A complexity measure based on selection and nesting

by *Huisheng Gong* and *Monika Schmidt*

Keywords: cyclomatic number, program complexity, degree of nesting, forward dominance.

Abstract: Many concepts concerning the quantification of program complexity have been developed during the last few years. One of the most accepted and easy-to-apply complexity measures, McCabe's cyclomatic number, has been discussed and improved in several studies. The cyclomatic number only considers the decision structure of a program. Therefore, this paper proposes a new method for calculating program complexity, the concept of postdomination. This takes into account the degree of nesting of a program. Combining this method and the cyclomatic number, a new complexity measure will be defined.

1. Introduction

Of the many proposals concerning the quantification of program complexity, McCabe's concept (MCCA76) seems to be one of the most accepted and easy-to-apply complexity measures. He proposes that complexity is not closely related to program size, but rather to the number of basic paths through a program-control graph. McCabe's concept uses a directed-graph representation of programs and fundamentals of graph theory to compute the compelxity measure. To start, one draws a control graph (directed graph) for a given program. A node in the graph corresponds to some statements, and an arc or edge corresponds to the possible control flow among the various nodes. For such a graph G, the cyclomatic number v(G) is defined by

$$v(G) = e - n + 2p,$$

where e denotes the number of edges, n the number of nodes, and p the number of connected components.

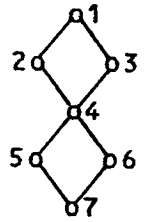
Therefore, a program's complexity, measured by v(G), is assumed to be only a factor of the program's decision structure. However, several anomalies have been found where a higher complexity would be calculated for a program of lesser complexity than for a more complex program.

For improvement, Myers suggests calculating v(G) as a complexity interval (MYER77). The lower bound of the interval is defined as the number of decision statements plus one (examples for decision statements are IF, DO WHILE, and interative DO statements), and the upper bound is the number of individual conditions plus one. Hansen developed a measure that combines the cyclomatic number and an operation count (HANS78). On the other hand, the McCabe measure does not consider the complexity of nesting. Chen describes the complexity of a program with a measure MIN (maximal intersect number; CHEN78). Harrison et al. measures the complexity of programs by the greatest lower bound (GLB) of selection node (HARR81). However, the meaning of the GLB of selection node is not clear.

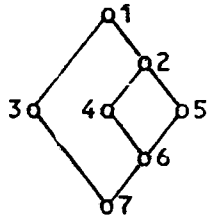
This paper introduces a new method for calculating complexity, the concept of "postdomination," which takes into account the degree of nesting. With the aid of this concept we will rectify McCabe's cyclomatic number and define a new complexity measure.

Therefore, we consider the following five programs and their control graphs:

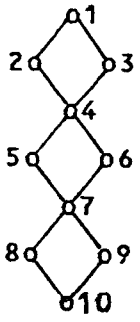
(a)
```
IF P1 THEN S2;
        ELSE S3;
IF P4 THEN S5;
        ELSE S6;
S7;
```



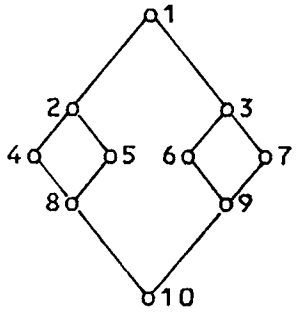
(b)
```
IF P1 THEN
        IF P2 THEN S4;
                ELSE S5;
        S6;
    ELSE S3;
S7;
```



(c)
```
IF P1 THEN S2;
        ELSE S3;
IF P4 THEN S5;
        ELSE S6;
IF P7 THEN S8;
        ELSE S9;
S10;
```



(d)
```
IF P1 THEN
        IF P2 THEN S4;
                ELSE S5;
        S8;
    ELSE
        IF P3 THEN S6;
                ELSE S7;
        S9;
S10;
```



(e) IF P1 THEN

IF P2 THEN

IF P3 THEN S4;

ELSE S5;

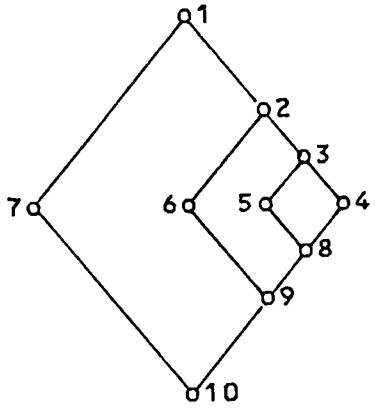
S8;

ELSE S6;

S9;

ELSE S1;

S10;



Obviously, calculating the complexities of the above control graphs by the classical cyclomatic number leads to:

$$v(a) = v(b) = 3;\ v(c) = v(d) = v(e) = 4.$$

It is not possible now to differentiate, for example, between the complexity of programs a and b. Therefore, if we want to distinguish control graphs having the same complexities measured by McCabe's cyclomatic number, we have to consider a new concept, which will be introduced in the following section.

2. Degree of nesting

Let P be a program or a module of a program. The directed graph of P is defined by G = (V,A), where V is a finite nonempty set, and A is a relation on V. Each element in V is called a node, and each pair in A is called an arc or edge. Then

$$V = V_1 + V_2$$

where

$V_1 = \{x \mid x \in V, |\,suc(x)\,| \leq 1\}$; and

$V_2 = \{x \mid x \in V, |\,suc(x)\,| \geq 2\}$;

suc(x) represents the set of (immediate) successors of node x (x is not the end node), suc(x) = {y | (x,y) ∈ A, x,y ∈ V}; |suc(x)| denotes the out-degree of nodes x (the number of elements of suc(x)).

The elements of V_2 are called selection or decision nodes. (Note: The arc between x and y leaves the node x and enters the node y. We say that x is a predecessor of y, and y is a successor of x.)

Any reference to a graph in the remainder of this paper will be to a connected directed graph with a single initial node (a node that has no predecessor).

In the following we will describe the so-called forward dominance relationship that exists in a directed graph and is of interest in control-flow analysis (for the definition of the backward dominance or predominance, see HECH77, p. 55). Let s be the initial node and e the exit node of the graph G.

Definition: If x, y ϵ V, x $\neq$ y, then x forward dominates or postdominates y if and only if every path in G from y to its exit node contains x. Note: x pdom y. For convenience, we let PDOM(y) = {x | x pdom y, x ϵ V} for each y ϵ V. We say that x directly postdominates y if and only if 1) x pdom y, and 2) if z ϵ V, z pdom y, then z pdom x.

Example: Consider the graph of program 1 with s = 1 and e = 7. Then V = {1, 2, 3, 4, 5, 6, 7}, V_1 = {2, 3, 5, 6, 7}, V_2 = {1, 4}, 1 pdom 1, 4 pdom 1, 7 pdom 1, and 4 directly postdominates 1.

The proof of the following lemma will be found in the appendix (see conclusion 5).

Lemma: If x ϵ V_2 , then x has a unique, direct-forward dominator x'.

We will now consider such a forward dominator x' of an element x of V (with the above lemma it exists and is unique). The nodes and edges between x and x' construct a subgraph G'$\subset$G; G' is called control subgraph of node x. Obviously, G' has only one entry node (x) and one exit node (x'). Let N = | V_2 | (| V_2 | represents the number of elements of V_2), n ϵ V_2, and G_n be the control subgraph of node n. Let d_n be the number of selection nodes of G .

Then the degree of nesting of the selection node n is defined as:

$$\epsilon_n = 1 - (1/d_n)$$

Since G_n has a selection node n at least, so $d_n \geq 1$, and $0 \leq \epsilon_n < 1$.

We now define the degree of nesting of a graph G by:

$$\epsilon = (\epsilon_1 + \epsilon_2 + \ldots + \epsilon_N)/N$$

Obviously, $0 \leq \epsilon < 1$.

3. Rectificaiton of v(G)

We define a new complexity measure for programs by:

$$c(G) = v(G) + \epsilon$$

where v(G) is the cyclomatic number, and ϵ is the degree of nesting of G as defined above.

McCabe's complexity measure only takes into account the decision structure of a program and is independent of the nesting structure. Although, for example, v(G) for two different programs is identical; it can be possible that they have a different degree of nesting, but the cyclomatic number doesn't distinguish them. However, adding the degree of nesting implicates a rectification of McCabe's complexity measure.

Calculating the new complexities of programs (a)-(c), given above, leads to:

1) $d_1 = d_2 = 1$, *so* $\epsilon_1 = \epsilon_2 = 0$, $\epsilon = 0$, $c(a) = v(a) = 3$;
2) $d_1 = 2, d_2 = 1$, *so* $\epsilon = 1/4$, $c(b) = 3{,}25$;
3) $d_1 = d_2 = d_3 = 1$, *so* $\epsilon = 0$, $c(c) = v(c) = 4$;
4) $d_1 = 3, d_2 = d_3 = 1$, *so* $\epsilon = 1/9$, $c(d) = 4{,}11$;
5) $d_1 = 3, d_2 = 2, d_3 = 1$, *so* $\epsilon = 5/18$, $c(e) = 4{,}28$.

It is possible now to differentiate between programs (a) and (b) (having the same cyclomatic number),

and between programs (c), (d), and (e) (which also have the same cyclomatic number):

$$c(a) < c(b) \text{ and } c(c) < c(d) < c(e).$$

Example: For twelve examples from McCabe (MCCA76, p. 310 ff.) the cyclomatic number v(G) and the extended complexity measure c(G) will be calculated and compared by their ranks.

Graph G (no.)	v(G)	c(G)	Rank of v(G)	Rank of c(G)
1	2	2	1	1
2	3	3,25	2	2
3	5	5,50	3	3
4	6	6,55	4	4
5	8	8,39	5,5	5
6	8	8,60	5,5	6
7	9	9,46	7	7
8	10	10,48	9	10
9	10	10,17	9	9
10	10	10,09	9	8
11	11	11,56	11	11
12	19	19,42	12	12

5. Appendix

Let G = (V,A,s,e) be a program control graph where V denotes the set of all nodes, A the set of all edges, s the initial node, and e the exit node. There is no loss of generality in assuming that G has exactly one initial node and one exit node.

Definition: If x, y $\in$ V, x $\neq$ y, then x forward dominates or postdominates y if and only if every path in G from y to its exit node contains x. Note: x pdom y. For convenience, we let PDOM(y) = {x | x pdom y, x $\in$ V} for each y $\in$ V.

We say that x directly postdominates y if and only if 1) x pdom y, and 2) if z $\in$ V, z pdom y, then z pdom x.

There are five conclusions for the relation "postdominance."

Conclusion 1: PDOM(e) = {e}.

Proof: Obvious.

Conclusion 2: The postdominance relation of a graph G is a partial ordering.
(Note: A partial ordering on a set S is a reflexive, antisymmetric, and transitive relation on S.)

Proof: 1) Reflexivity: For each $x \in V$, it follows from the definition that x pdom x.

2) Antisymmetry: We claim that if $x, y \in V$, x pdom y, and y pdom x, then $x = y$. First, if $x = e$ or $y = e$, it follows from conclusion 1 that $x = y = e$. Now consider the second case, $x \neq e$ and $y \neq e$. Let (w1, w2, . . ., wk) be any path from y to e. Since x pdom y, there exists an index i_1, $1 < i_1 < k$, such that $wi_1 = x$. Since y pdom x, there exists j_1, $i_1 < j_1 < k$, such that $wj_1 = y$. We can continue to do this. Finally, there exists $i_1, i_2, \ldots, j_1, j_2, \ldots, 1 < i_1 < j_1 < i_2 < j_2 < \ldots < k$, such that $wi_1 = wi_2 = \ldots = x$, $wj_1 = wj_2 = \ldots = y$. Because k is a finite number, it follows that $x = y$.

3) Transitivity: If x pdom y, y pdom z, then x pdom z (obvious).

Conclusion 3: The exit node e of G postdominates all nodes.

Proof: Obvious.

Conclusion 4: If x V, then PDOM(x) forms a chain. (Note: A chain or linear ordering on a set S is a partial ordering on S such that every pair of elements is comparable.

Proof: For x V, PDOM(x) is a partial ordering. We shall show that every pair of elements of PDOM(x) is comparable by the relation pdom. That means for $y, z \in$ PDOM(x) that either y pdom z or z pdom y. Let (w1, w2, . . ., wk) be any cycle-free path with $w1 = x$, $wk = e$. Then there exist some indices i, j, $1 \leq i, j \leq k$, such that $wi = y$, $wj = z$. Without loss of generality, assume $i < j$ ($j < i$). We claim that z pdom y (y pdom x). Suppose that z does not postdominate y. Then there is a path p from y to e that does not contain z. Now (w1, . . ., wi) + p is a path from x to e that does not contain z. But this contradicts the hypothesis that $z \in$ PDOM(x).

Conclusion 5: Every node except e has a unique, direct postdominator.

Proof: Let x be any node and $x \neq e$. Then $\{x,e\} \subset$ PDOM(x). By conclusion 4, the postdominance relation is a chain on PDOM(x) and also on PDOM(x) - {x}. We know that any finite nonemply set has a unique minimum and maximum element. PDOM(x) - {x} is a finite nonempty chain. Therefore, it has a unique least element that must be the direct postdominator of x.

References

(CHEN78) Chen, E., 1978, Program complexity and programmer productivity; IEEE Transaction on Software Engineering, v. SE-4, pp. 187-194.

(HANS78) Hansen, W. J., 1978, Measurement of program complexity by the pair (cyclomatic number, operator count); ACM SIGPLAN Notices, v. 13, no. 3, pp. 29-33.

(HARR81) Harrison, W. A. and Magel, K. I., 1981, A complexity measure based on nesting level; ACM SIGPLAN Notices, v. 16, no. 3, pp. 63-74.

(HECH77) Hecht, M. S., 1977, Flow analysis of computer programs; North-Holland.

(MCCA76) McCabe, T., 1976, A complexity measure; IEEE Transactions on Software Engineering, v. SE-2, no. 4, pp. 308-320.

(MYER77) Myers, G., 1977, An extension to the cyclomatic measure of program complexity; ACM SIGPLAN Notices, v. 12, no. 10, pp. 61-64.