Analysis of Graphs by Ordering of Nodes

C. P. EARNEST, K. G. BALKE,* AND J. ANDERSON

Computer Sciences Corporation, El Segundo, California

ABSTRACT. A method of analyzing directed graphs by establishing a particular ordering for the nodes is presented, and properties of the ordered graph are derived. Both the method and the resultant order are interesting graph-theoretically, and also practically: the method is quite efficient, and the results of the analysis are particularly useful to an object code optimizer for programs. The paper includes examples of interesting graphs, and a conjecture about a refinement of the method.

KEY WORDS AND PHRASES: program flow graph, directed graph, strongly connected region, predominator, object code optimization

CR CATEGORIES: 4.12, 5.32

1. *Introduction*

A program can be and often is represented as a directed graph, in which each arc represents a flow path, and each vertex represents a "basic block"—that is, a set of instructions having the property that each time any member of the set is executed, all are. For many purposes, it is necessary to analyze the structure of a graph representing a program—to discover strongly connected regions, basic blocks which predominate others, basic blocks which can occur on a path from one basic block to another, etc.

We present here a new method of graph analysis, based on the notion of establishing an ordering for the vertices. Algorithms are presented which establish an order which is consistent with certain natural precedence relations between the vertices, and which makes a certain set of strongly connected regions readily apparent.

Previous work reported in the literature has been concerned with finding either all [3] or only the maximal [4] strongly connected regions. For many purposes, the first is not necessary, and the second is not sufficient. The method given here finds a set of nested strongly connected regions, such that for any pair of regions R_i and R_j in the set, either $R_i \subseteq R_j$ or $R_j \subseteq R_i$ or $R_i \cap R_j$ is empty. Further, every strongly connected region which is not found is a subset of, and has a region entry node in common with, some found region.

The method and the theoretical results, while applicable to any directed graph,

Copyright © 1972, Association for Computing Machinery, Inc.
General permission to republish, but not for profit, all or part of this material is granted, provided that reference is made to this publication, to its date of issue, and to the fact that reprinting privileges were granted by permission of the Association for Computing Machinery.

* Present address: Burroughs Corporation, City of Industry, California. This research was partly supported by the National Science Foundation under Grant NSF-GJ-95, at the Courant Institute of Mathematical Sciences, New York University.

are of most use for graphs having a single "entry" vertex from which there is a path to every other vertex. The algorithms are quite fast and require little space—probably not much more efficient ones are possible. They were developed originally as eminently practical solutions to the problem of analysis of program flow graphs for the purpose of object code optimization. The order established is useful for either of two rather different styles of optimizers. The first views the program as a set of nested regions, to be processed from the inside out, at each step reducing the inner region to a loop-free set of one or more vertices. The other views the program as a single straight sequence of steps, with flow arrows connecting different parts of the sequence, to be processed in order, taking note of significant events which affect each step by virtue of the flow arrows. Both methods[1] are described in [2]; in both cases, information about strongly connected regions and the flow within them is important. Such information is developed by the analysis algorithms of this paper.

2. *Definition; Notation*

Our usage will be consistent with that of Berge [1]. Assume we have a finite set X, and a function Γ mapping X into X; the pair (X, Γ) is a finite (directed) graph. An element of X is a *vertex* or a *node*. It will be necessary to refer to nodes by either their names, or their ordering indices within a particular order, or both. The name of a node never changes (it is assumed to exist to start with); the ordering index may change, and is initially undefined. Thus we establish the following notation:

N	is any node,
N^i	is the unique node with name i,
N_k	is the node (at most one) with ordering index $= k$,
$N_k{}^i$	is the node with name i and ordering index $= k$,
$O(N^i)$	is the ordering index, or O attribute, of node N^i, defined by $O(N_k{}^i) = k$.

An *arc* of the graph is a pair (N^i, N^j), where $N^j \in \Gamma N^i$. Such an arc is said to be *from* N^i *to* N^j. The notation $N^i \rightarrow N^j$ will be used both as a name for an arc from N^i to N^j and as a relation indicating that such an arc exists; which is meant will be clear from the context. If $N^i \rightarrow N^j$ and $N^j \rightarrow N^k$, this may be written as $N^i \rightarrow N^j \rightarrow N^k$ (etc.). If $N^i \rightarrow N^j$, then N^i is an *immediate predecessor* of N^j, and N^j is an *immediate successor* of N^i.

A *path* in the graph is a sequence of two or more (not necessarily distinct) nodes $N^{i_1}, N^{i_2}, \cdots, N^{i_k}$ such that $N^{i_1} \rightarrow N^{i_2} \rightarrow \cdots \rightarrow N^{i_k}$. The path is said to *include* N^{i_1}, N^{i_2}, $\cdots$, N^{i_k}; the path is said to be *from* N^{i_1} *to* N^{i_k}. The path is said to be *within a set* of nodes S if and only if $\{N^{i_1}, N^{i_2}, \cdots, N^{i_k}\} \subseteq S$.

A *strongly connected region* is a set of nodes such that there is a path within the set from any node to any node. (For convenience, this definition allows a single node as a strongly connected region; Berge's *strongly connected graph* must have at least two nodes.)

The graph is assumed, without real loss of generality, to have one and only one *entry node* N^e such that there is no N^k for which $N^k \rightarrow N^e$.

[1] The second method, called the linear nested region scheme in [2], was developed chiefly by the present authors, particularly K. G. Balke.

A node N^b *predominates* another node N^c if and only if every path from the entry node to N^c includes N^b; in that case N^b is a *predominator* of N^c. A node N^b is the *immediate predominator* of a set of nodes C if and only if N^b predominates every node in C, and N^b does not predominate any other node which predominates every node in C. A node N^a is a *serial predecessor* of another node N^c if and only if N^a predominates N^c, or N^d is the immediate predominator of the set $\{N^a, N^c\}$ and there exists a path not including N^d from N^a to N^c, but not such a path from N^c to N^a.

Note that the set of predominators of a node N^c is ordered by the predomination relationship: Let N^a and N^b both be predominators of N^c; every path from the entry node to N^c thus includes both N^a and N^b. In every such path, either N^a or N^b must be closest to N^c; we assume that in one such path, N^b is closest—that is, the latter part of the path from the entry node to N^c is a path not including N^a from N^b to N^c. There can then be no path not including N^a from the entry node to N^b, for such a path could obviously be extended to reach to N^c without including N^a, contradicting the assumption that N^a predominates N^c. Thus every path from the entry node to N^b includes N^a, or in other words, N^a predominates N^b.

A number of terms dependent on an ordering of the nodes of the graph are now defined; in the absence of such an ordering, these terms are undefined: A *backward arc* is an arc from N_i to N_k, where $i \geq k$; all nodes N_{j_n} for which $k \leq j_n \leq i$ are said to be *under the backward arc.* Two backward arcs $N_m \rightarrow N_j$, $N_k \rightarrow N_i$ are said to be *interlocked* if $i < j \leq k < m$. A *forward arc* is an arc from N_i to N_k, where $i < k$. A *forward path* from N_{i_1} to N_{i_k} is a sequence of two or more nodes $N_{i_1}, N_{i_2}, \cdots, N_{i_k}$ such that $N_{i_1} \rightarrow N_{i_2} \rightarrow \cdots \rightarrow N_{i_k}$ and $i_n < i_{n+1}$ for all n, $1 \leq n < k$ (that is, a path from N_{i_1} to N_{i_k} which contains only forward arcs). For convenience, we denote a contiguously indexed set of nodes $\{N_i, N_{i+1}, N_{i+2}, \cdots, N_j\}$, $i \leq j$, as $(N_i{:}N_j)$. A *potential loop* is a set of nodes $(N_i{:}N_j)$ such that either $N_j \rightarrow N_i$, or there exist N_k, N_m such that $i < k \leq m < j$ and both $(N_i{:}N_m)$ and $(N_k{:}N_j)$ are potential loops. If $(N_i{:}N_j)$ is a potential loop, then N_i is a *loop head*, and for each backward arc $N_n \rightarrow N_i$, $i \leq n$, N_n is a *latching node.* (Informally, a potential loop is the set of nodes under a backward arc, or is the union of sets of nodes under interlocking backward arcs. Note that by this definition, a potential loop need not contain a strongly connected region, since it is defined in terms of an arbitrary ordering.)

3. *Basic Numbering Algorithm*

The ordering algorithm consists of two distinct parts: the basic numbering algorithm (BNA), which establishes an initial ordering for the nodes by assigning a distinct ordering index to each node; and the loop cleansing algorithm (LCA), which establishes a somewhat different ordering by changing some of the initial index assignments. We present first the basic numbering algorithm, which is so simple that it is not likely to be new. Next, we demonstrate some of the properties of the ordered graph, which have apparently not been explicitly stated before.

It should be noted that the versions of both the BNA and the LCA given herein have been chosen to be as clear as possible; a more efficient version of the entire combined BNA/LCA algorithm is possible, but has neen omitted to save space.

For convenience, we denote the set of all indexed nodes by S. S is initially empty;

the algorithm will insert the nodes into it one at a time. The number of nodes in S at a given time is denoted by $n(S)$.

THE BASIC NUMBERING ALGORITHM

(1) Insert the entry node into S by assigning it the index 1. That is, set $O(N^e) := 1$, where N^e is the entry node.
(2) Set $n(S) := 1$ (count the entry node as a member of S).
(3) Set $p := 1$ (p is a cursor pointing to the node currently being processed).
(4) If there is an immediate successor N^a of N_p which is not yet indexed (i.e., $N_p \rightarrow N^a$ and $N^a \notin S$), then:
  (a) Increase the indices of all following nodes by 1. That is, for all i such that $p < i \leq n(S)$, set $O(N_i) := O(N_i) + 1$.
  (b) Insert N^a into S by assigning it the index $p + 1$. That is, set $O(N^a) := p + 1$.
  (c) Set $p := p + 1$ (p now points to the node just inserted; that is, N^a is now $N_p{}^a$).
  (d) Set $n(S) := n(S) + 1$ (count the new member of S).
  (e) Return to Step (4).
(5) If there is no immediate successor of N_p which is not yet indexed, and $p > 1$, then set $p := p - 1$ and return to Step (4). Otherwise ($p \leq 1$) the algorithm terminates. (Note that if there is a path from the entry node to every other node, then the termination can be made to occur when every node has been indexed, with a probable saving of time over the above method).

4. *Properties of the Graph After Basic Numbering*

It is clear that the basic numbering algorithm assigns a distinct index i_j, $1 \leq i_j \leq n(S)$, to each node to which there is a path from the entry node. For the remainder of the paper, we consider only these nodes—that is, the nodes in S.

It should be noted that the order established by application of the BNA to a given graph is not unique in general—it depends on the order in which the immediate successors of a node are chosen.

For use in showing the properties of the graph after application of the BNA, we need another definition: a *direct path* to a node N_k is a path from N_1 to N_k such that for all i, $1 \leq i < k$, $N_i \rightarrow N_{i+1}$ (i.e., $N_1 \rightarrow N_2 \rightarrow \cdots \rightarrow N_k$).

LEMMA 1. *At all times during the application of the basic numbering algorithm, a direct path to the node being processed (N_p) exists, except when $N_p = N_1$ = the entry node.*

PROOF. If the lemma is true just before the indexing of a node N^a, it is still true thereafter, because N^a is an immediate successor of the former N_p, by definition of the BNA; the arc $N_{p-1} \rightarrow N_p{}^a$ either creates (if $p = 2$) or extends a direct path to $N_p{}^a$. Decreasing the value of p by 1 only removes a node from the direct path to N_p; a direct path continues to exist (unless of course $p = 1$). Thus the lemma is (vacuously) true to start with, and remains true through any possible change in the value of p during the BNA. QED

LEMMA 2. *If after application of the BNA, $N_k{}^c \rightarrow N_i{}^a$ and $i \leq k$ (i.e., there is a backward arc from N^c to N^a), then just after N^c was inserted in S, the direct path to it included N^a.*

PROOF. Note first that the BNA never changes the relative order of nodes as determined by their indices, once these have been assigned. Thus, if after the BNA, for $N_i{}^a$ and $N_k{}^c$, if $i \leq k$, then it is never true during the BNA that $O(N^a) > O(N^c)$.

If N^a was not in S just after N^c was added, then N^a would have been inserted to follow N^c—that is, so that $O(N^a) > O(N^c)$. This is so because the cursor p could never have been set to a smaller value than $O(N^c)$ until N^a was in S, because $N^c \rightarrow N^a$ [Step (5) could not be operated with $N^c \equiv N_p$ unless N^a was in S]. But as noted above, the relation $O(N^a) > O(N^c)$ is impossible; thus N^a was in S just after N^c was added. But at that time, by Lemma 1, there was a direct path to N^c, and since N^a was in S and $O(N^a) \leq O(N^c)$ was true, that direct path must have included N^a. QED

THEOREM 1. *After application of the BNA, for each backward arc, there is a forward path from the loop head to every other node under the backward arc.*

PROOF. Let the given backward arc be $N_k^c \rightarrow N_i^a$, where $i \leq k$. When N^c was inserted in S, there was a direct path to it which included N^a, by Lemma 2; thus at that time the theorem was clearly true. For any node inserted later with a larger index than $O(N^a)$, there was at the time of its insertion a direct path to it which included N^a, by Lemma 1 and the definition of direct path. The existence of these paths meeting the conditions of the theorem could not be affected by later insertions into S. QED

Theorem 2 is immediately clear from Theorem 1; Theorem 3 and Corollary 1 are clear from Lemma 1; all are given without proof.

THEOREM 2. *After application of the BNA, every potential loop includes a strongly connected region.*

THEOREM 3. *After application of the BNA, there is a forward path from N_1 to every other node in S.*

COROLLARY 1. *After application of the BNA, every potential loop can be entered through the loop head.*

That is, if after application of the BNA $N_k \rightarrow N_i$ and $i \leq k$, there is a forward path from N_1 to N_i.

THEOREM 4. *If N^a predominates N^b and there is a forward path from N_1 to every other node (as there is after application of the BNA), then $O(N^a) < O(N^b)$.*

PROOF. By definition of predominator, every path from the entry node to N^b includes all predominators of N^b. Thus the forward path from N_1 to N^b includes N^a, and so $O(N^a) < O(N^b)$. QED

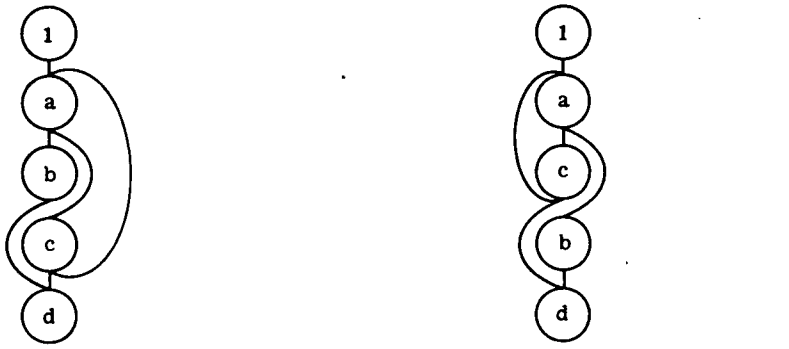
5. *Loop Cleansing Algorithm*

The basic numbering algorithm ensures that the set of nodes under every backward arc includes a strongly connected region. It does not ensure that every loop head is the first node of a contiguously indexed set of nodes which is strongly connected, however. For example, the order shown in Figure 1 could result from basic numbering.[2]

The loop cleansing algorithm (LCA) is designed to remedy this situation, in a sense which will be made clear.

As mentioned above, the loop cleansing algorithm accepts as initial input the indexed set S produced by the basic numbering algorithm. The LCA reassigns indices to nodes in S, to establish a different ordering. The algorithm follows.

[2] In all examples of graphs in the paper, arcs enter only at the top of nodes and leave only at the bottom. The assigned order is the order of the nodes from top to bottom of the page.



BNA order
(Order of insertion into S: N_1, N^a, N^c, N^d, N^b)

Preferred order
[(N^a; N^c) in a strongly connected region]

FIG. 1

The Loop Cleansing Algorithm

(1) Set $i := n(S)$; erase all marks.
(2) Inspect N_i as a possible loop head: for the set of all backward arcs to N_i, denote the corresponding set of latching nodes by $L = \{N_j \mid i < j \leq n(S) \text{ and } N_j \rightarrow N_i\}$. If L is empty (i.e., N_i is not a loop head[3]), go to Step (3). Otherwise, cleanse separately one potential loop for each backward arc to N_i as follows.
   (a) Select from L the N_l having the smallest index of any node in L (i.e., $N_l \in L$, and there is no $N_k \in L$ such that $k < l$). (This causes the potential loops to be cleansed in order from the inner to the outer.) If N_l is marked, go to Step (2f).
   (b) Mark N_l.
   (c) Recursively mark all nodes N_k for which $i < k \leq n(S)$ and there is an N_m such that $N_k \rightarrow N_m$ and N_m is marked. (That is, mark all immediate predecessors of N_l, then mark all immediate predecessors of those nodes, etc., except that no node N_b, $b \leq i$, is marked.)
   (d) Set $t :=$ the largest index belonging to a marked node. We define N_t to be the *loop tail*.
   (e) Cleanse the potential loop $(N_i{:}N_t)$ by removing unmarked nodes from it: Let M be the set of marked nodes and U be the set $\{N_u \mid i < u < t \text{ and } N_u \text{ not marked}\}$. Reassign consecutive indices $i + 1, i + 2, \cdots, t$ to all nodes in $M \cup U$ in such a way that the original relative order within M and within U is preserved, but so that every node in M now has a smaller index than any node in U.
   (f) Set $L := L - \{N_l\}$. (Remove the latching node from L.) If L is not yet empty, return to Step (2a). Otherwise, proceed.
   (g) Erase all marks.
(3) Set $i := i - 1$. If $i \geq 1$, return to Step (2).
   If $i < 1$, the algorithm terminates.

6. *Properties of the Graph after Loop Cleansing*

Note that, for a given order established by the BNA, the order after the LCA is unique; the converse is in general not true. That is, the LCA does not introduce any new nonuniqueness; neither does it necessarily preserve the nonuniqueness created by the BNA. We will refer to the order established by the LCA as *LCA order*.

[3] Except possibly for the trivial loop $N_i \rightarrow N_i$.

As we will show, LCA order has the important property that every *formal loop* is a strongly connected region, where *formal loop* is defined recursively as follows:

$(N_i{:}N_k)$ is a *primitive formal loop* if and only if $N_k \rightarrow N_i$, $i \leq k$, and there is no $N_m \rightarrow N_j$ such that $i < j \leq k < m$.

$(N_h{:}N_k)$ is a *formal loop* if $(N_h{:}N_k)$ is a primitive formal loop, or if $(N_i{:}N_k)$ is a formal loop and there is some $N_j \rightarrow N_h$ such that $h < i \leq j < k$.

A more intuitive, less precise definition of *formal loop* follows: For each backward branch $N_j \rightarrow N_h$, $h \leq j$, there is a *formal loop* R with loop head N_h. R can be constructed as follows: R is initially $(N_h{:}N_j)$. If there is a backward arc from outside R to a node in R other than N_h (i.e., $N_k \rightarrow N_i$, $h < i \leq j < k$) then extend R to include the latching node of that arc and all nodes between [i.e., $R{:=}\ (N_h{:}N_k)$]. Repeat this extension procedure as long as possible—the final set R is then a (not necessarily unique) formal loop. It is clear that these two definitions of formal loop are equivalent; the first "builds" formal loops from the bottom up, the second from the top down. The second definition reflects the way in which the LCA works: Each potential loop cleansed is in fact a formal loop.

Some examples of formal loops are shown in Figure 2. Note that the set under a backward arc is not necessarily a formal loop, and is not necessarily a strongly connected region after application of the LCA. Figure 3 shows the simplest example which illustrates this. The example shows a possible LCA order, but one in which $(N^a{:}N^c)$ is neither a formal loop nor a strongly connected region (though it is a potential loop). Other examples appear in Appendix A.

The properties of an order which are of interest to us can be summarized in a definition, which does not depend on any particular algorithm: *Straight order* is an ordering of the nodes of a single entry directed graph such that:

(1) every formal loop is a strongly connected region,
(2) there is a forward path from the entry node to every other node, and
(3) For every backward arc, there is a forward path from the loop head to every other node under the backward arc.

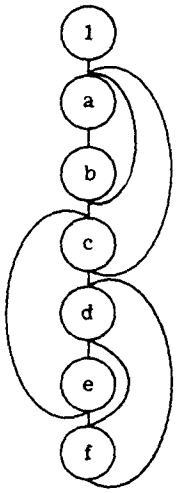


FIG. 2

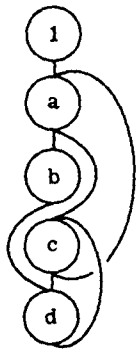


FIG. 3

These three properties are independent, in that no one can be derived from the other two, as is shown in Figure 4. As was shown in Theorem 4, property (2) ensures that, in a straight order, predominators precede the nodes they predominate.

It will come as no surprise that an order established by the LCA is a straight order (although not vice versa), as we will now show, by demonstrating that the LCA preserves properties (2) and (3) from the order established by the BNA and in addition establishes every formal loop as a strongly connected region.

LEMMA 3. *The LCA does not destroy any forward paths. That is, if after the BNA there is a forward path from N^a to N^b, the same forward path exists after the LCA.*

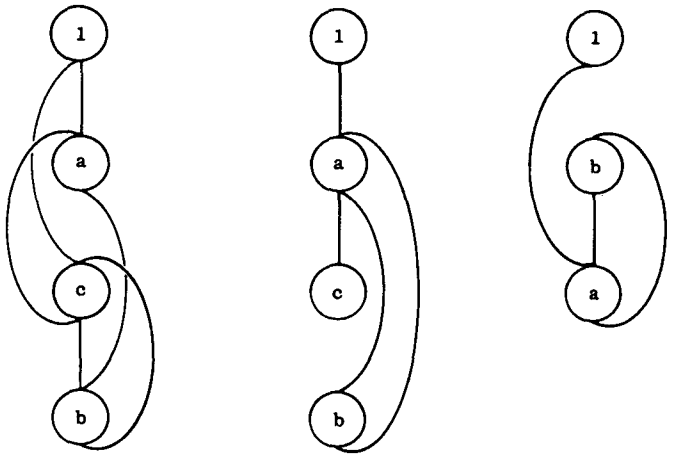
PROOF. If before cleansing a potential loop, $O(N^a) < O(N^b)$, then that same relation holds after completion of the cleansing unless $N^a \in\cdot U$ and $N^b \in M$ [the sets U and M for that cleansing—see Step (2e), LCA]. For this case, however, there cannot have been a forward path from N^a to N^b, for then N^a would have been marked (by definition of the marking procedure) and would be in M, not U. Clearly, forward paths not including some $N^a \in U$ and some $N^b \in M$ (i.e. *all* forward paths) are not affected by the cleansing. QED

LEMMA 4. *The LCA does not change the direction of any arcs. That is, if $N^a \rightarrow N^b$, the LCA does not change the sign of $O(N^a) - O(N^b)$.*

PROOF. This follows immediately from Theorem 1 and Lemma 3. If before the LCA $O(N^a) < O(N^b)$, then there is a forward path from N^a to N^b, which by Lemma 3 is not destroyed. If before the LCA $O(N^a) > O(N^b)$, there is by Theorem 1 a forward path from N^b to N^a; by Lemma 3 this is not destroyed. If $O(N^a) = O(N^b)$, it is clear that N^a and N^b are the same node, which is of course not split.

LEMMA 5. *The LCA inspects each node exactly once as a possible loop head in the reverse order from that established by the BNA; further, the order of nodes not yet inspected is that established by the BNA.*

PROOF. At any given time during the LCA, the cursor i divides the set S of nodes into the two disjoint sets $A = \{N^k \mid O(N^k) < i\}$ and $B = \{N^j \mid O(N^j) \geq i\}$. Cleansing potential loops with loop head N_i cannot affect the order within or the composition of the set A; the setting of i to $i-1$ [Step (3), LCA] simply removes the node with largest index from set A and adds it to set B for inspection as a



(1) and (2) only (2) and (3) only (1) and (3) only

FIG. 4

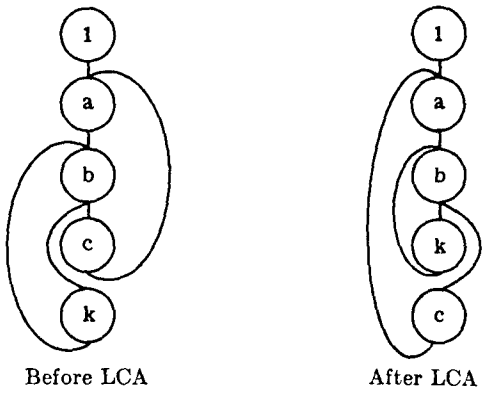


FIG. 5

possible loop head. The value of i is never increased, so that once a node is in B it cannot be removed. QED

THEOREM 5. *The LCA terminates, given a finite graph.*[4]

PROOF. By Lemma 5, the LCA inspects each node as a possible loop head exactly once; each such inspection can trigger at most one cleansing for each backward branch to the loop head. The number of nodes is finite; the number of backward branches to each node is also finite, and by Lemma 4, no new backward branches are inserted during the LCA. The marking and moving procedure obviously terminates each time (at most every node could be marked, and there is a finite number of nodes). QED

LEMMA 6. *The LCA inserts a new node N^k into the set $(N^a{:}N^c)$ under a backward arc $N^c \rightarrow N^a$, $O(N^a) \leq O(N^c)$ only during the cleansing of some potential loop with loop head N^b, where before the cleansing, $O(N^a) < O(N^b) < O(N^c) < O(N^k)$.*

PROOF. If N^k is not already in $(N^a{:}N^c)$ before the cleansing, either $O(N^k) < O(N^a)$ or $O(N^c) < O(N^k)$. Consider first the case in which (before the cleansing) $O(N^k) < O(N^a)$. The LCA can reverse the order of N^k and N^a only if $N^k \in U$ and $N^a \in M$ for some cleansing. Note that if $N^a \in M$ during a cleansing, then $N^c \in M$ also, because $N^c \rightarrow N^a$ and $O(N^a) \leq O(N^c)$. Thus even if $N^k \in U$ for the cleansing, the resultant order after moving will be $O(N^a) \leq O(N^c) < O(N^k)$; N^k will not be inserted into the set $(N^a{:}N^c)$.

The opposite case: $O(N^c) < O(N^k)$, $N^c \in U$, and $N^k \in M$ can occur and can result in the order $O(N^a) < O(N^k) < O(N^c)$; Figure 5 shows an example. But if this is the resultant order, N^a is not in U for the cleansing (else the cleansing would reverse the order of N^a and N^k). N^a also is not in M, nor is $N^a = N^b$, for in that case N^c would be in M, as noted earlier. Therefore $N^a \notin (N^b{:}N^x)$, the potential loop being cleansed. Since $N^c \in U \subset (N^b{:}N^x)$, and $O(N^a) \leq O(N^c)$, we have $O(N^a) < O(N^b) < O(N^c)$. QED

THEOREM 6. *For a graph in LCA order, for every backward arc $N^c \rightarrow N^a$, $O(N^a) \leq O(N^c)$, there is a forward path from the loop head N^a to every other node N^k in the set under the arc [i.e. for which $O(N^a) < O(N^k) \leq O(N^c)$].*

[4] This is proven because it may not be immediately clear; it is obvious that the BNA terminates, given a finite graph.

PROOF. By Theorem 1, this is true just after the BNA; Lemma 3 shows that the forward paths from the loop head to nodes under the arc after BNA cannot be destroyed. It is therefore sufficient to show that the LCA does not insert into $(N^a{:}N^c)$ nodes to which there is not a forward path from N^a. The proof is by recursive reasoning: we assume that at the start of any cleansing the theorem is true, and show that the cleansing does not affect its truth.

By Lemma 6, any insertion of a node N^k into the set $(N^a{:}N^c)$ can take place only during the cleansing of a potential loop with loop head N^b, such that $O(N^a) < O(N^b) < O(N^c) < O(N^k)$; by assumption, we therefore have a forward path from N^a to N^b [since $N^c \rightarrow N^a$, $O(N^a) < O(N^b) < O(N^c)$].

It is easy to see that for the cleansing of a potential loop with loop head N^b and loop tail N^t, the marking procedure in the LCA guarantees a path within the set $M^* = M \cup \{N^b\}$ from every marked node N^m to N^b. Each such path must obviously include a backward arc $N^y \rightarrow N^x$, such that $O(N^b) \leq O(N^x) \leq O(N^m) \leq O(N^y)$. By the recursion assumption, either $N^m = N^x$, or there is a forward path from N^x to N^m. But note that N^x is itself in M^*, so either $N^x = N^b$, or there is a forward path from some loop head N^z, $O(N^b) \leq O(N^z) < O(N^x)$, to N^x, by the same reasoning. Clearly, this sequence of loop heads with smaller and smaller indices, all greater than or equal to $O(N^b)$, will eventually lead to N^b; thus there is a forward path from N^b to every marked node, including N^k.

Putting these two forward paths together, we have a forward path from N^a to N^b to N^k. QED

COROLLARY 2. *The LCA does not insert nodes into a previously cleansed potential loop [i.e. into the set $(N^a{:}N^b)$, where N^a was the loop head and N^b the loop tail for a previous cleansing].*

PROOF. It is clear from the definition of the LCA that $(N^a{:}N^b)$ is a formal loop just after it has been cleansed. That is, at that time $(N^a{:}N^b)$ is a set under a backward arc, or the union of two or more sets under interlocked backward arcs. For each such backward arc $N^d \rightarrow N^c$, $O(N^a) \leq O(N^c) \leq O(N^d) \leq O(N^b)$. By Lemma 5, if $(N^a{:}N^b)$ has been cleansed, $O(N^a) \geq i$ (the LCA cursor). But by Lemma 6, insertion of a new mode into $(N^c{:}N^d)$ can occur only if $O(N^c) < i$, which is not the case.

Note that the LCA never causes interlocked arcs to become disjoint: that is, for $N^r \rightarrow N^p$, $N^s \rightarrow N^q$, $O(N^p) < O(N^q) \leq O(N^r) < O(N^s)$ at any time after the BNA, the LCA will not remove N^q from $(N^p{:}N^r)$, because the proof of Theorem 6 shows that there are forward paths from N^p to N^q and from N^q to N^r, which by Lemma 3 are not destroyed by the LCA. Therefore the set $(N^a{:}N^b)$, once it has been cleansed, will always consist of a set under a backward arc, or the union of two or more such sets, and as we have shown, insertion under any backward arc within $(N^a{:}N^b)$ is impossible. QED

This proof, together with Theorem 6, leads immediately to the following:

COROLLARY 3. *For a previously cleansed set $(N^a{:}N^b)$, there is a forward path from N^a to every N^k for which $O(N^a) < O(N^k) \leq O(N^b)$.*

LEMMA 7. *The LCA does not remove nodes from previously cleansed potential loops. That is, given the previously cleansed set $(N^a{:}N^c)$ and any N^b such that $O(N^a) < O(N^b) < O(N^c)$, the LCA will not reverse the order of N^a and N^b, or of N^b and N^c.*

PROOF. By Corollary 3, there is a forward path from N^a to N^b; by Lemma 3 this cannot be destroyed, so the order of N^a and N^b cannot be reversed.

By Corollary 2, N^b was in $(N^a{:}N^c)$ just after that set was cleansed (it could not have been inserted later). It is clear that the marking procedure guarantees that, at that time, $(N^a{:}N^c)$ also contained at least one immediate successor N^k of N^b—that is, $N^b \to N^k$, $N^b \neq N^k$, $N^k \in (N^a{:}N^c)$. If $O(N^b) < O(N^k)$, then removal of N^b without removal of N^k would reverse the direction of the $N^b \to N^k$ arc, which cannot happen, by Lemma 4. If $O(N^b) > O(N^k)$, the removal of N^b without removal of N^k would cause the insertion of at least N^c into the $N^b \to N^k$ backward arc; since $O(N^k) \geq O(N^a)$, this cannot happen, by Lemma 6.

Therefore N^b can be removed from $(N^a{:}N^c)$ only if every immediate successor of N^b in $(N^a{:}N^c)$ is removed as well. Thus all nodes which are immediate predecessors of N^a cannot be removed; neither can immediate predecessors of these nodes, etc. But this is precisely the set M, and $\mathrm{M} \cup \{N^a\} = (N^a{:}N^c)$. QED

THEOREM 7. *A graph in LCA order is also in straight order.*

PROOF. Lemma 3 and Theorems 3 and 6 have established that LCA order has properties (2) and (3) of straight order. It remains to be shown only that for LCA order, every formal loop $(N^h{:}N^t)$ is a strongly connected region. As noted earlier, it is clear that the marking procedure guarantees the existence of a path within $M^* = \mathrm{M} \cup \{N^h\}$ from any $N^k \in M$ to the loop head N^h of the cleansed potential loop. By Corollary 3, there is a forward path from N^h to any $N^j \in M$. M^* is therefore a strongly connected region, and thus so is $(N^h{:}N^t)$ just after it has been cleansed [the new $(N^h{:}N^t) \equiv M^*$].

It has also been shown (Corollary 2 and Lemma 7) that the LCA neither inserts nodes into nor removes nodes from previously cleansed potential loops. Therefore, at the completion of the LCA, every cleansed potential loop is a strongly connected region. It remains to be shown that every formal loop (as defined after completion of the LCA) was either a cleansed potential loop or is a strongly connected region in any case.

Let $(N^a{:}N^b)$ be a formal loop after the completion of the LCA. By the definition of formal loop, there is a backward arc $N^c \to N^a$, $O(N^a) \leq O(N^c) \leq O(N^b)$, which by Lemmas 4 and 5, triggered a cleansing unless either (1) $N^c = N^a$, or (2) N^c was marked when $N^c \to N^a$ was inspected. Case (1) is the trivial loop $N^a \to N^a$, which is not affected by the BNA or the LCA. In case (2), N^c could have been marked only if it were in $(N^a{:}N^d)$, a previously cleansed set with the same loop head. The cleansing of $(N^a{:}N^d)$ could have been triggered only by a backward arc $N^k \to N^a$, where $O(N^a) < O(N^k) < O(N^c)$, because of the order in which the LCA processes latching nodes. From Theorem 6 and Corollary 3, it is apparent that there would be a forward path from N^k to N^c. Thus even if all marks were erased and the marking procedure begun again with N^c, N^k would again be marked, and so at least all of $(N^a{:}N^d) - \{N^a\}$ would be marked—in other words, the fact that $N^c \to N^a$ did not trigger a cleansing could not cause a different result than if it had triggered one. Thus we can assume that at least $(N^a{:}N^c)$ was cleansed.

By the definition of formal loop, either $(N^a{:}N^c) \equiv (N^a{:}N^b)$ or there is some $N^g \to N^f$, for which $O(N^a) < O(N^f) \leq O(N^c) < O(N^g)$. If such an arc exists after completion of the LCA, it must have existed when $(N^a{:}N^c)$ was cleansed, because by Lemma 7, N^g could not have been removed since that cleansing, by Corollary 2, N^f could not have been inserted, and by Lemma 4, the backward arc $N^g \to N^f$ forces $O(N^f) < O(N^g)$ at all times during the LCA. But then the set cleansed would have been at least $(N^a{:}N^g)$. This same argument continues (a finite number of steps) to show that the potential loop cleansed was at least $(N^a{:}N^b)$. Since there

is no $N^k \rightarrow N^j$, $O(N^a) < O(N^j) \leq O(N^b) < O(N^k)$, by the definition of formal loop, there could have been none when $(N^a{:}N^b)$ was cleansed, by Lemmas 4 and 7 and Corollary 2 (as above). Thus the potential loop cleansed was precisely $(N^a{:}N^b)$. QED

Given a graph in straight order, a set of nested strongly connected regions—i.e., the formal loops—is easily found. It is not so easy to decide which strongly connected regions will be formal loops in some straight order. It is clear that, for two strongly connected regions R_i and R_j for which $R_i \cap R_j$ is not empty, and $R_i \not\subset R_j$ and $R_j \not\subset R_i$, at most one of the two will be a formal loop in a particular straight order. If $R_i \subset R_j$, then in some cases both regions will be formal loops, and in some cases only R_j will be; Figure 6 gives an example. The following exposition is intended to show that, for some purposes, it is more convenient to have available the set of formal loops for some straight order rather than the set of all strongly connected regions of the graph.

We define an *innermost formal loop* to be a formal loop $R^i = (N^a{:}N^b)$, such that there is no formal loop $R^j = (N^c{:}N^d)$ for which $(N^c{:}N^d) \subset (N^a{:}N^b)$. Note that an innermost formal loop is a primitive formal loop, but not necessarily vice versa.

THEOREM 8. *For an innermost formal loop $R = (N^a{:}N^b)$, the only backward arc to a node in R is the arc $N^b \rightarrow N^a$.*

PROOF. Since an innermost formal loop is a primitive formal loop, we have $N^b \rightarrow N^a$. There can be no included backward arc $N^d \rightarrow N^c$, where either $O(N^a) \leq O(N^c) \leq O(N^d) < O(N^b)$ or $O(N)^a < O(N^c) \leq O(N^d) \leq O(N^b)$, because then $(N^c{:}N^d)$, not $(N^a{:}N^b)$, would be an innermost formal loop. There can be no backward arc $N^d \rightarrow N^c$, where $O(N^a) < O(N^c) \leq O(N^b) < O(N^d)$, for then R would not be a formal loop. QED

We now define a *reduced graph* to be the graph produced by performing the following alterations on any graph in straight order which contains a formal loop:

Choose any innermost formal loop $R = (N^a{:}N^b)$.

Remove all nodes in R from the graph and replace them with a single new node N^R, such that:

(1) $O(N^R) = O(N^a)$,
(2) $N^j \rightarrow N^R$ iff $N^j \notin R$ and there was some $N^k \in R$ for which $N^j \rightarrow N^k$ [note that $O(N^j) < O(N^a)$, because R is a formal loop],
(3) $N^R \rightarrow N^m$ iff $N^m \notin R$ and there was some $N^n \in R$ for which $N^n \rightarrow N^m$.

An example of such a graph reduction is shown in Figure 7.

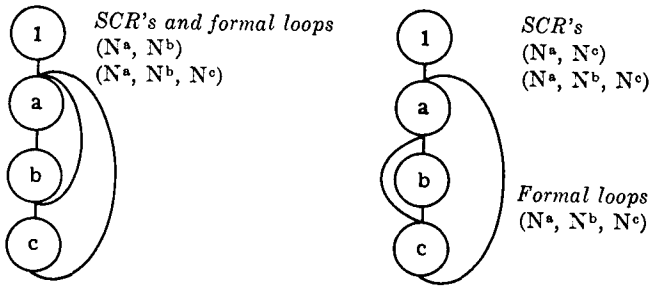
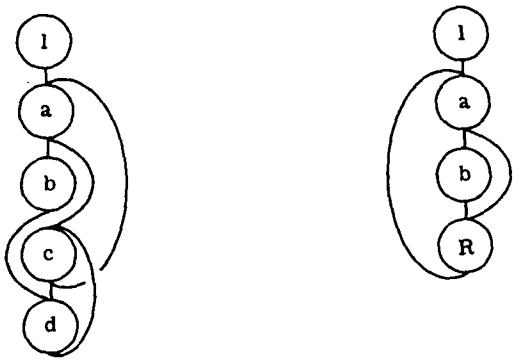


FIG. 6



Graph in straight order Reduced graph

FIG. 7

THEOREM 9. *Every reduced graph is in straight order.*

PROOF. It is clear that all paths not within R which existed before reduction continue to exist thereafter, except that wherever a path included a sequence of one or more nodes in R, it includes after reduction the single node N^R. Further, all new arcs are forward or backward accordingly as the arcs they effectively replace were forward or backward. Thus the reduction preserves the three properties of straight order. QED

(It is also possible to prove that if the reduction process is performed on a graph in LCA order, the result is a graph in LCA order, which is a bit stronger. As the proof is tedious, and the result not useful unless other useful properties of LCA order are discovered, it is omitted.)

THEOREM 10. *For a graph in any given order, every strongly connected region in the graph is a subset of, and includes the loop head of, some formal loop.*

PROOF. Assume there is a strongly connected region $R = \{N^{a_1}, N^{a_2}, \cdots, N^{a_k}\}$, with $O(N^{a_1}) < O(N^{a_2}) < \cdots < O(N^{a_k})$ in the given order. By definition, there is a path within R from N^{a_k} to N^{a_1}. Clearly such a path includes a backward arc to N^{a_1} (a forward arc to N^{a_1} would perforce be from a node not in the region). It is easy to see from the definition of formal loop that N^{a_1} is thus the loop head of one or more formal loops; of these, choose the one having the most nodes—call it F. Assume that there is a node N^r in R but not in F. Since the nodes in F are contiguous, and $N^{a_1} \in F$, and $O(N^r) > O(N^{a_1})$, then $O(N^r) > O(N^f)$ for any $N^f \in F$. By definition of strongly connected region, there is a path from N^r to N^{a_1}, which must clearly include a backward arc from a node not in F to a node in F. But then there is a formal loop with loop head N^{a_1} which contains more nodes than F, which is a contradiction—thus every node in R is in F. QED

THEOREM 11. *If N^a is a serial predecessor of N^b, then when the graph is in straight order, $O(N^a) < O(N^b)$.*

PROOF. The result is immediate if N^a predominates N^b, by Theorem 4.

If not, let N^{BD} be the immediate predominator of $\{N^a, N^b\}$; we will use the fact throughout the proof that N^a, N^b, and N^{BD} are three distinct nodes. By the definition of serial predecessor, there is a path not including N^{BD} from N^a to N^b; if $O(N^a) > O(N^b)$ that path must include a backward arc $N^d \to N^c$, where $O(N^{BD}) < O(N^c) \leq$

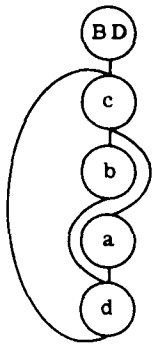


FIG. 8

$O(N^b) < O(N^a) \leq O(N^d)$. Figure 8 shows an example. Note that $O(N^{BD}) < O(N^c)$: Every path from N_1 to N^a must include N^{BD}, by definition of predominator. By definition, there is a forward path from N_1 to N^c (and to every other node); thus if $O(N^c) < O(N^{BD})$, every path from N^c to N^a would have to include N^{BD}, which is not the case.

Thus there is a formal loop $(N^c{:}N^g)$, where $O(N^g) \geq O(N^d)$, which is by definition a strongly connected region. Thus there is a path within $(N^c{:}N^g)$ from N^b to N^a, not including N^{BD}. But this is not the case by the definition of serial predecessor. Therefore, $O(N^a) < O(N^b)$, since $O(N^a) > O(N^b)$ leads to a contradiction. QED

7. *Summary*

We have presented a graph analysis algorithm in two parts—called basic numbering and loop cleansing—which establishes a *straight order* for the nodes of a directed graph. In general, for a given graph, there is more than one possible straight order. The most important or useful properties of straight order are summarized informally here:

(1) For every backward arc, there is a contiguously ordered set of nodes which is a strongly connected region. The first node in the set is the loop head of the backward arc; the last node in the set is the latching node of the latest backward arc which interlocks, directly or indirectly, with the first one (cf., the definition of *formal loop*). The set of ranges so defined is thus a set of nested strongly connected regions.

(2) The set of formal loops is convenient in that an innermost formal loop is the set under a single backward arc, and if an innermost formal loop is replaced by a single node, the resultant *reduced graph* is again in straight order. Thus the set of formal loops can be used to direct a straightforward "inside-out" processing of the entire graph, if desired: an innermost formal loop, which is topologically very simple, is first inspected, then reduced to a single node (with which is presumably associated the pertinent information about the former formal loop). This procedure is simply repeated until the entire graph has been processed.

(3) Straight order is compatible with serial predecessor order, so that if the graph is inspected in straight order, all the immediate predecessors of a node are inspected before the node itself, unless the node is a loop head. Straight order also ensures that predominators precede the nodes they predominate.

(4) There is a forward path to every node, both from the entry node and from the loop heads of any enclosing backward arcs. This of course means that every potential loop can be entered through the loop head.

8. *A Conjecture*

For some purposes, single entry strongly connected regions are more convenient to deal with—a strongly connected region is single entry if every branch from a node not in the same region to one in the region is to the same node. This partly explains the rationale behind building formal loops from the bottom up: for interlocked back-

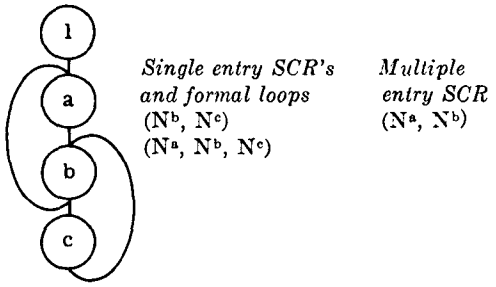


FIG. 9

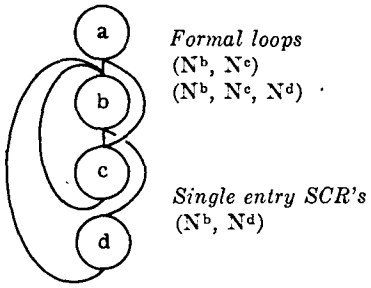


FIG. 10

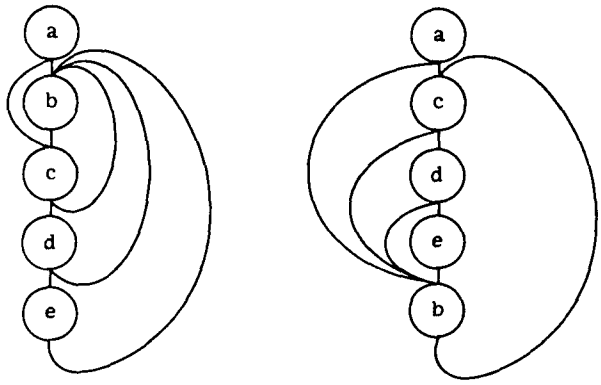


FIG. 11

ward arcs, the set under the first is never single entry, while the set under the last may be, as is shown in Figure 9, for example.

It is in general not true that for a graph in straight order, every single entry strongly connected region R_i is a formal loop, except that where there is another

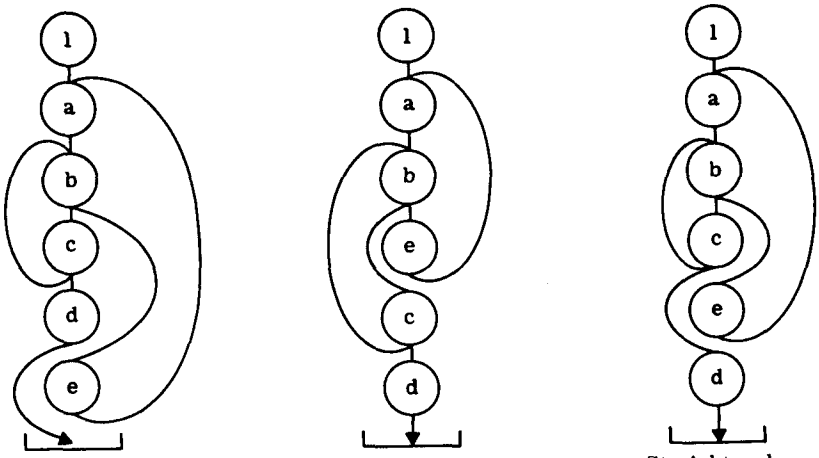
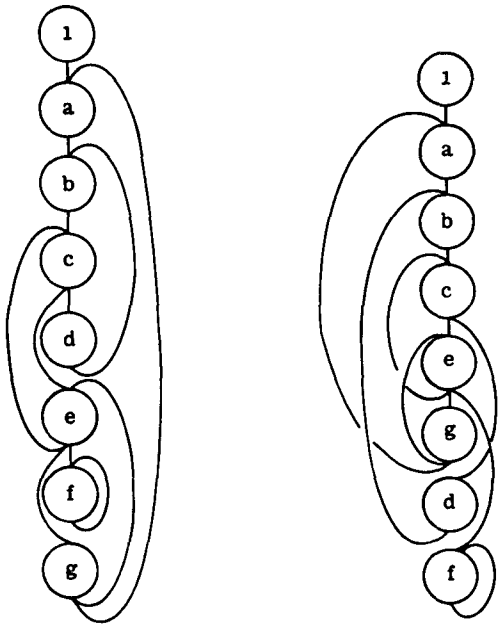


FIG. 12



BNA order		*Straight order*	
Nested	$N^g \rightarrow N^e$,	$N^f \rightarrow N^f$	Disjoint
Nested	$N^g \rightarrow N^a$,	$N^d \rightarrow N^b$	Interlocked
Interlocked	$N^d \rightarrow N^b$,	$N^e \rightarrow N^c$	Nested
Disjoint	$N^d \rightarrow N^b$,	$N^g \rightarrow N^e$	Nested

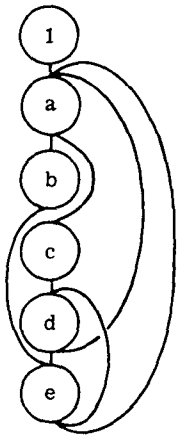
FIG. 13

single entry strongly connected region R_j such that $R_i \cap R_j$ is not empty, and $R_i \not\subset R_j$ and $R_j \not\subset R_i$ then at most one of R_i, R_j is a formal loop. This is shown by the counterexample in Figure 10.

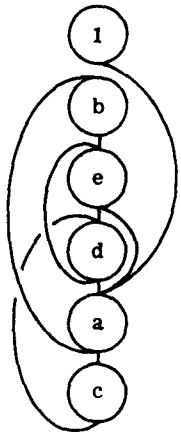
It is conjectured that if the following changes are made to the loop cleansing algorithm, the order which it then establishes is a straight order for which the above statement is true. The changes enable the LCA to re-order latching nodes from which there is an arc to the same loop head. The changes:

(1) Associate with each node N_j a loop entry count, denoted by $E(N_j)$, initially zero.

(2) Reverse the order of Steps (2f) and (2g). This causes all marks to be erased after each backward arc is processed, rather than waiting until all for a given loop head have been processed.

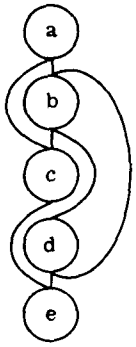


Valid straight order; (N^a:N^d) is not a strongly connected region. Other orders are possible, but in none of them are both the conditions true

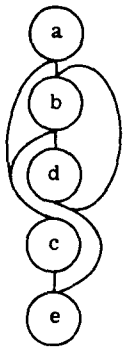


Every set under a backward arc is strongly connected, but there is not a forward path from N_1 to N^b

FIG. 14



Possible BNA order (N^d is a serial predecessor of N^e)



Valid straight order

FIG. 15

(3) Insert the following steps between Steps (2c) and (2d):

(2c1) Set $E(N_l) := 1$. For each marked node N_k for which there is at least one N_j such that $N_j \rightarrow N_k$ and $j < i$, set $E(N_l) := E(N_l) + 1$ (that is, count the entries to the potential loop, and keep the count with the latching node).

(2c2) Mark all nodes N_p for which $0 < E(N_p) \leq E(N_l)$. If any additional nodes are thereby marked, repeat Step (2c); otherwise, proceed. (Informally: mark all nodes belonging to potential loops having the same loop head as, and no more entries than, the current potential loop.)

(4) Add Step (2h), following Step (2g):

(2h) Set $E(N_k) := 0$ for all k.

Appendix

A number of graphs which for some reason are interesting follow:

(1) Figure 11 shows two possible straight orders for the same graph, one having more backward arcs than the other. Neither order seems preferable to the other.

(2) It is not true that for every graph, one of the possible orders producible by basic numbering only is a valid straight order; see Figure 12.

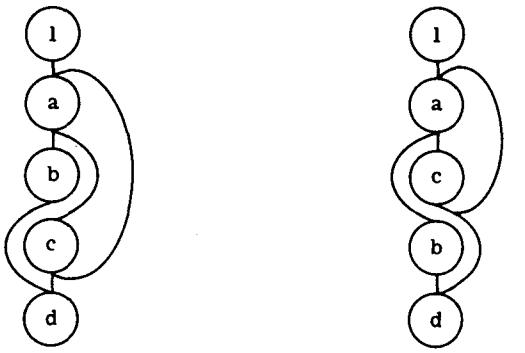
(3) The LCA sometimes changes nested backward arcs to interlocked ones, and vice versa, and disjoint backward arcs to nested ones, and vice versa. Figure 13 gives an example.

(4) It is not true that every graph can be ordered so that (1) *every* set under a backward arc is a strongly connected region and (2) there is a forward path from N_1 to every loop head. Figure 14 illustrates this.

(5) In general, the BNA order is not compatible with serial predecessor order, as is shown in Figure 15.

(6) Even where the BNA order is compatible with serial predecessor order, loop cleansing may be necessary to produce a valid straight order, as Figure 16 shows.

(7) It is not true in general that if N^a is not a serial predecessor of N^b, and N^b is not a serial predecessor of N^a, then there is a straight order in which $O(N^a) < O(N^b)$, and a straight order in which $O(N^a) > O(N^b)$. This is shown by Figure 16:

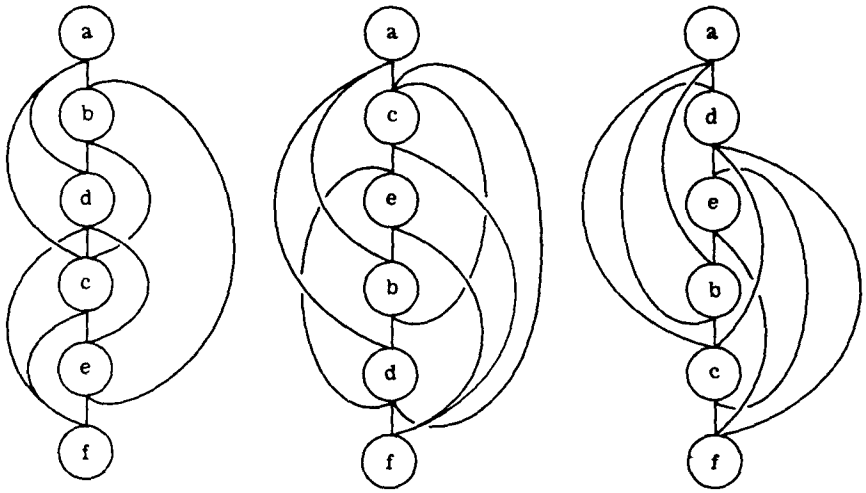


Possible BNA order Straight order

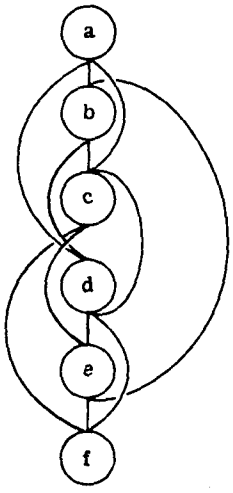
FIG. 16

Neither N^b nor N^c is a serial predecessor of the other, but the only straight order has $O(N^c) < O(N^b)$. It is not clear how to simply define a relation $<\cdot$ between nodes which has the property that $N^a <\cdot N^b$ implies that $O(N^a) < O(N^b)$ in a straight order, and so that $\neg((N^a <\cdot N^b) \vee (N^b <\cdot N^a))$ implies that both $O(N^a) < O(N^b)$ and $O(N^a) > O(N^b)$ are possible in different straight orders.

In any case, straight order cannot be defined by a pairwise relation between nodes (although of course a straight order can be). In Figure 17, three possible straight orders are shown, illustrating that for each of the six possible pairs of nodes (N^b, N^c, N^d, N^c), either member of the pair may precede the other. Another possible nonstraight order is shown for the graph.



Possible straight orders



Nonstraight order

FIG. 17

REFERENCES

1. BERGE, CLAUDE. *The Theory of Graphs and Its Applications.* Wiley, New York, 1964.
2. COCKE, JOHN, AND SCHWARTZ, J. T. *Programming Languages and Their Compilers: Pr liminary Notes.* Courant Institute of Mathematical Sciences, New York U., New York, 197
3. PROSSER, R. Applications of Boolean matrices to the analysis of flow diagrams. *Pro Eastern Joint Comput. Conf. 1959* (Spartan Books, Washington, D.C.), pp 133–137.
4. RAMAMOORTHY, C. V. Analysis of graphs by connectivity considerations. *J. ACM 1* (1966), 211–222.

RECEIVED MAY 1969; REVISED AUGUST 1971