THE EXPONENTIAL STORAGE COST OF D-SCHEMES

by

Ralph TINDELL

A 76/1o XI/1976

Abstract :

Structured programming has been studied recently in the context of program schemes. It is in this setting that we wish to examine the question of the "inefficiency" of structured programs. In particular, we study the "intrinsic size" of structured program schemes when compared to equivalent nonstructured schemes. The notion of equivalence used is the one requiring equivalent schemes to compute the same function for each interpretation of their common operator and predicate symbols.

To study the "intrinsic size" of a structured scheme, we examine the size of a smallest equivalent structured scheme, and compare this with the size of a smallest equivalent nonstructured scheme.

The general class of schemes studied in the present paper is the class of Ianov schemes, and the "structured" schemes considered

are the so-called Dijkstra schemes. The primary result is, from some points of view, a negative one :

the intrinsic size of Dijkstra schemes may be exorbitant.

To be precise, we construct a sequence F_n of Dijkstra schemes such that for each n, no smaller Dijkstra scheme is equivalent to F_n, and the number of edges in F_n grows exponentially. We then show there are weak equivalent nonstructured schemes G_n whose size grows only linearly.

Introduction

Flowchart schemes have been studied in recent times by many authors, with the schemes modelled after structured programs recieving a great deal of attention. Essentially, a flowchart scheme is a flowchart in which actual functions and predicates have been replaced by appropriate uninterpreted symbols. Then, by allowing a variety of interpretations of these symbols as functions and predicates over various sets one obtains a class of structurally similar flowchart algorithms, each of which computes a partial function on the domain of the interpretation. A notion of equivalence widely considered is that of weak equivalence, for which one requires that the two schemes compute the same partial function for each specific interpretation of the symbols.

In the present paper we consider monadic schemes with one input and one output, and study simple while schemes [3], or D-schemes [2] built using composition, if-then-else, and while-do operations. In particular, we are interested in the size of smallest D-schemes relative to smallest "nonstructured" schemes within weak equivalence classes. The major result of the paper is the construction of weak equivalent schemes F_n and G_n, with F_n a "smallest" D-scheme, such that the size of F_n grows exponentially with n, while the size of G_n grows linearly. These examples may be viewed as a theoretical verification of the opinion held by many that structured programs tend to be very large.

Definitions and Notation

Let $\Gamma=(\Omega,\Pi)$ be an ordered pair of disjoint sets; the elements of Ω are called operator symbols and those of Π, predicate symbols. A Γ-flowchart scheme F is a (locally) ordered directed graph F with a distinguished begin vertex b, a distinguished exit vertex e of outdegree 0, and a labelling function assigning to each outdegree 1 vertex an operator symbol and to each outdegree 2 vertex a predicate symbol. We assume there are no vertices of outdegree 3 or more, and that the exit e is the only vertex of outdegree 0. "Locally ordered" may in this context be taken to mean that the two outedges of each predicate vertex have been labelled in a nonrepetitive way by ϕ("false") and τ("true").

An interpretation of Γ over a set X is an assigment to each $f\in\Omega$ a function $I(f):X\to X$ and to each $\pi\in\Pi$ a predicate $I(\pi):X\to\{\phi,\tau\}$. Given an interpretation I, a Γ-scheme F determines, in a fashion we assume clear, a partial function $\Gamma(I):X\to X$. Two Γ-schemes F and G are said to be w-equivalent (Elgot [2], Kosaraju [5]) if they compute the same partial function under any interpretation; that is, for any interpretation I, F(I) = G(I). We shall later define a set of words determined by F, called the weak behaviour |F| with the property that F is w-equivalent to G iff |F| = |G|. We shall utilize this latter formulation in our proofs.

Certain schemes are singled out by their simplicity. A scheme with one vertex and no exit is called a trivial scheme, and a

scheme with two vertices, begin and exit, and a single edge from the begin to the exit is called an atomic scheme. There is an obvious notion of isomorphism of schemes and we henceforth identify isomorphic schemes. Thus, there is now a unique trivial scheme, henceforth denoted T; and for each f in Ω, a unique atomic scheme with begin labelled by f. We shall denote this latter atomic scheme by its operator label f.

The elementary structured programming operations of concatenation, if-then-else, and while-do may be formulated as operations on schemes. More precisely, given schemes F and G, and a predicate symbol π, we define the following schemes. F•G is the scheme obtained from the disjoint union of F and G by identifying the exit of F with the begin of G. F:π:G is the scheme obtained from the disjoint union by identifying the two exits and adding a new begin vertex labelled by π, with ϕ-edge directed to the begin of F and the τ-edge to that of G. π[F] is the scheme obtained from F by adding a new begin labelled π, having its ϕ-edge directed to a new exit and its τ-edge to the begin of F, and then identifying the exit of F with the new begin. One similary defines the scheme $\tilde{\pi}$[F] modelled after "while(not π)do F". A D-scheme is defined to be a scheme constructed using only trivial and atomic schemes and the four types of operations given above. That is, the class $\mathfrak{D}$ of D-schemes is the smallest class containing T and all atomic schemes and closed under all the D-operations. Some simple D-schemes are given in figure 1.

Contracted Schemes and Weak Behaviour

Given a Γ-scheme F we now define an associated labelled directed graph F' called the contraction of F. To do so, we need some preliminary definitions. First we must order the predicate vertices: $\Gamma=\{\pi_1,\ldots,\pi_r\}$. Now, given a nonexit vertex u of F and a string $t_1\ldots t_r\in\{\phi,\tau\}^r$, we define a path initiating at u as follows. Beginning at u, traverse the unique outedge if u is an operator vertex, and traverse the outedge labelled t_j if u is labelled by π_j. From this point on, the path terminates if an operator vertex is encountered; it continues along the edge labelled t_k if we have entered a vertex labelled π_k. The above process defines a path we denote by $P[u;t_1\ldots t_r]$, which either terminates at an operator vertex, at the exit, or is an infinite path.

We may now define the contraction F' of a scheme F. The vertices of F' are the operator nodes of F, which retain their operator labels, together with the exit e of F and a new vertex b' called the begin of F'. We now specify the directed edges of F', each of which recieves a unique label from $\{\phi,\tau\}^r$. If the begin b of the scheme F is an operator vertex or the exit (F=T) then all edges in F' from b' are directed to b, and there is one such for every possible label $t_1\ldots t_r$. If, on the other hand, b is a predicate vertex in F there is to be an edge in F', labelled $t_1\ldots t_r$, from b' to a vertex v iff the path $P[b;t_1\ldots t_r]$ in F terminates at v. The edges originating at an operator vertex u

are specified similary : there is an edge with label $t_1t_2...t_r$ in F' from u to v iff the path $P[u;t_1...t_r]$ in F terminates at v. The description of F' is now complete. By way of example, the D-scheme $F_1=(f\cdot(T:\pi_2:g)):\pi_1:g$ and its contraction F'_1 are depicted in figure 2.

The weak behaviour $|F|$ of a scheme F is determined from its contraction F' as follows. $|F|$ is to be a set of words of the form $w=\alpha_0f_1\alpha_1f_2...f_k\alpha_k$ where each f_i is in Ω and each α_i is in $\{\phi,\tau\}^r$. The words in $|F|$ are to be those which are the "traces" of "successful" paths Q in F'. If we as usual view a path as an alternating sequence $Q=v_0e_0v_1...e_kv_{k+1}$ of vertices and edges, then it is successful if $v_0=b'$ and v_{k+1} is the exit e; the word w above is the trace of Q if, for each $i\geq o$, α_i is the label in F' of the edge e_i and, for $1\leq i\leq k$, f_i is the label of the vertex v_i. Referring to figure 2, we see that the weak behaviour of F_1 is given by :

$$|F| = [\phi]_1f([\phi]_2\cup[\tau]_2g\{\phi,\tau\}^r)\cup[\tau]_1g\{\phi,\tau\}^r$$

where the symbol $[t]_j$ denotes the set of all strings $t_1...t_r$ in $\{\phi,\tau\}^r$ satisfying $t_j=t$.

Our definition of contracted scheme and weak behaviour is taken directly from Elgot [2], who points out that by Rutledge [7] and Luckham, Park and Paterson [6], it follows that schemes F and G are w-equivalent iff $|F| = |G|$.

We close this section with several useful lemmas. First, we extend the definition of $P[u;t_1 \ldots t_r]$ given above to show how an initial segment $w=\alpha_0 f_1 \ldots f_i \alpha_i$ of a word $\alpha_0 f_1 \ldots f_i \alpha_i \ldots f_k \alpha_k$ in $|F|$ uniquely determines a path in F starting at the begin b of F. If b is an operator vertex or the exit (in case F=T), define P_0 to be the constant path from b to itself; otherwise set $P_0=P[b;\alpha_0]$. If $i>0$ and v_1 is the terminal point of P_0, define $P_2=P[v_1;\alpha_1]$. Continuing in this fashion, we have paths $P_0, P_1, \ldots, P_i$ and we define the path P(w) to be the composition of these paths: $P(w)=P_0 P_1 \ldots P_i$. It is clear from the definition that P(w) terminates at the exit iff w is in $|F|$. Since a path terminating at the exit may not be extended, we may deduce the following "initial segment lemma".

LEMMA 1 :

No proper initial segment of a word in $|F|$ is itself in $|F|$.

It is convenient to define a special product on words of the type found in weak behaviours, exactly analogous to composition of paths. Given $w=\alpha_0 f_1 \ldots f_i \alpha_i$ and $z=\beta_0 g_1 \ldots g_j \beta_j$ their composite $w \cdot z$ is undefined if $\alpha_i \neq \beta_0$ and otherwise is given by $w \cdot z = \alpha_0 f_1 \ldots f_i \alpha_i g_1 \ldots g_j \beta_j$. This definition extends in the usual way to sets of words of the appropriate type and the definition is so constructed that the following are easily established.

LEMMA 2 :

If F and G are schemes, $|F \cdot G| = |F| \cdot |G|$.

LEMMA 3 :

If F and G are schemes and $\pi_j \in \Pi$, then

$$|F:\pi_j:G| = [\phi]_j \circ |F| \cup [\tau]_j \circ |G|.$$

For the final lemmas of the section, we introduce two more notions. If $w=\alpha_o f_1 \ldots f_i \alpha_i$, and F is a scheme, then $L_w(|F|) = \{z \mid w \cdot z \text{ is in } |F|\}$. Next, given a vertex v of F, define the flow from v in F, denoted F<v,e>, to be the labelled subdigraph of F consisting of the vertices and edges lying on paths initiating at v. If the exit e is in F<v,e>, then we may, by designating v as the begin of F<v,e>, turn F<v,e> into a scheme. An example where this is possible would be where v is the terminal vertex of the path determined by an initial segment of a word in |F|. It is a straightforward matter to deduce the next result.

LEMMA 4 :

If v is the terminal vertex of the path in F determined by an initial segment w of a word in |F|, then $L_w(|F|) = |F<v,e>|$.

The final lemma of the section is a direct corollary of the characterization theorem for D-schemes given in [8].

LEMMA 5 :

If vertex v of a D-scheme F lies on no cycle in F, then F<v,e> is itself a D-scheme.

An Exponentially Inefficient Sequence of D-Schemes

We now come to the construction of the schemes promised in the introduction, and the verification of their properties. We define the sequence of D-schemes inductively by setting $F_0=T$ and for $n\geq o$, setting $F_{n+1} = (f\bullet(T:\pi_2:g\bullet F_n)):\pi_1:(g\bullet F_n)$. We make precise the notation of "smallest" D-scheme in the obvious way : a D-scheme F is w-minimal over $\mathfrak{D}$ if no weak equivalent D-scheme has fewer edges than F. F_{n+1} is pictured in figure 3.

Our first, and only difficult task is to show that each of the D-schemes just constructed is w-minimal over $\mathfrak{D}$. First we give the recursive calculation of their weak behaviours :
$|F_0| = |T| = \{\phi,\tau\}^r$ and $|F_{n+1}| = ([\phi]_1 f([\phi]_2 U[\tau]_2 g|F_n|)U([\tau]_1 g|F_n|)$.

The scheme F_1 of the present sequence is the same as the scheme depicted in figure 2. It will be notationally simpler for us to presume $\Pi=\{\pi_1,\pi_2\}$; however, subsequent arguments, with only minor modifications, are valid even if Ω and Π are infinite. As a preliminary to showing w-minimality, we examine forced properties on D-schemes w-equivalent to the D-schemes under consideration.

LEMMA 6 :

If H is a D-scheme w-equivalent to F_{n+1}, $n\geq o$, then the begin of H is a predicate vertex, and H is not of either of the forms $\pi[G]$ or $\tilde{\pi}[G]$.

Proof :

The opposite of the second claim may be ruled out since $|F_{n+1}| \cap \{\phi,\tau\}^2$ is empty; and that of the first by the fact that $|F_{n+1}|$ may not be written as $\{\phi,\tau\}^2 hL$ for any h in Ω and any set L.

The next lemma follows by a straight forward induction on n.

LEMMA 7 :

If $o \leq l \leq n$, the following are subsets of $|F_{n+1}|$:

(7.1) $([\tau]_1 g)^{n+1} \{\phi,\tau\}^2$;

(7.2) $([\tau]_1 g)^l [\phi]_1 f [\phi]_2$;

(7.3) $([\tau]_1 g)^l [\phi]_1 f [\tau]_2 (g[\tau]_1)^{n-l+1}$

In light of (7.1) and (7.3), the initial segment lemma implies the following.

LEMMA 8 :

If w is a word from $([\tau]_1 g)^l [\phi]_1 f [\tau]_2 (g[\tau]_1)^m$, then w is in $|F_{n+1}|$ iff $o \leq l \leq n$ and $m=n-l+1$.

Having shown in lemma 6 that no "while-do" is w-equivalent to F_{n+1}, we now consider compositions.

LEMMA 9 :

Suppose $H=H_1 \cdot H_2$ is w-equivalent to F_{n+1}, $n \geq o$. Then one of H_1 and H_2 is w-equivalent to the trivial scheme.

Proof :

By lemma 2, every word in $|F_{n+1}|$ may be "split" as a composite of a word in $|H_1|$ with a word in $|H_2|$, and lemma 1 implies this splitting is unique. To establish lemma 9, we will consider certain words known to be in $|F_{n+1}|$ by lemma 7, and determine their unique splittings, using lemma 8 as needed.

First, suppose w is in $([\tau]_1 g)^{n+1}$ and y is in $\{\phi,\tau\}^2$. By (7.1), wy is in $|F_{n+1}|$. If the unique splitting of wy places wy in $|H_1|$ and y in $|H_2|$, the initial segment lemma would then imply that this is the splitting for wy for <u>all</u> y in $\{\phi,\tau\}^2$. Thus, we would have $\{\phi,\tau\}^2 \subset |H_2|$, and it is then immediate that $|H_2| = |T|$, and lemma 9 is valid. We may now suppose that no word wy in $([\tau]_1 g)^{n+1}\{\phi,\tau\}^2$ splits with wy in $|H_1|$.

To be more specific, if ν is in $[\tau]_1$ and y is in $\{\phi,\tau\}^2$, there is a unique k with $o \leq k \leq n$ such that $(\nu g)^k \nu$ is in $|H_1|$ and $(\nu g)^{n-k+1} y$ is in $|H_2|$. Lemma 1 implies this splitting is valid for every y in $|H_2|$, so we have :

(9.1) $(\nu g)^k \nu$ is in $|H_1|$;

(9.2) $(\nu g)^{n-k+1}\{\phi,\tau\}^2 \subset |H_2|$, and $o \leq k \leq n$.

By lemma 8, we know that $(\nu g)^k \nu$ is not in $|F_{n+1}|$, so (9.1) implies that ν is not in $|H_1|$. Since this is valid for all ν in $[\tau]_1$ (because in each instance the value for k is not n+1), we may conclude :

(9.3) $[\tau]_1 \cap |H_2|$ is empty.

Combining (9.3) with the fact from (7.2) that $[\phi]_1 f\tau\phi \subset |F_{n+1}|$, we have :

(9.4) $[\phi]_1 \subset |H_1|$

(9.5) $[\phi]_1 f\tau\phi \subset |H_2|$.

Because $\{\phi,\tau\}^2 \cap |F_{n+1}|$ is empty, it follows from (9.4) that :

(9.6) $[\phi]_1 \cap |H_2|$ is empty.

We next determine the splitting of $(\nu g)^k \phi\tau f\phi\phi$, which is in $|F_{n+1}|$ by (7.2). In view of (9.6), the initial segment lemma applied to (9.1) eliminates all possibilities except :

(9.7) $(\nu g)^k \phi\tau$ is in $|H_1|$;

(9.8) $\phi\tau f\phi\phi$ is in $|H_2|$.

We now examine $w=(\nu g)^k \phi\tau f\phi\tau (g\nu)^{n-k+1}$. We claim that the unique splitting for this word of $|F_{n+1}|$ places $(\nu g)^k \phi\tau$ in $|H_1|$ and $\phi\tau f\phi\tau (g\nu)^{n-k+1}$ in $|H_2|$. To establish this, we eliminate all other possibilities. Applying the initial segment lemma to (9.1) and to (9.2) we have that $(\nu g)^m \nu$ is in $|H_1|$ iff m=k, and $(\nu g)^m \nu$ is in $|H_2|$ iff m=n-k+1. This eliminates all splittings of w except the desired one and the one with $(\nu g)^k \phi\tau f\phi\tau$ in $|H_1|$. However, when combined with (9.5), this implies that $(\nu g)^k \phi\tau f\phi\tau f\phi\tau$ is in $|F_{n+1}|$; this is not possible since one readily sees that f never appears as consecutive operator symbols in a word of

$|F_{n+1}|$. Therefore, we have :

(9.9) $\phi\tau f\phi\tau(g\nu)^{n-k+1}$ is in $|H_2|$.

Applying (9.4), we have $\phi\tau$ in $|H_1|$, hence $\phi\tau f\phi\tau(g\nu)^{n-k+1}$ is in $|F_{n+1}|$. From lemma 8, it follows that k=o, so that, returning to (9.1), we see that $\nu\in|H_1|$. As ν was an arbitrary element of $[\tau]_1$, we have $[\tau]_1\subset|H_2|$. Combining this with (9.4), we have $\{\phi,\tau\}^2\subset|H_2|$, and thus H_2 is w-equivalent to T. This completes the proof of lemma 9.

THEOREM 1

For each $i\geq o$, F_i is w-minimal over $\mathfrak{D}$.

The proof of theorem 1 is by induction on i, beginning easily as F_o=T has no edges. Now suppose, for $o\leq i\leq n$, that each F_i is w-minimal over $\mathfrak{D}$ and let H be a D-scheme w-minimal over $\mathfrak{D}$ and w-equivalent to F_{n+1}. From lemma 6, we know that the last D-operation used in forming H was not a while-do; nor was it a nontrivial composition, due to the w-minimality of H and the result of lemma 9. Therefore we may assume that $H=H_1:\pi:H_2$ for some D-schemes H_1, H_2 and some π in Π.

We shall have need of the following corollary to the characterization of D-schemes given in [8]; this result also appears in Elgot [2].

LEMMA 1o :

Let C be a cycle in a D-scheme G. Then there is a unique predicate vertex z in C such that, for any path Q in G from b to e, if Q intersects C, then z is both the first and the last point of C encountered in traversing Q.

Recall that a simple path is one in which no vertex is repeated. Evidently a simple path from b to e can intersect a cycle C in at most its unique "base point" z. One way in which simple paths occur that is of particular interest to us is when the path terminates at the exit and always, when exiting from a predicate vertex, obeys a constant "instruction string" α in $\{\phi,\tau\}^2$. Such a path must be simple as a repeated vertex would force the path to cycle infinitely. For example the path determined in F_{n+1} by the word $\phi\phi f\phi\phi$ is simple. An important consequence of this is the fact that the terminal point v_1 of $P[b;\phi\phi]$ lies on no cycle in F; this is because v_1 is an operator vertex (labelled by f) and a simple path contacts cycles in predicate vertices only.

Now consider the path P in F_{n+1} determined by the word $w=\phi\phi f\tau\tau(g\tau\tau)^{n+1}$, which is in $|F_{n+1}|$ by (7.3). We may write $P=P_oP_1P'$ where $P_o=P[b;\phi\phi]$ terminates at v_1, and $P_1=P[v_1;\tau\tau]$ terminates at v_2. P_o is an initial segment of the simple path determined by $\phi\phi f\phi\phi$, and so is itself a simple path. Also, since P_1P' terminates at the exit and obeys the constant "instruction"

$\tau\tau$, it too is a simple path. Since v_1 lies on no cycle we may conclude that it is the unique point of intersection of P_o and P_1P'. Thus $P=P_oP_1P'$ is a simple path, so the operator vertex v_2 on P cannot lie on any cycle in F_{n+1}.

We may now invoke lemma 1o to show that $H\langle v_2,e\rangle$ is a D-scheme. Moreover, since the begin of H is a predicate vertex with $H=H_1:\pi:H_2$, and we reached v_2 after leaving the begin by "obeying" the "instruction" $\phi\phi$, v_2 is in H_1 and $H\langle v_2,e\rangle=H_1\langle v_2,e\rangle$. Because v_2 is labelled by the operator symbol g, $H_1\langle v_2,e\rangle$ = $=g\cdot H_1\langle v_3,e\rangle$, where v_3 is the target of the unique outedge from v_2. If we now apply lemma 4 to the word $\bar{w}=\phi\phi f\tau\tau g\tau\tau$, we have $L_{\bar{w}}(|H|) = L_{\bar{w}}(|F_{n+1}|) = |H_1\langle v_2,e\rangle| = g|H_1\langle v_3,e\rangle|$. By direct calculation one sees that $L_{\bar{w}}(|F_{n+1}|) = g|F_n|$, so we conclude that $|H_1\langle v_3,e\rangle| = |F_n|$. Since $H_1\langle v_3,e\rangle$ is a D-scheme and F_n is w-minimal over $\mathfrak{D}$, we have that $H_1\langle v_3,e\rangle$ has at least as many edges as F_n. If we let E(G) denote the <u>number</u> of edges in G, we may write this as $E(H_1\langle v_3,e\rangle)\geq E(F_n)$.

If we examine the simple paths determined by $\phi\phi f\phi\phi$ and $\phi\phi f(\tau\tau g)^{n+1}\tau\tau$ respectively, we see that they coincide until reaching v_1 and that at some predicate node, prior to v_2 on the latter path, they part. The predicate node cannot be in $H_1\langle v_2,e\rangle$, so we may count edges to conclude that $E(H_1)\geq 3+E(H_1\langle v_2,e\rangle)=$ $=4+E(H_1\langle v_3,e\rangle)\geq 4+E(F_n)=E(f\cdot(T:\pi_2:g\cdot F_n))$.

An even simpler version of this argument shows that

$E(H_2) \geq 1+E(F_n)=E(g \cdot F_n)$. Since $H=H_1:\pi:H_2$, H_1 and H_2 have no edges in common; taking into account the two outedges of the begin of H, we have

$|H|=2+|H_1|+|H_2| \geq 2+E(f \cdot (T:\pi_2:g \cdot F_n))+E(g \cdot F_n)=E(F_{n+1})$.

We have thus shown that F_{n+1} is w-minimal over $\mathfrak{D}$, completing the proof of theorem 1. It should be noted that the above arguments, slightly extended, show that in fact $H=F_{n+1}$, so that for this particular weak equivalence class we have a unique minimal D-scheme.

We now define w-equivalent "nonstructured" schemes G_n, $n \geq 1$, by taking the "minimal automaton" of F_n. Specifically, G_1 and the recursive construction for G_{n+1}, are given in figure 4. That G_n is w-equivalent to F_n is immediate from the notion of minimal automaton (more precisely, minimal scheme [8]), but this may be verified by computing $|G_1|$, and by seeing that the recursive computation for $|G_{n+1}|$ is exactly the same as for $|F_{n+1}|$. It is a simple matter, after the proof of theorem 1, to show that each G_n is w-minimal over the class of all schemes.

Computing $E(F_{n+1})$ and $E(G_{n+1})$ is made simple by their recursive definitions, so that $E(F_{n+1})=7+2E(F_n)$ and $E(G_{n+1})=5+E(G_n)$; hence $E(F_{n+1})=7(2^{n+1}-1)$ and $E(G_n)=5n$. We have thus met the objectives given in the introduction, and we gather together our observations into a theorem.

THEOREM 2 :

For each $n \geq 1$, there are schemes F_n and G_n such that

(1) F_n is a D-scheme w-minimal over $\mathfrak{A}$;

(2) G_n is w-equivalent to F_n;

(3) F_n has $7(2^n-1)$ edges; and

(4) G_n has $5n$ edges.

ACKNOWLEDGEMENTS

The work in this paper was carried out while the author was at the Universität des Saarlandes on leave from Stevens Institute of Technology. I am pleased to express my gratitude to the Universität des Saarlandes, and especially to Professor Günter Hotz, for support provided during this period. I would also like to thank Professor Kurt Mehlhorn for posing a question to the author that provided the stimulus for the present work.

REFERENCES

[1] C.Böhm and G.Jacopini,"Flow diagrams, Turing machines, and languages with only two formation rules". Comm.ACM 9(1966), 366-371.

[2] C.C.Elgot, "Structured programming with and without go to statements". IEEE Trans. on Software Engineering, SE-2 (1976), 41-54.

[3] S.A.Greibach, "Theory of program structures : schemes, semantics, verification". Lect. Notes in Computer Science 36(1975), Springer-Verlag.

[4] D.E.Knuth and R.W.Floyd, "Notes on avoiding 'Go To' statements". Information Processing Letters 1(1971), Errata 1(1972).

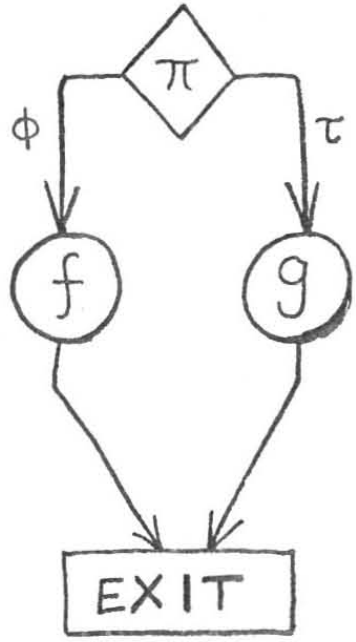
[5] S.R. Kosaraju, "Analysis of structured programs". Journal Computer System Sci. 9(1974), 232-255.

[6] D.C. Luckham, D.M.Park, and M.S.Paterson, "On formalized computer programs". J. Comput. Syst. Sci. 4(197o), 22o-249.

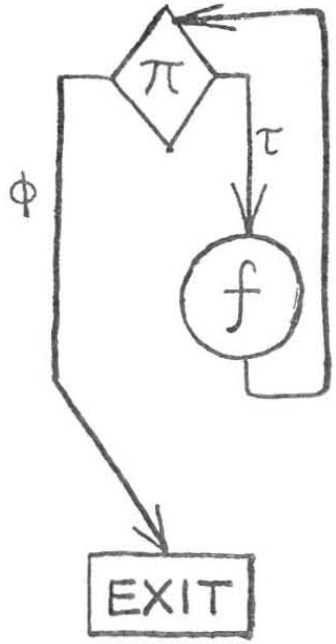
[7] J.D.Rutledge, "On Ianov's program schemata". J.ACM 11(1964), 1-9.

[8] R.Tindell, "The existence of unique minimal D-schemes", to appear.

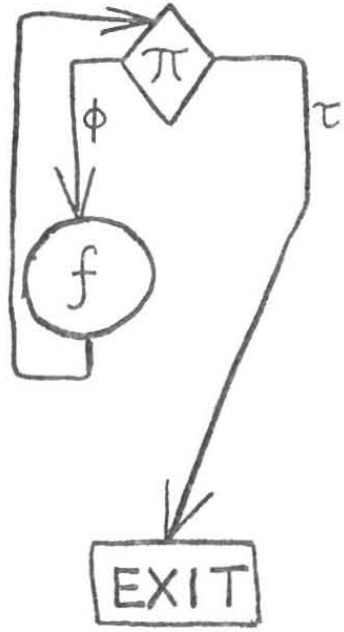
1.1 f



1.2 $f:\pi:g$

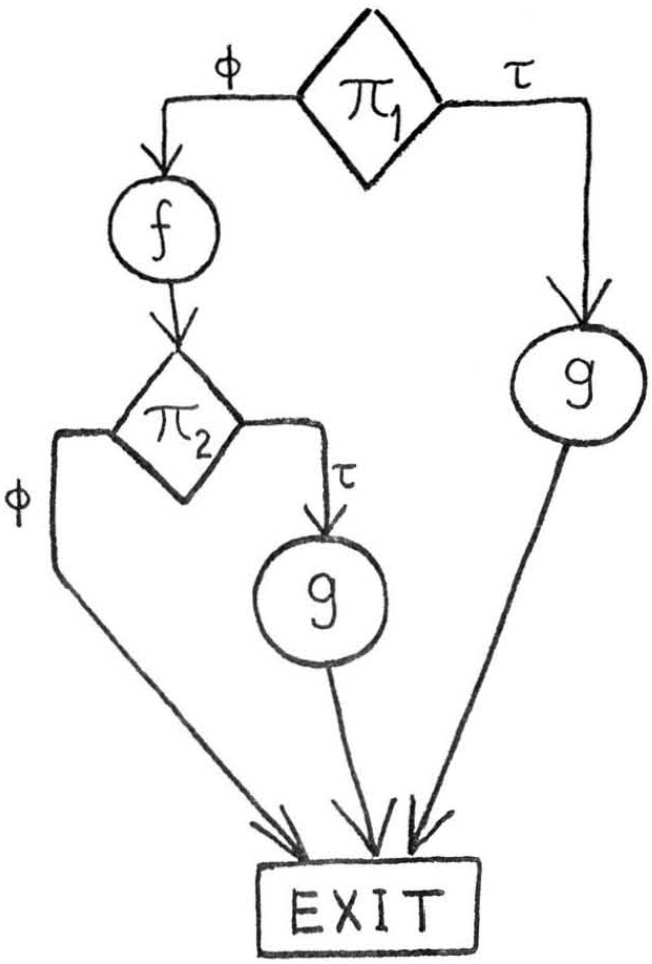


1.3 $\pi[f]$

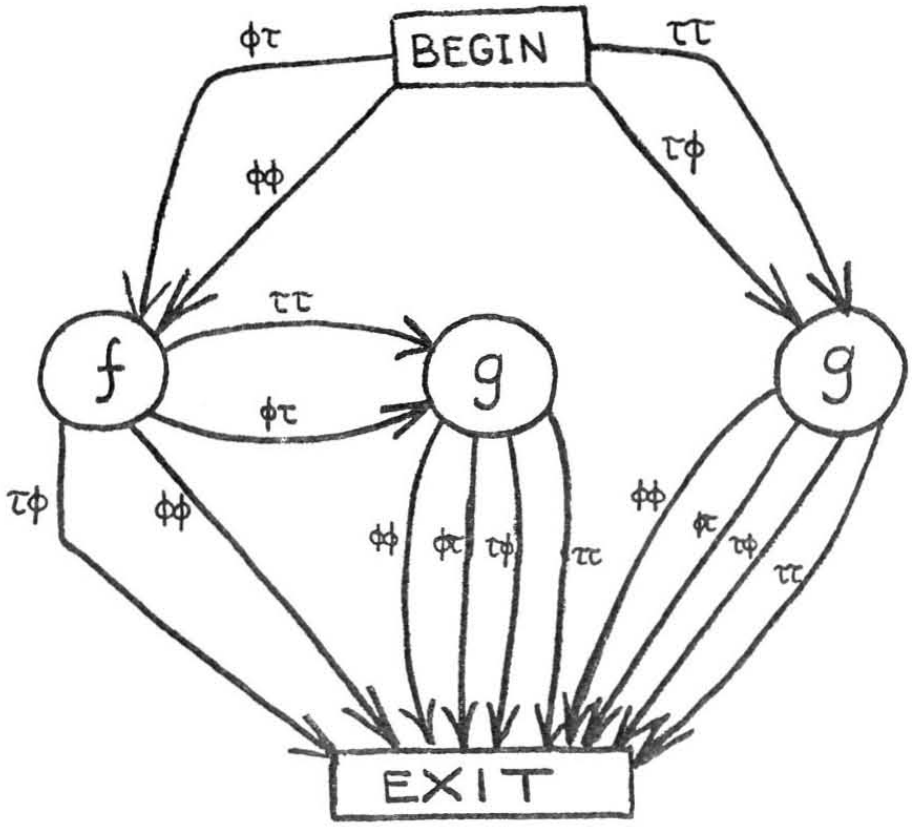


1.4 $\tilde{\pi}[f]$

FIGURE 1



F_1



F_1'

FIGURE 2

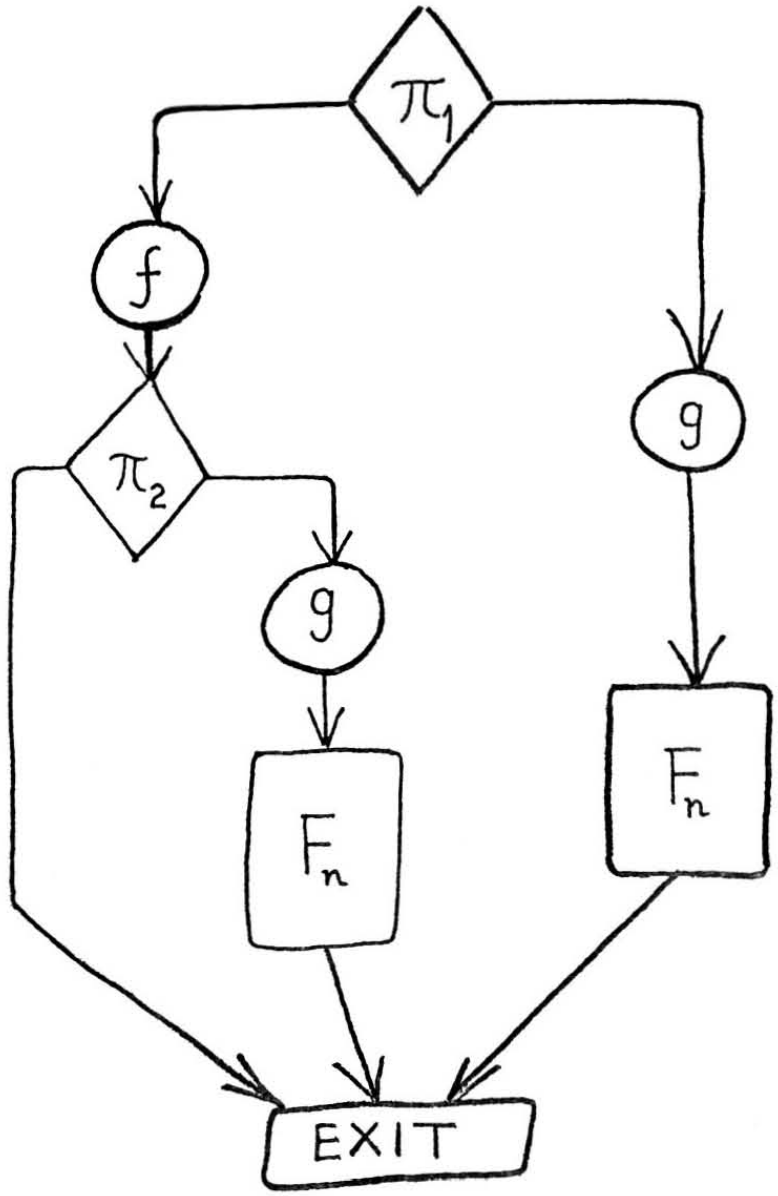


FIGURE 3 : F_{n+1}

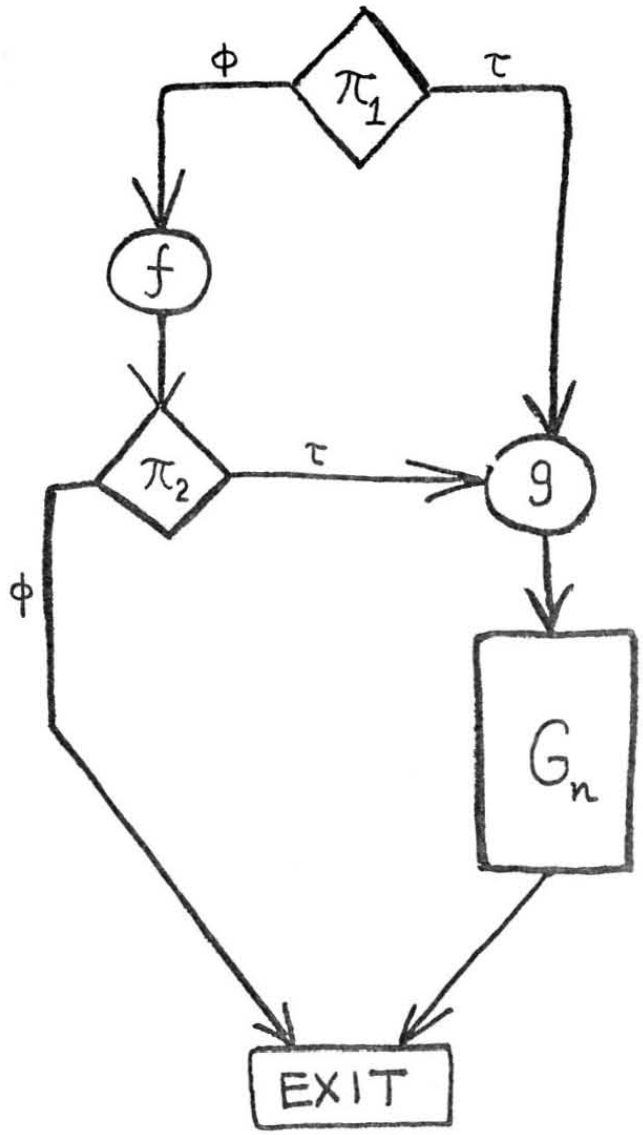


FIGURE 4: G_{n+1}