Characterizations of Reducible Flow Graphs

M. S. HECHT

University of Maryland, College Park, Maryland

AND

J. D. ULLMAN

Princeton University, Princeton, New Jersey

ABSTRACT. It is established that if G is a reducible flow graph, then edge (n,m) is backward (a back latch) if and only if either $n = m$ or m dominates n in G. Thus, the backward edges of a reducible flow graph are unique.

Further characterizations of reducibility are presented. In particular, the following are equivalent: (a) $G = (N,E,n_0)$ is reducible. (b) The "dag" of G is unique. (A dag of a flow graph G is a maximal acyclic flow graph which is a subgraph of G.) (c) E can be partitioned into two sets E_1 and E_2 such that E_1 forms a dag D of G and each (n, m) in E_2 has $n = m$ or m dominates n in G. (d) Same as (c), except each (n, m) in E_2 has $n = m$ or m dominates n in D. (e) Same as (c), except E_2 is the back edge set of a depth-first spanning tree for G. (f) Every cycle of G has a node which dominates the other nodes of the cycle.

Finally, it is shown that there is a "natural" single-entry loop associated with each backward edge of a reducible flow graph.

KEY WORDS AND PHRASES: flow graph, reducibility, dominance, dag, depth-first spanning tree, cycle, interval

CR CATEGORIES: 4.12, 5.32

1. *Introduction*

Many global flow analysis algorithms are designed for "reducible" flow graphs, a class of flow graphs which occur quite often in practice. Here we present several new results about flow graph reducibility. This work continues some of the ideas and the approach to flow analysis suggested in [1–3]. We begin by reviewing the pertinent definitions and results of [1].

Definition. A *flow graph* is a triple $G = (N,E,n_0)$, where:

(1) N is a finite set of *nodes.*

(2) E is a subset of $N \times N$ called the *edges.* The edge (n_1, n_2) *enters* node n_2 and *leaves* node n_1. We say that n_1 is a *predecessor* of n_2, and n_2 is a *successor* of n_1.

A *path* from n_1 to n_k is a sequence of nodes $(n_1, \cdots, n_k)$ such that (n_i, n_{i+1}) is in E for $1 \leq i < k$. The *path length* of $(n_1, \cdots, n_k)$ is $k - 1$. If $n_1 = n_k$, the path is a *cycle.*

(3) n_0 in N is the *initial node.* There is a path from n_0 to every node.

This work was supported by NSF grant GJ-1052.

Authors' addresses: M. S. Hecht, Department of Computer Science, University of Maryland, College Park, MD 20742; J. D. Ullman, Department of Electrical Engineering, Princeton University, Princeton, NJ 08540.

The usual definition of reducibility is in terms of "intervals" [4–11]. We use an equivalent definition, taken from [1], based on two flow graph transformations.

Definition. Let $G = (N,E,n_0)$ be a flow graph. Let (n, n) be an edge of G. Transformation T_1 is removal of this edge.

Let n_2 not be the initial node and have a single predecessor, n_1. Transformation T_2 is the replacement of n_1, n_2 and (n_1, n_2) by a single node n. Predecessors of n_1 become predecessors of n. Successors of n_1 or n_2 become successors of n. There is an edge (n, n) if and only if there was formerly an edge (n_2, n_1) or (n_1, n_1).

The next three results are known [1].

THEOREM 1. *If T_1 and T_2 are applied to a flow graph until no longer possible, then a unique flow graph results, independent of the sequence of applications of T_1 and T_2 actually chosen.*

THEOREM 2. *A flow graph is reducible by intervals* [5] *if and only if repeated application of T_1 and T_2 yields a single node.*

Therefore, we define a flow graph to be a *reducible flow graph* (*rfg*) if and only if it becomes a single node upon repeated application of T_1 and T_2.

Definition. Let (*) denote any of the flow graphs represented in Figure 1, where the wiggly lines denote node disjoint (except for endpoints, of course) paths; nodes a, b, c, and n_0 are distinct, except that a and n_0 may be the same.

THEOREM 3. (The (*)-Characterization Theorem). *A flow graph is nonreducible if and only if it contains* (*).

2. *Dominance, Regions, Parses, and Backward Edges*

Definition. Let $G = (N,E,n_0)$ be a flow graph and let m and n be two distinct nodes in N. Node m *dominates* node n in G if every path in G from n_0 to n contains m. (Usually, we omit the reference to G when it is obvious.)

LEMMA 1. *If $G = (N,E,n_0)$ is a flow graph, then n_0 dominates all nodes in $N - \{n_0\}$.*

PROOF. Obvious. □

Definition. Let $G = (N,E,n_0)$ be a flow graph, let $N_1 \subseteq N$, let $E_1 \subseteq E$, and let n_1 be in N_1. We say $R = (N_1,E_1,n_1)$ is a *region* of G with *header* n_1 if in every path $u_1, \cdots, u_k$, where $u_1 = n_0$ and u_k is in N_1, there is some $i \leq k$ such that (1) $u_i = n_1$; and (2) $u_{i+1}, \cdots, u_k$ are in N_1; and (3) $(u_i, u_{i+1}), (u_{i+1}, u_{i+2}), \cdots, (u_{k-1}, u_k)$ are in E_1. That is, access to every node in the region is through the header only.

LEMMA 2. *A region $R = (N_1,E_1,n_1)$ of a flow graph is itself a flow graph.*

PROOF. Obvious. □

LEMMA 3. *The header of a region dominates all nodes in the region except itself.*

PROOF. By Lemma 2 and Lemma 1. □

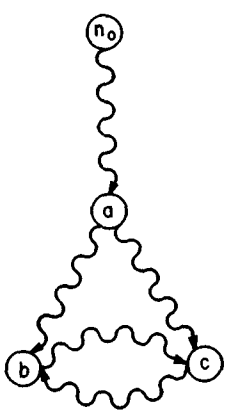

FIG. 1. The (*) subgraph

As we proceed to apply T_1 and T_2 to a flow graph, each edge of an intermediate graph represents a set of edges, and each node represents a set of nodes and edges in a natural way.

Definition. We say that each node and edge in the original flow graph *represents* itself. If T_1 is applied to node n with edge (n, n), then the resulting node *represents* what node n and edge (n, n) represented. If T_2 is applied to n_1 and n_2 with edge (n_1, n_2) eliminated, then the resulting node n *represents* what n_1, n_2 and (n_1, n_2) represented. In addition, if two edges (n_1, n') and (n_2, n') are replaced by a single edge (n, n'), then (n, n') *represents* what (n_1, n') and (n_2, n') represented.

The two lemmas which follow are from [3].

LEMMA 4. *(a) Let n be a node constructed during the reduction of some flow graph G. If n represents edge (n_1, n_2) of G, then n_1 and n_2 are represented by n.*

(b) Let n_1 and n_2 be (not necessarily distinct) nodes constructed during the reduction of G and e the edge constructed from n_1 to n_2. If e represents edge (n_3, n_4) of G, then n_3 is represented by n_1 and n_4 by n_2.

(c) In any graph formed while reducing G, all nodes and edges represent disjoint sets of objects (nodes and edges).

LEMMA 5. *Let $G = (N,E,n_0)$ be an rfg, and let $N_1 \subseteq N$ and $E_1 \subseteq E$ be a set of nodes and edges represented by a single node at some stage of the reduction of G. Then there is a node n_1 in N_1 such that (N_1,E_1,n_1) is a region of G with header n_1.*

Definition. If T_2 is applied to nodes n_1 and n_2 representing regions R_1 and R_2 respectively, such that n_1 is the unique predecessor of n_2, then we say that n_1 *consumes* n_2 or, equivalently, that R_1 *consumes* R_2.

LEMMA 6. *In a flow graph if region R results from region R_1 consuming region R_2, then the header n_1 of R_1 dominates all nodes in R_2.*

PROOF. It is easy to show that n_1 must be the header of R. Thus, n_1 dominates all nodes in R by Lemma 3. □

Since T_1 and T_2 may be applied to an rfg in different sequences (Theorem 1), it becomes necessary to discuss certain specific sequences of applications of T_1 and T_2. Informally, a "parse" of an rfg is a list of the reductions made (T_1 or T_2) and the regions to which they apply.

Definition. A *parse* π of an rfg $G = (N,E,n_0)$ is a sequence of objects of the form (T_1,n_1,n_2,S) or (T_2,n_1,n_2,n_3,S), where n_1, n_2, and n_3 are names of nodes and S is a set of edges. We define the parse of an rfg recursively as follows:

(1) A single node with no edge has only the empty sequence as its parse.

(2) If G_1 (which may not be the original flow graph in a sequence of reductions) is reduced to G_2 by an application of T_1 to node n, and the resulting node is named n' in G_2, then (T_1,n,n',S) followed by a parse of G_2 is a parse of G_1, where S is the set of edges represented by the edge (n, n) eliminated from G_1.

(3) If G_1 is reduced to G_2 by an application of T_2 to nodes n_1 and n_2 (with n_1 consuming n_2), and the resulting node is called n_3, then (T_2,n_1,n_2,n_3,S) followed by a parse of G_2 is a parse of G_1, where S is the set of edges represented by the edge (n_1, n_2) in G_1.

A given rfg may have many parses, but there are certain things in common among all these parses.

Definition. Let $G = (N,E,n_0)$ be an rfg and let π be a parse of G. We say that an edge in E is *backward* if it appears in set S of an entry (T_1,n,n',S) of π and *forward* otherwise. Let B be the set of edges in E which are backward in every parse of G.

We shall show that B is the (unique) set of backward edges of an rfg, but for the present we shall assume that the backward edges of two distinct parses of an rfg are not necessarily the same.

LEMMA 7. *If $G = (N,E,n_0)$ is an rfg, then $\{(n, n) \in E\} \subseteq B$.*

PROOF. Obvious. □

LEMMA 8. *If (n_0, m) is backward, then $m = n_0$.*

PROOF. If $m \neq n_0$, then there exists a region which includes n_0 but does not have n_0 as its header, which is impossible. □

LEMMA 9. *Let G be an rfg and let π be a parse of G. If (n, m) is a backward edge, then there is a path from m to n in G.*

PROOF. Since (n, m) is a backward edge, at some intermediate stage of the reduction of G there is a node k and edge (k, k) such that (k, k) represents (n, m). Furthermore, node k represents a region R of G headed by m and containing n. (If m did not head R, then there would be a path from n_0 to m containing an edge, namely (n, m), not in R. This path is easily seen to violate the definition of "region.") Since R itself is a flow graph, there is a path from m to n in R, and hence in G. □

LEMMA 10. *If G is an rfg and $R = (N_1,E_1,m)$ is any region of G with $n \in N_1$ and $(n, m) \in E_1$, then $(n, m) \in B$.*

PROOF. Since R is a region with header m containing n, then either m dominates n (by Lemma 3) or $m = n$. If $m = n$, then $(n, m) \in B$ by Lemma 7.

Now suppose that $m \neq n$. If (n, m) were forward, then there would exist a region $R' = (N_2,E_2,h)$ containing n. Since R' consumes a region containing m, it follows that h dominates m by Lemma 6. Either $n = h$ or $n \neq h$. If $n = h$, then n dominates m. This is a contradiction, since nodes cannot mutually dominate each other. Thus, $n \neq h$. But then there is a cycle free path from the initial node to n, as shown in Figure 2, that goes from h to m to n. This path contradicts the assumption that R' is a region, since there is a path to n from outside (m) which does not pass through the header of R'. □

Definition. If m and n are two not necessarily distinct nodes of a flow graph G, then we define the *distance from m to n*, denoted by $d(m, n)$, to be the minimum length of a path from m to n.

Definition. If P and Q are paths of a flow graph, i.e. ordered sets of nodes, then we define $P \cap Q$ to be the set of nodes in both P and Q. In other words, we ignore the fact that P and Q are ordered and just take their unordered intersection.

THEOREM 4. *Let $G = (N,E,n_0)$ be an rfg and let π be a parse of G. Edge (n, m) is backward if and only if either $n = m$ or m dominates n.*

PROOF. ($\Rightarrow$). Assume (n, m) is backward. If $n = m$, then we are done. Suppose $n \neq m$. Then $n \neq n_0$ by Lemma 8. If $m = n_0$, then m dominates n by Lemma 1. If $m \neq n_0$, suppose that m does not dominate n. Let P be a path from n_0 to n not containing m. Let R be a path from m to n (by Lemma 9). Let Q be a path from n_0 to m. Figure 3 shows this situation, where the paths shown are not necessarily disjoint. We may, however, assume R and Q to have no edges in common, since one is inside and one outside a region with header m.

Let i be a node from $P \cap Q$ such that, if k is any other node from $P \cap Q$, then $d(i, m)$

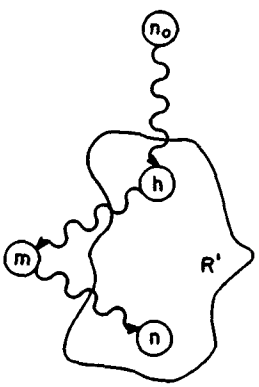

FIG. 2. Region R' of Lemma 10

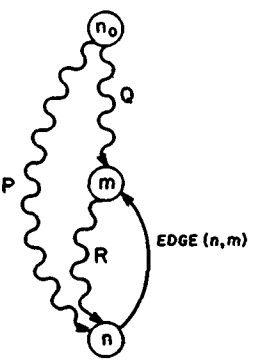

FIG. 3. Paths in G

$\leq d(k, m)$. Similarly, let j be a node from $P \cap R$ such that, if k is any other node from $P \cap R$, then $d(m, j) \leq d(m, k)$. (Note that n_0 is in $P \cap Q$ and that n is in $P \cap R$, so that both intersections are nonempty. Furthermore, Q and R are edge disjoint and $Q \cap R = \{m\}$.) But nodes i, j, m correspond to nodes a, b, c of $(*)$, and thus G is nonreducible, a contradiction. Thus, m dominates n.

$(\Leftarrow)$. Assume that either $n = m$ or m dominates n. If $n = m$, then edge (n, m) is backward by Lemma 7. Now suppose that m dominates n. There is a region $R = (N_1,E_1,m)$ in G where $n \in N_1$ and $(n, m) \in E_1$. In particular, $N_1 = \{m\} \cup \{i \in N \mid m \text{ dominates } i\}$ and $E_1 = E \cap (N_1 \times N_1)$ specifies one such R. But by Lemma 10, (n, m) is in B and hence backward in every parse π of G. □

In the second part of the proof of Theorem 4 we have really shown that if $G = (N,E,n_0)$ is an rfg and if m dominates n in G, then $(n, m) \in B$. Thus, we have as an immediate result that the backward edge set of an rfg is independent of the parse chosen and is B.

COROLLARY 4.1. *If G is an rfg and if B' is the set of backward edges produced by any parse of G, then $B = B'$. That is, the backward edges of an rfg are unique.*

3. *Dag of a Flow Graph*

In this section we show that all and only the rfg's can be decomposed uniquely into a dag and backward edges. This dag is exactly the dag formed by a depth-first spanning tree of G including its forward and cross edges but excluding its back edges. Also, the dominance relation of an rfg and its dag are the same.

Some definitions are in order, namely, "dag of a flow graph," and the notation surrounding depth-first spanning trees.

Definition. A *dag of a flow graph* $G = (N,E,n_0)$ is an acyclic flow graph $D = (N,E',n_0)$, such that $E' \subseteq E$ and for any edge e in $E - E'$, $(N, E' \cup \{e\}, n_0)$ is not a dag. That is, D is a maximal acyclic subflowgraph.

Definition. A *depth-first spanning tree* (*DFST*) of a flow graph G is a directed, rooted, ordered spanning tree grown by Algorithm A [12].

ALGORITHM A. DFST OF A FLOW GRAPH.

Input. Flow graph G.

Output. DFST of G.

Method. A1. The root of the DFST is the initial node of G. Let this node be the node n "under consideration."

A2. If the node n under consideration has a successor x not already on the DFST, select node x as the rightmost son of n found so far in the spanning tree. If this step is successful, node x becomes the node n under consideration; repeat A2. If there is no such x, go to A3.

A3. If the node under consideration is the root, then halt. Otherwise, climb down the DFST one node toward the root and consider this node by going to step A2. □

If (u, v) is an edge in a DFST, then u is the *father* of v and v is a *son* of u. The *ancestor* and *descendant* relations are the transitive closures of the father and son relations.

Let $G = (N,E,n_0)$ be a flow graph and let $T = (N,E')$ be a DFST of G. The edges in $E - E'$ fall into three classes.

(1) Edges which run from ancestors to descendants we call *forward* edges.[1]

(2) Edges which run from descendants to ancestors or from a node to itself we call *back* edges.

(3) Edges which run from one subtree to another subtree in T we call *cross* edges.

We follow the convention of drawing trees as nature grows them, with the root on the bottom. We show edges in T by solid lines directed upward, and edges of G not in T by dashed lines. The sons of a node in T are always ordered from left to right in a diagram.

Note that there may be more than one DFST for a given flow graph G. Thus, the

[1] Do not confuse this definition of "forward" edges in a DFST with the previous one for edges in an rfg. They are not necessarily the same, and context should distinguish which one is meant.

classification of edges of G into the different DFST edge categories is relative to a fixed DFST of G.

Example 1. Let G be the flow graph of Figure 4(a). If we consider the nodes in the order 1, 2, 3, 4, then back to 3, then to 5, we obtain the DFST of Figure 4(b). Edge (1, 3) is a forward edge. Edges (3, 2), (4, 2), and (5, 5) are back edges. Edge (5, 4) is a cross edge. □

LEMMA 11. *If $G = (N,E,n_0)$ is an rfg, then $D = (N,E-B,n_0)$ is a dag of G.*

PROOF. It is shown in [3] that D is acyclic. It is obvious that D is a subgraph of G which is also a flow graph; i.e. there is a path from the initial node to any node of G. Also, D is clearly maximal, in that the addition of any back edge creates a cycle. □

THEOREM 5. *A flow graph $G = (N,E,n_0)$ is reducible if and only if its dag is unique.*

PROOF. ($\Rightarrow$). Assume that G is reducible but that its dag is not unique. Then G has at least two dags, say $D_1 = (N,E_1,n_0)$ and $D_2 = (N,E_2,n_0)$. By Lemma 11, there is no loss of generality in assuming that $E_1 = E - B$.

First, we note that $E_1 \neq E_2$ since we assume D_1 and D_2 are distinct. Next, we observe that $E_1 \not\subseteq E_2$ and $E_2 \not\subseteq E_1$ by the maximality property of a dag of a flow graph. Thus, there is an edge, say (n, m), in $E_2 - E_1$. Edge (n, m) is not a forward edge (removed by T_2) because all forward edges are in E_1. Thus, (n, m) is in B. If $n = m$, then D_2 has a cycle and is not a dag. So, $n \neq m$. Also, $n \neq n_0$ by Lemma 8. Since D_2 is a flow graph, there is a path P from n_0 to n in D_2. If $m = n_0$, then P and edge (n, m) form a cycle. Thus, $m \neq n_0$. P does not contain m, for otherwise D_2 is cyclic. But since (n, m) is in B and $n \neq m$, it follows that m dominates n by Theorem 4. However, the existence of a path P in G from n_0 to n not containing m is a contradiction.

($\Leftarrow$). Let G be a nonreducible flow graph. We shall show that its dag is not unique.

G contains at least one $(*)$. Let V, W, X, Y, Z denote respectively the sets of edges from n_0 to a, a to b, a to c, b to c, and c to b in one particular $(*)$ of G. These five sets are disjoint, and V may be empty.

We now construct two distinct dags, D_1 and D_2, for G. Let G_1 be a DFST of G containing edges $(V \cup W \cup Y \cup Z) - \{(d, b)\}$, where (d, b) is the edge in Z which enters b. Figure 5(b) shows how this DFST can be started. $Z' = Z - \{(d, b)\}$. Similarly, let G_2 be a DFST of G containing edges $(V \cup X \cup Z \cup Y) - \{(e, c)\}$, where (e, c) is the edge in Y which enters c. Figure 5(c) shows how this DFST can be started. $Y' = Y - \{(e, c)\}$. Let D_1 be G_1 plus its nonback edges and let D_2 be G_2 plus its nonback edges.

Clearly, both D_1 and D_2 are dags of G. Also, neither one contains all the edges in $Y \cup Z$, since this would yield a cycle. Each contains a different subset of the edges of $Y \cup Z$. Thus, $D_1 \neq D_2$. □

COROLLARY 5.1. *Let $G = (N,E,n_0)$ be an rfg. Then $D = (N,E-B,n_0)$ is its dag.*

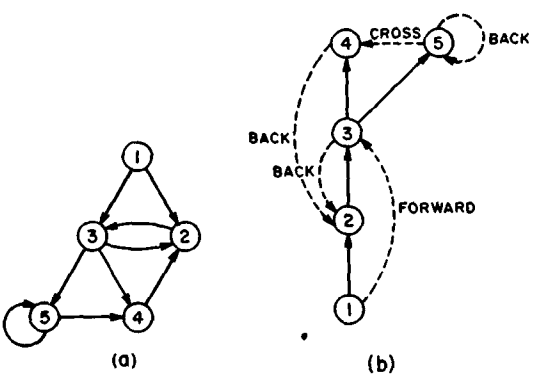

FIG. 4. Example of a DFST

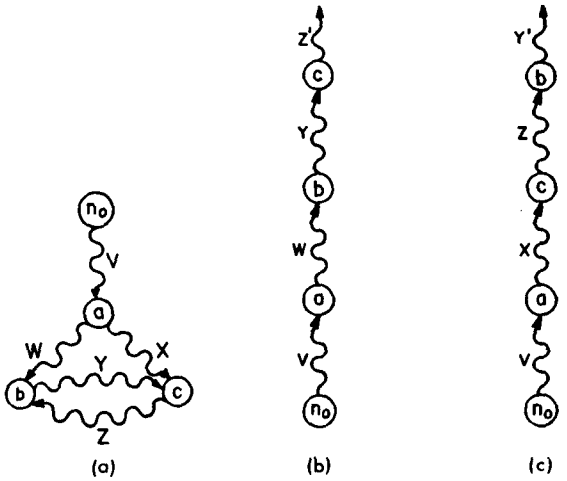

FIG. 5. (a) The subgraph (*) in G. (b) The beginning of D_1. (c) The beginning of D_2

PROOF. It is a dag of G (Lemma 11), and the dag of G is unique (Theorem 5). □

COROLLARY 5.2. *The dag of an rfg G is any DFST of G plus its forward and cross edges. (Alternatively, the backward edges of an rfg G are exactly the back edges of any DFST for G.)*

PROOF. A DFST of G plus its forward and cross edges is a maximal acyclic subflowgraph of G. Since G is an rfg, then its dag is unique (Theorem 5). □

COROLLARY 5.3. *A flow graph $G = (N,E,n_0)$ is reducible if and only if its edge set can be partitioned into two sets E_1 and E_2 such that $D = (N,E_1,n_0)$ is a dag of G and each (n, m) in E_2 has $n = m$ or m dominates n in G.*

PROOF. ($\Rightarrow$). Let $E_2 = B$.

($\Leftarrow$). Any dag for G other than D would include an edge (n, m) such that $n = m$ or m dominates n, an obvious impossibility. □

THEOREM 6. *A flow graph is reducible if and only if its edge set can be partitioned into two sets E_1 and E_2 such that E_1 is a dag D of G and each (n, m) in E_2 has $n = m$ or m dominates n in D.*[2]

PROOF. By Corollary 5.3, it suffices to show that the dominance relations of D and G are the same.

If (n, m) is in B and m dominates n in G, then the removal of (n, m) will not change the dominance relation of the resulting graph.

Conversely, assume G is reducible and its edge set is partitioned into two sets E_1 and E_2 such that E_1 forms a dag D and each (n, m) in E_2 has $n = m$ or m dominates n in D. Let H be a list of all edges (n, m) in E_2. Consider the sequence of flow graphs $D = G_0$, $G_1, G_2, \cdots, G_k = G$, where G_{i+1} is G_i plus the $(i + 1)$-st edge from H.

Inductive hypothesis. If (n, m) is in H and m dominates n in D, then m dominates n in G_i.

Basis. $(i = 0)$. Trivial.

Induction step. $(i > 0)$. Assume the inductive hypothesis and consider G_i. Let (x, y) be the ith edge in H. If $x = y$, we are done. Thus, let $x \neq y$, and suppose in contradiction that m dominates n in G_{i-1} yet m does not dominate n in G_i. Then $m \neq n_0$. Also, there exists a cycle-free path from n_0 to n in G_i which does not contain m but must include edge (x, y). But then y does not dominate x in G_{i-1}, a contradiction. □

[2] Note the distinction between Theorem 6 and Corollary 5.3. Here the dominance relation is calculated in D rather than in G.

4. *Cycles and Single-Entry Loops*

Occasionally, it may be useful to talk about reducibility and what properties it induces on the cycles of a flow graph. One such property is that each cycle C of an rfg has an "entry node" which dominates all the other nodes in C.

Definition. An *entry node of a cycle* C in a flow graph G is a node y in C entered by an edge (x, y) such that x is not in C. If C contains the initial node n_0 of G, then n_0 is also an entry node of C. We call a cycle with one entry node a *single-entry cycle.*

LEMMA 12. *(a) All flow graphs with only single-entry cycles are reducible.*

(b) If a cycle is single-entry, then its entry node dominates all other nodes in the cycle.

(c) If C is a cycle and n a node of C which dominates all other nodes in C, then n is an entry node of C.

PROOF. (a) No such flow graph can contain (*). (b) By the definition of "entry node of a cycle." (c) By the definitions of "flow graph" and "entry node of a cycle." □

THEOREM 7. *A flow graph G is reducible if and only if for each cycle C of G there is an entry node of C which dominates all other nodes in C.*

PROOF. ($\Leftarrow$). If G is nonreducible and we let C be the cycle in (*) from b to c and from c to b, then it is easy to see that no node of C dominates all other nodes of C.

($\Rightarrow$). Let G be an rfg and let C be a cycle of G. There must be a back edge (n, m) in C, where $n \neq m$. Then m dominates n by Theorem 4. Thus, m is an entry node of C. Suppose there were some node $x \neq m$ on C such that m does not dominate x. Then there would be a path from the initial node to x, then along C to n, which does not pass through m. Since m dominates n, it follows that m must dominate x. Thus, m is the desired entry node. □

Each backward edge (n, m) of an rfg defines a "natural loop" which is single-entry. This result is important for the following reason. One purpose of analyzing flow graphs is to determine flow information for code improvement of computer programs. Removing invariant code from "loops" is an example of code improvement, and is usually a very worthwhile operation to perform. But what is a "loop" of a computer program? Also, which "loops" do we choose when there are mutually exclusive alternatives? If code removed from a "loop" has to be replicated at each entrance to the "loop," then single-entry loops are desirable.

Definition. Let $G = (N,E,n_0)$ be an rfg and let (n, m) be a backward edge of G. Furthermore, suppose $R = (N_1,E_1,m)$ is a region containing (n, m). We define *the loop of backward edge (n, m) in R* to be the flow graph $L = (N_2,E_2,m)$ such that: (1) N_2 contains n, m and all nodes x such that there is a path from x to n in R that does not pass through m, (2) $E_2 = E_1 \cap (N_2 \times N_2)$.

We caution the reader not to confuse this loop with the one meaning an edge from a node to itself. Any reference to the word loop in this paper now refers to the above definition.

LEMMA 13. *Any region R headed by m and containing n defines the same loop.*

PROOF. If (n, m) is backward and $n \neq m$, then m dominates n. If there is a path in the rfg from some node r to n that does not pass through m, then m must dominate r. Moreover, r must be in R, since R is a region. Thus, the set N_2 is independent of the region R headed by m and containing (n, m). □

We define an *entry node of a loop* and a *single-entry loop* exactly as we did for a cycle, except that we replace each occurrence of the word "cycle" in that definition by the word "loop."

THEOREM 8. *If $R = (N_1,E_1,m)$ is a region of an rfg and backward edge (n, m) is in E_1, then the loop L of (n, m) is single-entry.*

Proof. If $n = m$, the result is obvious. So, assume that $n \neq m$ and L is not single-entry. Then there is an entry x to L such that $x \neq m$. Since x is in R, m dominates x (Lemma 3). Since x is an entry of L, it is entered from a node y not in L. But, y must be in R and hence in L by the definitions of "region" and "the loop of (n, m)." □

REFERENCES

1. HECHT, M. S., AND ULLMAN, J. D. Flow graph reducibility. *SIAM J. Computing 1*, 2 (June 1972), 188–202.
2. HOPCROFT, J. E., AND ULLMAN, J. D. An $n \log n$ algorithm for detecting reducible graphs. Proc. 6th Annual Princeton Conf. on Information Sciences and Systems, March 1972, pp. 119–122.
3. ULLMAN, J. D. Fast algorithms for the elimination of common subexpressions. *Acta Informatica 2*, 3 (Dec. 1973), 191–213.
4. ALLEN, F. E. Program optimization. *Annual Rev. Automatic Prog., Vol. 5*, Pergamon Press, New York, 1969.
5. ALLEN, F. E. Control flow analysis. *SIGPLAN Notices 5*, 7 (July 1970), 1–19.
6. ALLEN, F. E. A basis for program optimization. Proc. IFIP Conf. 71, Vol. 1, North-Holland Publishing Co., Amsterdam, 1971, 385–390.
7. ALLEN, F. E., AND COCKE, J. Graph-theoretic constructs for program control flow analysis. IBM Res. Rep. RC 3923, T. J. Watson Research Center, Yorktown Heights, N.Y., July 1972.
8. AHO, A. V., AND ULLMAN, J. D. *The Theory of Parsing, Translation and Compiling, Vol. II—Compiling*. Prentice-Hall, Englewood Cliffs, N.J., 1973.
9. SCHAEFER, M. *A Mathematical Theory of Global Program Optimization*. Prentice-Hall, Englewood Cliffs, N. J., 1973.
10. COCKE, J. Global common subexpression elimination. *SIGPLAN Notices 5*, 7 (July 1970), 20–24.
11. KENNEDY, K. A global flow analysis algorithm. *Internat. J. Computer Math. 3* (Dec. 1971), 5–15.
12. TARJAN, R. E. Depth-first search and linear graph algorithms. *SIAM J. Computing 1*, 2 (Sept. 1972) 146–160.

RECEIVED MARCH 1973