase 1: Suppose Ψ is a generalized predicate memory transformation scheme such that Ψ does not contain D, and for an empty algorithmic table L in a constructive set M, (Ψ, L) is a generalized graph scheme. Then Ψ' is D-equivalent either to $(((H, \Pi_1), (\Pi_1, \omega_1, \omega_2))$, $(\Pi_1 \leftarrow \mathrm{T})$, W_1, $W_2)$ if B is $\mathrm{Adj}\,(n-1, \cdots, \mathrm{Adj}\,(0, \mathrm{T}))$ or to $(((H, \Pi_1), (\Pi_1, \omega_1, \omega_2))$, $(\Pi_1 \leftarrow \mathrm{L})$, W_1, $W_2)$ if B is $\mathrm{Adj}\,(n-1, \cdots, \mathrm{Adj}\,(0, \mathrm{L}) \cdots)$.

Subcase 2: Suppose Ψ is a generalised predicate memory transformation scheme, such that either Ψ contains the F-term D, or for an empty algorithmic table L, (Ψ, L) is not a generalized graph scheme. Then the required generalized predicate memory transformation scheme of dimension 0 is $(((H, \Pi_1), (\Pi_1, \omega_1, \omega_2))$, $(\Pi_1 \leftarrow \mathrm{Comp}\,(\mathrm{Adj}\,(0, \mathrm{T}), \mathrm{D}))$, W_1, $W_2)$.

We show that Ψ is D-equivalent to Ψ' by a similar method as in case 2 of lemma 5.

Case 3: Suppose W_1 is not empty. Let x_s be an object variable in W_1 with maximum index. Let us construct P-terms R_i $(i=1,2)$, such that each has one of either forms Adj $(n-1, \cdots,$ Adj $(0, T)\cdots)$ or Adj $(n-1, \cdots,$ Adj $(0,$ L$))\cdots)$ in case $\omega_i = O_T$ or $\omega_i = O_F$ respectively. The required generalized predicate memory transformation scheme Ψ' of dimension s is constructed as follows: $(((H, \Pi_1), (\Pi_1, \omega_1, \omega_2)), (\Pi_1 \leftarrow\!\!\rightarrow \text{Comp}(Br(B,$ Comp $(R_1, T_1, \cdots, T_n),$ Comp $(R_2, T_1, \cdots, T_n),$ $I_s, \cdots, I_s)),$ $W_1, W_2)$.

It is evident that Ψ' is D-equivalent to Ψ. This completes the proof of the lemma.

Lemma 7: For any one-sided pseudpredicate strongly regular generalized memory transformation scheme Ψ, it is possible to construct a one-sided strongly regular generalized memory transformation scheme $\tilde{\Psi}$ without cycles which is D-equivalent to Ψ and has the same dimension as Ψ.

Proof: Let $\Psi = (\Sigma, \Xi, W_1, W_2)$ be a one-sided strongly regular generalized memory transformation scheme of dimension n that has a skeleton scheme $\Sigma = (S_1, \cdots, S_h)$.

We shall carry out the proof by induction on the number of inverse predicate skeleton variables occurring in Ψ. If this number is equal to zero, the assertion of the lemma is evident, since in this case Ψ is a generalized memory transformation scheme without cycles and we can write $\tilde{\Psi} = \Psi$.

Now let us assume that Ψ has inverse predicate skeleton variables. Let q be the smallest length of a cycle for the inverse predicate skeleton variables occurring in Ψ. We shall denote by $\alpha_1, \cdots, \alpha_h$ the first members, by $\beta_1, \cdots, \beta_h$ the second members, and by $\gamma_1, \cdots, \gamma_h$ the third members of skeleton terms $S_1, \cdots, S_h$.

Since Ψ is a one-sided pseudopredicate generalized memory transformation scheme, we have $\beta_i = \alpha_{i+1}$ for $(1 \leqslant i \leqslant h-1)$, and moreover α_1 will be a starter and β_h will be a stop. Let α_k be an inverse predicate skeleton variable whose cyclehas length q. By denoting the number $k-q+1$ by l, we can write the cycle of the variable α_k in the form $(\alpha_l, \cdots, \alpha_k)$.

Let us denote by λ a subscheme of Ψ generated by the system of skeleton variables $(\alpha_l, \cdots, \alpha_k)$.

Now we shall construct a generalized memory transformation scheme D-equivalent to λ such that by substituting it for λ in Ψ we obtain a generalized memory transformation scheme with a smaller number of inverse predicate skeleton variables as compared to Ψ.

In the skeleton scheme λ let us permute the skeleton terms in such a way that the initial terms precede all the others, whereas the skele-

ton terms which are not initial are located in the same order as in λ; the resulting generalized memory transformation scheme (evidently D-equivalent to λ) will be denoted by λ'. The skeleton scheme of λ' will be $R_1, R_2, \cdots, R_d, S'_l, S'_{l+1}, \cdots, S'_k$, where $R_1, R_2, \cdots, R_d$ are initial terms ($d \geqslant 1$) and $S'_l, \cdots, S'_k$ are skeleton terms $S_l, \cdots, S_k$ that have been transformed in accordance with the definition of a subscheme. Let us denote the first, second and third member of each term S'_l by α'_l, β'_l and γ'_l. Hence, in accordance with the definition of a subscheme, all the α'_l will coincide with α_l and each of the variables β'_l, γ'_l will either coincide with β_l, γ_l or it will be a stop. Moreover, γ'_k coincides with α'_l.

It is easy to see that λ' is one-sided. Since α_z is an inverse variable with smallest length of cycle in Ψ, the generalized memory transformation scheme λ' will have only one inverse predicate skeleton variable α'_k.

Let us construct a generalized memory transformation scheme Δ that can be obtained from λ' by the following transformation:

1) The group of initial terms $R_1, \cdots, R_d$ in the skeleton scheme λ' is replaced by the initial term (H, α'_l).

2) The skeleton term $S'_k = (\alpha'_k, \beta'_k, \alpha'_l)$ in the skeleton scheme of λ' is replaced by the skeleton term (α'_k, O_T, O_F).

3) In all the skeleton terms other than S_1 all the stops are replaced by the stop O_T.

It is clear that Δ is a one-sided generalized memory transformation scheme. By construction, Δ is a pseudopredicate generalized memory transformation scheme without cycles. Since all the object variables contained in Δ belong to the system of its input variables, it follows that Δ is a strongly regular generalized memory transformation scheme. Hence according to lemma 4 we can construct a primitive generalized memory transformation scheme Δ' D-equivalent to Δ which have the form $(((H, \Pi_1), (\Pi_1, \varphi_1, \varphi_2), (\varphi_1, \omega_1), (\varphi_2, \omega_2)), (\Pi_1 \leftarrow\!\!\rightarrow R^*_1, \varphi_1 \leftarrow\!\!\rightarrow [T_1, \cdots, T_n], \varphi_2 \leftarrow\!\!\rightarrow [T_1, \cdots, T_n]), (x_1, \cdots, x_n), (x, \cdots, x_n))$, where each of the stops ω_1 and ω_2 is either O_T or O_F.

Let us construct a generalized memory transformation scheme Δ'' by replacing the system of output variables of Δ' by an empty system of variables. Hence Δ'' will be a primitive strongly regular generalized predicate memory transformation scheme of dimension n. Hence by lemma 6 it is possible to construct a generalized memory transformation scheme Δ^* of dimension n which is D-equivalent to Δ'' and have the form $(((H, \Pi_1), (\Pi_1, \omega_1, \omega_2)), (\Pi \leftarrow\!\!\rightarrow \bar{R}_1), (x_1, \cdots, x_n), (\ ))$. For any algorithmic table L adjoined to λ' such that (λ', L) is a generalized graph scheme, it is easy to verify that the predicate U specified by (Δ^*, L) and realized by the P-term $\bar{R}_1$ has the following property: for any state P of the generalized graph scheme (λ', L) such that $\overline{P} = \alpha'_l$ we

have $U(P(x_1), \cdots, P(x_n)) = F$ iff it is possible to construct a state Q of the graph scheme (λ', L) such that PtQ and $\overline{Q} = \alpha_i'$; moreover $U(P(x_1), \cdots, P(x_n)) = T$ iff it is possible to construct a finite sequence of states $P_1, \cdots, P_t$ of the generalized graph scheme (λ', L) such that $P_1 \underset{(\lambda', L)}{\vdash} P_2 \vdash \cdots \cdots \underset{(\lambda', L)}{\vdash} P_t$; $P_1 = P$; $\overline{P}_t$ is a stop and $\overline{P}_t \neq \alpha_i'$ for any i such that $2 \leqslant i \leqslant t$.

We can say that the P-term R_1 recognizes for a given state P and a given algorithmic table L whether or not the graph scheme (λ', L) "performs" a cycle by operating over the state P, or whether it "drops out" the cycle.

Let every $G_i (1 \leqslant i \leqslant n)$ denotes the F-term $B_r(\overline{R}_1, T_i, T_i)$. Then it is easy to verify that the algorithms $V_i (1 \leqslant i \leqslant n)$ realized by the F-terms C_i in L have the property that for any finite non-empty sequence of states $P_1 \cdots, P_t$ of (λ', L) that satisfies the conditions

a) $P_1 \underset{(\lambda', L)}{\vdash} P_2 \underset{(\lambda', L)}{\vdash} \cdots \underset{(\lambda', L)}{\vdash} P_t \quad (t > 1)$, b) $\overline{P}_1 = \alpha_i'$,

c) $\overline{P}_t = \alpha_i'$ or $\overline{P}_t$ is a stop, d) $\overline{P}_t = \alpha_i'$ for $1 < i < t$,

we have $P_t(x_j) = V_j(P_1(x_1), \cdots, P_1(x_n)) \ (1 \leqslant j \leqslant n)$.

We can say that the F-terms $C_1, \cdots, C_n$ describe a transformation performed by the cycle contained in the generalized memory transformation scheme λ'.

Let us also construct F-terms $D_1, \cdots, D_n$ such that

$$D_i = \mathrm{Rep}\ (i, \overline{R}_1, C_1, \cdots, C_n) \ (1 \leqslant i \leqslant n).$$

Now we can construct a generalized memory transformation scheme λ^* as follows:

1) Let $\varepsilon_1, \cdots, \varepsilon_w$ be all the functor and predicate skeleton variables occurring in λ. Let us consider skeleton variables $\overset{*}{\varepsilon}_1, \overset{*}{\varepsilon}_2, \cdots \cdots, \overset{*}{\varepsilon}_w, \overset{**}{\varepsilon}_1, \cdots, \overset{**}{\varepsilon}_w$, that differ from one another and from all the variables ε_i, we shall select these variables in such a way that $\overset{*}{\varepsilon}_i$ and $\overset{**}{\varepsilon}_i$ are functor (predicate) variable if ε_i is a functor (predicate) variable. We shall also consider a functor skeleton variable η and predicate skeleton variable θ which are distinct from all the variables ε_i, $\overset{*}{\varepsilon}_i$, $\overset{**}{\varepsilon}_i$.

2) We construct skeleton terms $\overset{*}{R}_1, \overset{*}{R}_2, \cdots, \overset{*}{R}_d$ that can be obtained from $R_1, R_2, \cdots, R_d$ by replacing all the ε_i by $\overset{*}{\varepsilon}_i$.

3) We construct skeleton terms $S_1, \cdots, S_{k-1}$ by replacing all the ε_i by $\overset{*}{\varepsilon}_i$ and all stops by $\alpha_i^{**'}$ in each term S_i.

4) We construct the skeleton terms $\overset{*}{S}_k = (\alpha_k^{*'}, 0, \eta)$, $U = (\theta, \eta, \alpha_k^{**})$ and $V = (\eta, \alpha_i^{**'})$.

5) We construct skeleton terms S_l^{**}, $S_{l+1}^{**}, \cdots, S_{k-1}^{**}$ that can be obtained from $S_l^{*}, \cdots, S_{k-1}^{*}$ by replacing all the ε_l by ε_l^{**}.

6) We construct the skeleton term $S_k^{**} = (\alpha_k^{**}, \beta_k^{*}, \beta_k^{*})$.

7) We construct the skeleton scheme Σ^* of the generalized memory transformation scheme λ^* as follows:

$$\Sigma^* = (R_1^{*}, \cdots, R_d^{*}, S_l^{*}, \cdots, S_{k-1}^{*}, U, V, S_l^{**}, \cdots, S_{k-1}^{**}, S_k^{**}).$$

8) We construct the generalized memory transformation table Ξ^* of λ^* by adjoining to the generalized transformation table of λ, firstly all possible rows of the form $\varepsilon_i^{*} \rightarrow [T_1^{*}, \cdots, T_n^{*}]$ and $\varepsilon_i^{**} \rightarrow [T_1^{*}, \cdots, T_n^{*}]$ where $\varepsilon_i \rightarrow [T_1^{*}, \cdots, T_n^{*}]$ and secondly, the rows $\eta \rightarrow [D_1, \cdots, D_n]$ and $\theta \rightarrow \mathrm{Adj}\,(n-1, \cdots, \mathrm{Adj}\,(\theta, \mathbf{L}) \cdots)$.

9) We construct the generalized memory transformation scheme

$$\lambda^* = (\Sigma^*, \Xi^*, (x_1, \cdots, x_n), (x_1, \cdots, x_n)).$$

It is easy to see that λ^* is D-equivalent to λ and is a one-sided scheme without cycles.

It is evident that λ^* and λ satisfy all the conditions of theorem 1; and hence the generalized memory transformation scheme Ψ^* obtained as a result of the substitution of λ^* for subscheme λ in Ψ will be D-equivalent to Ψ. In accordance with the definition of substitution, the skeleton scheme of Ψ^* will be

$$(\overline{S}_1, \overline{S}_2, \cdots, \overline{S}_{l-1}, \overline{S}_{k+1}, \cdots, \overline{S}_h, \overline{S}_l^{*}, \overline{S}_{l+1}^{*}, \cdots, \overline{S}_{k-1}^{*}, \overline{S}_k^{*}, \overline{U}, \overline{V}, \overline{S}_l^{**}, \cdots$$
$$\cdots, \overline{S}_k^{**}),$$

where each symbol of the form $\overline{z}$ denotes a skeleton term z transformed in accordance with the operation of substitution of a generalized memory transformation scheme into a generalized memory transformation scheme. Let us permute the skeleton terms in the skeleton scheme of Ψ^* by arranging them in the folliwing order:

$$(\overline{S}_1, \overline{S}_2, \cdots, \overline{S}_{l-1}^{*}, \overline{S}_l^{*}, \cdots, \overline{S}_{k-1}^{*}, \overline{S}_k^{*}, \overline{U}, \overline{V}, \overline{S}_l^{**},$$
$$\overline{S}_{l+1}^{**}, \cdots, \overline{S}_{k-1}^{**}, \overline{S}_k^{**}, \overline{S}_{k+1}, \cdots, \overline{S}_h).$$

As a result we obtiain a generalized memory transformation scheme $\overline{\Psi}^*$ which is evidently D-equivalent to Ψ and is one-sided, and the inverse predicate skeleton variable in $\overline{\Psi}^*$ being by one smaller than in Ψ. By the inductive assumption it is possible to construct a one-sided generalized memory transformation scheme $\widetilde{\Psi}$ without cycles which is D-equivalent to Ψ. This completes the proof of the lemma.

Theorem 2: *For every strongly regular generalized functor or predicate memory transformation scheme* (Σ, Ξ, W_1, W_2) *of dimension* n *it is possible to construct a corresponding D-equivalent generalized*

memory transformation scheme (Σ, Ξ, W_1, W_2) *of dimension* t, *where it is the maximum index of the object variables in* $W_1 \cup W_2$, *such that*

$$\Sigma' = \begin{cases} ((H, \Pi_1), (\Pi, \omega_1, \omega_2)) & \text{if } (\Sigma, \Xi, W_1, W_2) \text{ is a generalized predicate memory transformation scheme,} \\ ((H, \Phi_1), (\Phi_1, \omega_1)) & \text{if } (\Sigma, \Xi, W_1, W_2) \text{ is a generalized functor memory transformation scheme,} \end{cases}$$

and Ξ' *consists of a sing.e row, and has only those functor and predicate variables which* (Σ, Ξ, W_1, W_2) *contains.*

Proof: Let Ψ be a generalized functor or predicate memory transformation scheme of dimension n. According to lemma 1 it is possible to construct a one-sided generalized memory transformation scheme Ψ_1 of the same dimension, D-equivalent to Ψ. Next, according to lemma 7 we can construct a one-sided generalized memory transformation scheme Ψ_2 of the same dimension as Ψ_1, without cycles, which is D-equivalent to Ψ. Since Ψ is a generalized functor or predicate memory transformation scheme and Ψ_2 is D-equivalent to Ψ, any superfluous stop which is produced in constructing Ψ_2 is replaced by the stop O_k of Ψ if Ψ is a generalized functor memory transformation scheme, or by a stop O_T if Ψ is a generalized predicate memory transformation scheme. Hence we obtain a one-sided pseudopredicate generalized memory transformation scheme Ψ_3 of dimension n, without cycles, D-equivalent to Ψ. By virtue of lemma 3 we can construct a primitive generalized memory transformation scheme Ψ_4 of dimension n D-equivalent to Ψ. Thus by lemma 5 or 6 we get the required generalized memory transformation scheme Ψ_5 of dimension t. This completes the proof of the theorem.

Lemma 8: For every strongly regular generalized functor memory transformation scheme Ψ of dimension n, with input variables $(x_{j_1}, \cdots, x_{j_t})(t \geqslant 0)$, it is possible to construct a strongly regular generalized functor memory transformation scheme Ψ' of dimension n, such that for every algorithmic table L in M matched with Ψ and Ψ' the generalized graph scheme (Ψ', L) specifies the same algorithmic functor as (Ψ, L) and the systems of input and output variables of Ψ' is $(x_1, \cdots, x_t)$ and (x_1) respectively.

Proof: Suppose $\Psi = (\Sigma, \Xi, W_1, W_2)$ is a generalized functor memory transformation scheme of dimension n, where $W_1 = (x_{j_1}, \cdots, x_{j_t})$ and W_2 is x_l.

Let us construct the required generalized functor memory transformation scheme of dimension n as follows:

1) Let Φ_1^*, Φ_2^* be functor skeleton variables which are not contained in Σ. Let us construct a skeleton scheme Σ' obtained from Σ by replacing the initial skeleton term (H, η) of Σ by (H, Φ_1^*), (Φ_1^*, η) and each terminal term (Φ_i, O_k) by (Φ_i, Φ_2^*), (Φ_2^*, O_k).

2) Let us construct a generalized memory transformation table Ξ' obtained from Ξ by adjoining to it the rows;

$$\Phi_1^* \leftarrow\!\!\rightarrow [I_n^1, I_n^2, \cdots, I_n^{j_1-1}, I_n^{j_1}, I_n^{j_1+1}, I_n^{j_1+2}, \cdots, I_n^{j_2-1}, I_n^{j_2}, I_n^{j_2+1}, I_n^{j_2+2}, \cdots, I_n^{j_t-1}, I_n^{j_t}, I_n^{j_t+1}, I_n^{j_t+2}, \cdots, I_n^n],$$ and

$$\Phi_2^* \leftarrow\!\!\rightarrow [I_n^1, I_n^2, \cdots, I_n^n].$$

3) Let W_1' is $(x_1, \cdots, x_t)$ and W_2' is (x_1).

Then the required generalized functor memory transformation scheme is $(\Sigma, \Xi', W_1', W_2')$. It is evident that (Ψ', L) specifies the same functor as (Ψ, L).

Lemma 9: For every strongly regular generalized predicate memory transformation scheme Ψ of dimension n, it is possible to construct a strongly regular generalized predicate memory transformation scheme Ψ' of dimension n such that for every algorithmic table L in M matched with Ψ and Ψ' the generalized predicate graph scheme (Ψ', L) specifies the same predicate as (Ψ, L) and the system of input variables in Ψ' is $(x_1, \cdots, x_t)$, where t $(t \geqslant 0)$ is the number of input variables in Ψ.

Proof: Suppose $\Psi = (\Sigma, \Xi, W_1, W_2)$ is a generalized predicate memory transformation scheme of dimension n, where $W_1 = (x_{j_1}, \cdots, x_{j_t})$ and W_2 is empty.

Let us construct the required generalised memory transformation scheme as follows:

(i) Let Φ_1^*, be a functor skeleton variable which is not contained in Σ. Let us construct a skeleton scheme Σ' from Σ by replacing the initial skeleton term (H, η) of Σ by (H, Φ_1^*), (Φ_1^*, η).

(ii) Let us construct a generalized memory transformation table Ξ' from Ξ by adjoining to it the row

$$\Phi_1^* \leftarrow\!\!\rightarrow [I_n^1, I_n^2, \cdots, I_n^{j_1-1}, I_n^{j_1}, I_n^{j_1+1}, I_n^{j_1+2}, \cdots, I_n^{j_2-1}, I_n^{j_2}, I_n^{j_2+1}, I_n^{j_2+2}, \cdots, I_n^{j_t-1}, I_n^{j_t}, I_n^{j_t+1}, I_n^{j_t+2}, \cdots, I_n^n].$$

(iii) Let W_1' is $(x_1, \cdots, x_t)$ and W_2' is ( ).

Then the required generalized predicate memory transformation scheme is $(\Sigma', \Xi', W_1', W_2')$. It is evident that (Ψ', L) specifies the same predicate as (Ψ, L).

Theorem 3: *For every strongly regular generalized functor (predicate) memory transformation scheme Ψ of dimension n it is possible to construct an F-term (P-term) T such that (i) for every algorithmic table L in M matched with Ψ, the term T realizes with respect to L the same functor (predicate) which is specified by (Ψ, L) and (ii) every functor or predicate variable contained in T is also contained in Ψ.*

Proof: Case (i) Suppose $\Psi = (\Sigma, \Xi, W_1, W_2)$ is a strongly regular generalized functor memory transformation scheme of dimension n. According to lemma 8, it is possible to construct a strongly regular generalized functor transformation scheme $\Psi_1 = (\Sigma', \Xi', W_1^*, W_2^*)$ of dimension n, where W_1^* and W_2^* are $(x_1, \cdots, x_t)$ and (x_1) respectively, and t is the number of variables in W_1 such that for every algorithmic table L in M matched with Ψ and Ψ_1, (Ψ_1, L) specifies the same functor as (Ψ, L). Next according to theorem 2, we can construct a generalized memory transformation scheme of dimension t_1 D-equivalent to Ψ_1 and having the form $(((H, \Phi_1), (\Phi_1, \omega_1)), (\Phi_1 \leftarrow [T_1, \cdots, T_{t_1}])\ W_1^*, W_2^*)$, where t_1 is the number of variables in $W_1^* \cup W_2^*$.

The required F-term is Br (Comp (Adj $(0, \cdots,$ Adj $(0,$ T)$\cdots)$, $T_1, \cdots, T_t)$, T_1, $T_1)$, if W_1 is non-empty and Comp (T_1, R) if W_1 is empty, where R is a functor algorithmic variable of dimension zero if is contained is Ψ, otherwise R is D.

Case (ii): Suppose $\Psi = (\Sigma, \Xi, W_1, W_2)$ is a strongly regular generalized predicate memory transformation scheme of dimension n. According to lemma 9 it is possible to construct a strongly regular generalized predicate transformation scheme $\Psi_1 = (\Sigma_1', \Xi', W_1^*, W_2^*)$ of dimension n where W_1^* is $(x_1, \cdots, x_t)$, and $t \geqslant 0$ is the number of object variables in W_1 such that for every algorithmic table L in M matched with Ψ and Ψ_1 (Ψ_1, L) specifies the same predicate as (Ψ, L). Next according to theorem 2 we can construct a strongly regular generalized predicate transformation scheme Ψ_2 of dimension t, D-equivalent to Ψ_1 and having the form $(((H, \pi_2), (\pi_1, \omega_1, \omega_2)), (\pi_1 \leftarrow R), W_1, W_2^*)$. Then the required P-term of dimension t is R-

This complets the proof of the theorem.

Def. 17. The depth of an F-term or P-terms is defined inductively as follows:

1) Every F-term of the form $f_{2^n(2l+1)}$, D or I has the depth zero.

2) Every P-term of the form $P_{2^n(2l+1)}$, T or L has the depth zero.

3) Every F-term or P-term of the form Comp $(T_1, \cdots, T_m)$ has $\sum_{k=1}^{m} t_k + 1$ depth, where $t_k (1 \leqslant k \leqslant m)$ is the depth of T_k.

4) Every F-term or P-term of the form Br (T_1, T_2, T_3) has $\sum_{k=1}^{3} t_k + 1$ depth, where t_k $(1 \leqslant k \leqslant 3)$ is the depth of T_k.

5) Every F-term of P-term of the form Adj (i, T) has the same depth as T.

6) Every F-term of the form Rep $(i, T_1, \cdots, T_m)$ has $\sum_{k=1}^{m} t_k + 1$ depth, where t_k $(1 \leqslant k \leqslant m)$ is the depth of T_k.

Def 18. The depth of an A-term $[T_1, \cdots, T_n]$ is defined as max $(t_1, \cdots, t_n)$ where $t_i (1 \leqslant i \leqslant n)$ is the depth of the F-term T_i.

Def. 19. The depth of a generalized memory transformation scheme is said to be k ($k>0$) iff these exists an A-term or P-term in Ψ whose depth is k and every A-term or P-term in Ψ has a depth m such that $m \leqslant k$.

Theorem 4: *For every strongly regular generalized memory transformation scheme Ψ of dimension n it is always possible to construct a generalized memory transformation scheme Ψ' of dimension m ($m \geqslant n$) and depth zero, which is D-equivalent to Ψ.*

Proof: Let $\Psi = (\Sigma, \Xi, W_1, W_2,)$ be a strongly regular generalized memory trasformation scheme of dimension n with depth k. The proof is by induction on the depth of Ψ. If $k = 0$, then Ψ' is Ψ.

Now let us assume that $k > 0$. For every skeleton variable α in the skeleton scheme Σ of Ψ such that $(\alpha \leftarrow\!\!\rightarrow B) \in \Xi$, B is P-term or A-term of depth k in Ψ, let us construct a subscheme λ of dimension n with depth k generated by $\{\alpha\}$. Let us construct a generalized memory transformation scheme λ^* of dimension m with the same input and output variable as λ with depth k_1 ($k_1 < k$), D-equivalent to λ as follows:

Case 1: Suppose α is a predicate skeleton variable, then without loss of generality let us assume that Ψ has the form $(((\nu_1, \alpha), (\nu_1, \alpha), \cdots, (\nu_t, \alpha), (\alpha, \omega_1, \omega_2)), (\alpha \leftarrow\!\!\rightarrow B), (x_1, \cdots, x_n), (x_2, \cdots, x_n))$.

Let us specify predicate skeleton variables π_1, π_2, π_3 and a functor skeleton variable Φ_1, that differ from one another and from all the skeleton variables of Ψ and λ.

Les R be a P-term defined as follows:

$$R = \begin{cases} B' & \text{if } B = \mathrm{Adj}\,(i_1, \cdots, \mathrm{Adj}(i_l, B')\cdots) \text{ where } l < n-1 \text{ and } B \\ & \text{does not have the form } \mathrm{Adj}\,(i_{l+1}, B''); \\ B & \text{otherwise.} \end{cases}$$

Now let us construct the required generalized memory transformation scheme λ^* as follows:

(i) If R has the form $Br\,(T_1, T_2, T_3)$, then $\lambda^* = (((v_1, \pi_1), (v_2, \pi_1), \cdots, (v_t, \pi_1), (\pi_1, \pi_2, \pi_3), (\pi_2, \omega_1, \omega_2), (\pi_3, \omega_1, \omega_2)), (\pi_1 \leftarrow\!\!\rightarrow T_2^*, \pi_2 \leftarrow\!\!\rightarrow T_2^*, \pi_3 \leftarrow\!\!\rightarrow T_3^*), (x_1, \cdots, x_n), (x_1, \cdots, x_n))$, where T_i^* ($1 \leqslant i \leqslant 3$) is obtained from T_i by adjoining all those object variables from $x_1, \cdots, x_n$ which T_i does not contain.

(ii) If R has the form $\mathrm{Comp}\,(T_1, T_2, \cdots, T_m)$, then $\lambda^* = (((v_1, \Phi), (v_2, \Phi_1), \cdots, (v_t, \Phi_1), \Phi_1, \pi_1), (\pi_1, \omega_1, \omega_2)), (\pi_1 \leftarrow\!\!\rightarrow T_1^*, \Phi_1 \leftarrow\!\!\rightarrow [L_1, L_2, \cdots, L_s]), (x_1, \cdots, x_n), (x_1, \cdots, x_n))$, here s is the number of all distinct object variables in $T_1, T_2, \cdots, T_m$ and $x_1, \cdots, x_n$, and every L_i ($1 \leqslant i \leqslant s$) is obtained from T_i, by adjoining all the object variables from $x_1, \cdots, x_s$ which T_j does not contain, if x_i is the $(j-1)-th$ position of T_1, otherwise l_i is l_s; and T_1^* if obtained from T_1 by adjoining all those object variables from $x_1, \cdots, x_s$ which T_1 does not contain.

It is easy to see that the constructed generalized memory transformation scheme λ^* has the depth $k_1 < k$ and is D-equivalent to λ.

Case 2: Suppose α is a functor skeleton variable. Then without loss of generality we can assume that the subscheme λ has the form $(((v_1, \alpha), (v_2, \alpha), \cdots, (v_t, \alpha), (\alpha, \omega_1)), (\alpha \leftarrow [T_1, \cdots, T_n]), (x_1, \cdots, x_n), (x_1, \cdots, x_n))$. Since B is an A-term of depth k, we can assume without loss of generality that $T_{r_1}, \cdots, T_{r_p}$ are all the F-terms in $[T_1, \cdots, T_n]$ which have depth k.

1) Let us specify functor skeleton variables $\Phi_1, \cdots, \Phi_{n+1}$ that differ from one another and from all the skeleton variables λ and Ψ.

2) Let us construct a generalized memory transformation scheme $\bar{\lambda}$ with dimension $2n$ and depth k as follows: $\bar{\lambda} = (((v_1, \Phi_1), \cdots, (v_t, \Phi_1), (\Phi_1, \Phi_2), \cdots, (\Phi_n, \Phi_{n+1}), (\Phi_{n+1}, \omega_1)), (\Phi_1 \leftarrow [I'_{2n}, \cdots, I^n_{2n}, T^*_1, I^{n+2}_{2n}, \cdots, I^{2n}_{2n}], \cdots, \Phi_t \leftarrow [I'_{2n}, \cdots, I^{n+t-1}_{2n}, T^*_t, I^{n+t+1}_{2n}, \cdots, I^{2n}_{2n}], \cdots, \Phi_n \leftarrow [I'_{2n}, \cdots, I^{2n-1}_{2n}, T^*_n], \Phi_{n+1} \leftarrow [I^{n+1}_{2n}, I^{n+2}_{2n}, \cdots, I^{2n}_{2n}, I'_{2n}, \cdots, I^n_{2n}]), (x_1, \cdots, x_n), x_1, \cdots, x_n))$, where for every i $(1 \leqslant i \leqslant n)$ $T^*_i = \mathrm{Adj}(2n-1, \mathrm{Adj}(2n-2, \cdots, \mathrm{Adj}(n, T_i)) \cdots)$.

It is obvious that $\bar{\lambda}$ is D-equivalent to λ.

3) Let us construct subschemes $\bar{\lambda}_{r_i}$ $(1 \leqslant i \leqslant p)$ from $\bar{\lambda}$ generated by $\{\Phi_{r_i}\}$. Then every $\bar{\lambda}_{r_i}$ will have the form $(((v'_{r_i}, \Phi_{r_i}), \Phi_{r_i}, \omega'_{r_i})), (\Phi_{r_i} \leftarrow [I'_{2n}, \cdots, I^{n+t-1}_{2n}, T^*_t, I^{n+t+1}_{2n}, \cdots, I^{2n}_{2n}], (x_1, \cdots, x_n), (x_1, \cdots x_n))$ (except $\tilde{\lambda}_1$ which contains t initial skeleton terms), where v'_{r_i} and ω'_{r_i} $(1 \leqslant i \leqslant p)$ are respectively initial and terminal skeleton variables which differ from one another and from all the initial and terminal skeleton variables of $\bar{\lambda}$.

4) We define F-terms

$$\bar{L}_{r_i} = \begin{cases} L' & \text{if } T^*_{r_i} = \mathrm{Adj}(s_1, \mathrm{Adj}(s_2, \cdots, \mathrm{Adj}(s_c, L')) \cdots) \text{ where} \\ & c+1 < 2n \text{ and } L' \text{ does not have the form } \mathrm{Adj}(s_{c+1}, L''), \\ T^*_{r_i} & \text{otherwise,} \end{cases}$$

where $(1 \leqslant i \leqslant p)$.

5) We specify functor skeleton variables $\varphi^{(r_i)}_m, \cdots, \varphi^{(r_i)}_1$ $(m > n)$ and predicate skeleton variables $\pi^{(r_i)}_1$ that differ from one another and from $\bar{\lambda}$ and $\tilde{\lambda}_{r_i}$. Let us construct generalized memory transformation scheme $\tilde{\tilde{\lambda}}_{r_i}$ $(1 \leqslant i \leqslant p)$ as follows:

(i) If $\bar{L}_{r_i}$ has the form $\mathrm{Comp}(s_1, \cdots, s_m)$ then $\tilde{\tilde{\lambda}}_{r_i} = (((v'_{r_i}, \varphi^{(r_i)}_1), (\varphi^{(r_i)}_1, \varphi^{(r_i)}_2), \cdots, (\varphi^{(r_i)}_{m-1}, \varphi^{(r_i)}_m), (\varphi^{(r_i)}_m, \omega'_{r_i})), \varphi^{(r_i)}_1 \leftarrow [L^{(1)}_1, \cdots, L^{(1)}_s], \cdots$

$\cdots, \varphi_m^{(r_l)} \leftarrow [L_1^{(m)}, \cdots, L_s^{(m)}])$, $(x_1, \cdots, x_n)$, $(x_1, \cdots, x_n))$, where s is the maximum index of the object variables in S_1 and $x_1, \cdots, x_{2n}$; $L_{r_l}^{(m)}$ is obtained from S_1 by adjoining all those variables from $x_1, \cdots, x_s$ which s_1 does not contain, and every $L_l^{(j)}$ $((1 \leqslant i \leqslant s), (1 \leqslant j \leqslant m))$ is obtained from S_{l+1} by adjoining all those variables from $x_1, \cdots, x_s$ which S_{j+1} does not contain, such that x_l is in the j^{th} position in S_1, otherwise $L_l^{(j)}$ is I_s^l.

(ii) If $\bar{L}_l$ has she form $Br\ (S_1, S_2, S_3)$, then $\tilde{\lambda}'_{r_l} = (((\nu'_{r_l}, \pi_1^{(r_l)}), (\pi_1^{(r_l)}, \varphi_1^{(l)}, \varphi_2^{(r_l)}), (\varphi_1^{(r_l)}, \omega'_{r_l}), (\varphi_2^{(r_l)}, \omega'_{r_l})), (\pi_1^{(r_l)} \leftarrow S_1^*, \varphi_1^{(r_l)} \leftarrow [L_1^{(j)}, \cdots$ $\cdots, L_{2n}^{(j)}], \varphi_2^{(r_l} \leftarrow [L_1^{(2)}, \cdots, L_{2n}^{(1)}]), (x_1, \cdots, x_n), (x, \cdots, x_n))$, where S_1^*, $L_{r_l}^{(1)}$ and $L_{r_l}^{(2)}$ are obtained respectively from S_1, S_2 and S_3 and adjoining all those object variables from $x_1, \cdots, x_{2n}$ which S_1, S_2 and S_3 does not contain and every other $L_k^{(j)}$ $((1 \leqslant j \leqslant 2), (1 \leqslant k \leqslant 2n))$ has the form I_{2n}^k.

(iii) If $\bar{L}_{r_l}$ has the form $\mathrm{Rep}\ (c, S_1, \cdots, S_t)$ $(1 < t \leqslant n+1)$ then $\bar{\lambda}'_{r_l} = (((\nu'_{r_l}, \pi_1^{(r_l)}), (\pi_1^{(r_l)}, \omega'_{r_l}, \varphi_1^{(r_l)}), (\varphi_1^{(r_l)}, \pi_1^{(r_l)})), (\pi_1^{(r_l)} \leftarrow S_1^*, \varphi_1^{(r_l)} \leftarrow$ $\leftarrow [S_2', \cdots, S_{2n+1}^*]), (x_1, \cdots, x_n), x_1, \cdots, x_n))$, where every S_k^* $(t < k \leqslant 2n+1)$ is I_{2n}^{k-1} and every S_k^* $(1 \leqslant k \leqslant t)$ is obtained from S_k by adjoining all those object variables from $x_1, \cdots, x_{2n}$ which S_k does not contain.

It is easy to see that the generalized memory transformation schemes $\bar{\lambda}_{r_l}$ and $\bar{\lambda}'_{r_l}$ $(1 \leqslant i \leqslant p)$ are D-equivalent. By substituting every $\tilde{\lambda}_{r_l}$ $(1 \leqslant i < p)$ in $\bar{\lambda}$ for the subscheme $\bar{\lambda}_{r_l}$ we obtain the required generalized memory transformation scheme λ^* with depth $k_1 < k$ and dimension $q \geqslant m$.

For every subscheme λ in Ψ with depth k, we substitute its D-equivalent generalized memory transformation scheme λ^* in Ψ. We obtain a generalized memory transformation scheme Ψ_1, of depth k_1 $(k_1 < k)$ and dimension q D-equivalent to Ψ. By repeating the same constructions for Ψ_1, we construct a generalized transformation scheme Ψ_2 with depth k_2 $(k_2 < k_1)$. By continuing this process h times $(h \leqslant k)$ we get our required generalized memory transformation scheme with depth zero and dimension m $(m > n)$, which is D-equivalens to the original generalized memory transformation scheme Ψ. This completes the proof of the theorem.

American University of Beirut,
Yerevan State University

Received 15.V.1974

Տ. Ա. ԷԴԻՆՋԻԿԼՅԱՆ. Հիշողություն ունեցող գրաֆ-սխեմաների միջոցով սահմանված ալգորիթմների որոշ ֆորմալ պատկերացումների մասին (ամփոփում)

Դիտարկվում են հիշողություն ունեցող գրաֆ-սխեմաների միջոցով որոշված ալգորիթմների պատկերացումները բազմատեղանի ֆակտորների ու պրեդիկատների կոմպոզիցիայի, ճյուղավորման և կրկնման գործողությունների վրա հիմնված որոշ հանրահաշվական լեզվի միջոցով։ Ապացուցվում է թեորեմա այն մասին, որ ամեն մի ալգորիթմի համար, որը ֆորմալ նկարագրված է հիշողություն ունեցող ինչ-որ մի (ոչ մեկնաբանված) գրաֆ-սխեմայի միջոցով, միշտ հնարավոր է կառուցել նրան պատկերացնող արտահայտությունը՝ նշված լեզվի մեջ։

Այս թեորեմայի ապացուցման համար սահմանվում է ընդհանրացված գրաֆ-սխեմայի դաղափարը և հիմնավորվում են այսպիսի գրաֆ-սխեմաների պատկերացման հնարավորությունները որոշ ստանդարտ ձևերում։

Т. А. ЭДИНДЖИКЛЯН. *О некоторых формальных представлениях алгорифмов, определяемых граф-схемами с памятью* (резюме)

Рассматриваются представления алгорифмов, определяемых граф-схемами с памятью, в алгебраическом языке, построенном на основе операций композиции, разветвления и повторения многоместных функторов и предикатов. Доказывается теорема о том, что для всякого алгорифма, формально описанного некоторой (неинтерпретированной) граф-схемой с памятью, возможно построить представляющее его выражение в указанном языке. Для доказательства этой теоремы вводится понятие обобщенной граф-схемы и устанавливается возможность представления таких граф-схем в определенных стандартных формах.

REFERENCES

1. *E. Ashcroft* and *Z. Manna*. The translation of „GOTO" Programs to „WHILE" Programs, Information Processing '71 North Holland publ. company 1972, 150—155.
2. *A. P. Ershov*. Operator algorithms I. Basic notions, Problemy Kibernet, № 3, 1960, 5—48 (Russian).
3. *A. P. Ershov*. Operator algorithms. II. (A description of the fundamental constructions of programming), Problemy Kibernet, № 8, 1962, 211—233 (Russian).
4. *A. P. Ershov*. Operator algorithms. III. (On Ianov's operator-schemes) Problemy Kibernet., № 20, 1968, 191—200 (Russian).
5. *Ju. I. Ianov*. On logical schemes of algorithms, Poblemy Kibernet, № 8, 1962. 235—241 (Russian).
6. *V. E. Itkin*. Logico-Termal equivalence of program schemes, Kibernet, № 1, 1972, 5—27 (Russian).
7. *L. A. Kaluznin*. On the algorithmisation of mathematical problems, Problemy Kibernet., № 2, 1959, 51—67 (Russian).
8. *A. A. Ljapunov*. Logical program-schemes, Problemy Kibernet., № 1, 1958, 46—74 (Russian).
9. *A. A. Ljapunov*. The algebraic treatment of programming Problemy Kibernet., № 8, 1962, 235—241 (Russian).
10. *D. C. Luckham, D. M. R. Park*, and *H. S. Paterson*. On formalized computer programs, Computer and System Sciences, June 1970.
11. *R. I. Podlovchenko*. On a system of concepts of programming, Dokl. Akad. Nauk SSSR, 132, 1960, 1287—1290 (Russian).
12. *R. I. Podlovchenko*. Transformations of program schemes and their use in programming, Problemy Kibernet., № 7, 1962, 161—188 (Russian).

13. *R. I. Podlovchenko, G . N. Petrosstan, V. E. Khatchatrtan.* The interpretation of algorithmic schemes and different relations of equivalence between the schemes, Izvestia AN Arm SSR, series Mathematica, vol. VII, № 2, 1972, 140—151 (Russian).

14. *J. D. Rutledge.* On Ianov's program schemata, J. Assoc. Comp. Mach., 11, 1964, 1—9.

15. *I. D. Zaslavskll.* Graph schemes with memory, translation of Trudy Math. Inst. Steklov, 72, 1964, 99—192. (Amer. Math. Soc. Trans. (2), vol. 98, 1971).