Notes on Dominators and Dominance Frontiers

Bryan Carpenter
NPAC at Syracuse University
Syracuse, NY 13244
{dbc}@npac.syr.edu

August 25, 1999

1 Dominators

A *control flow graph* (CFG) is a directed graph with a unique source node called *Entry*, and a path from *Entry* to every node in the graph. Node X in a CFG is said to *dominate* node Y if every path from *Entry* to Y passes through X. This property is written mathematically as a relation:

$$X \operatorname{DOM} Y$$

The set of nodes dominated by X is sometimes written as $\operatorname{DOM}^{-1}(X)$. Four important properties of the dominance relation follow from the definition.

1. Every node dominates itself:

   $$X \operatorname{DOM} X$$

   This property is called *reflexivity*.

2. If a node X dominates Y and the node Y dominates Z, then X dominates Z:

   $$X \operatorname{DOM} Y \wedge Y \operatorname{DOM} Z \Rightarrow X \operatorname{DOM} Z$$

   This property is called *transitivity*.

3. If a node X dominates Y, then Y does not dominate X unless X and Y are identical:

   $$X \operatorname{DOM} Y \wedge Y \operatorname{DOM} X \Rightarrow X = Y$$

   This property is called *antisymmetry*.

4. If two nodes X and Y both dominate Z, then either X dominates Y or Y dominates X:

   $$X \operatorname{DOM} Z \wedge Y \operatorname{DOM} Z \Rightarrow X \operatorname{DOM} Y \vee Y \operatorname{DOM} X$$

Property 1 is obvious. Property 2 follows because if every path from *Entry* to Z goes through Y and every path from *Entry* to Y goes through X, clearly every path from *Entry* to Z goes through X.

To prove property 3, assume X and Y dominate one another and are distinct. Choose a path from *Entry* that passes through X. Node Y must appear on the path before the first occurrence of X, because Y dominates X. But then X must appear again before Y, because X dominates Y. So X appears on the path before its first appearance, which is a contradiction.

To prove property 4, choose a cycle-free path, p, from *Entry* to Z. Since X and Y both dominate Z, they must both appear on the path. If X and Y are identical the property is trivial, so assume they are distinct. Suppose X appears on p before Y. If X does not dominate Y then there is another path, q, from *Entry* to Y that does not pass through X. But that means that by following q from *Entry* to Y then following the remainder of the original path to Z, we reach Z without passing through X (Figure 1). This contradicts the premise the X dominates Z. We conclude that X *does* dominate Y. Conversely if Y appears before X on p, we conclude that Y dominates X[1].

DOM is a partial order. Properties 1 to 3 establish that the dominance relation is a partial order on the nodes of the control flow graph.

The dominators of any given node are totally ordered. Property 4 implies that furthermore the DOM relation is a *total order* on the nodes that dominate Z—every dominator of Z is related to every other by the DOM relation.

2 Strict and Immediate Dominators

Node X *strictly dominates* node Y if X dominates Y, and X and Y are distinct. This property is written mathematically as:

$$X \,\mathrm{SDOM}\, Y$$

At the end of the last section we noted that the dominators of any particular node form a set that is totally ordered by the dominance relation. Of course the strict dominators of a node Y (being a subset of the dominators) are also totally ordered by DOM. If X is the *least* element of the strict dominators of Y (the one closest to Y), X is called the *immediate dominator* of Y. This is written

$$X \,\mathrm{IDOM}\, Y$$

As an alternative characterization of the immediate dominator, note that if p is *any* path from *Entry* to Y, the immediate dominator of Y is the last node on p that strictly dominates Y.

[1]This proof is taken from "Principles of Compiler Design" by Aho and Ullman, Section 13.1, p443.

Because all dominators of a particular node are totally ordered, they can be arranged in a line with adjacent points related by IDOM. This line is the path reaching the given node through the *dominance tree.* A dominance tree is uniquely associated with any CFG. It is a graph with the same nodes as the original CFG, but the significance of the edges is completely different. The root of the tree is *Entry* and an edge represents an IDOM relationship.

A convenient recursive formula for the dominance relation is

$$X \text{ DOM } Y \Leftrightarrow (X = Y) \vee \left(\underset{P \in \text{Pred}(Y)}{\forall} X \text{ DOM } P \right) \tag{1}$$

As a special case a node dominates itself. If $X \neq Y$ and X does not dominate some predecessor of Y then there is a path from *Entry* through that predecessor to Y that does not pass through X, so X does not dominate Y. This establishes $\Rightarrow$. To establish $\Leftarrow$ when $X \neq Y$ note that every path to Y passes through some predecessor of Y. So if all the predecessors are dominated by X, every path from *Entry* to Y passes through X.

Note that although a node can dominate some of its own predecessors, it cannot dominate all of them, otherwise every path from *Entry* to X would pass through X before reaching X, which is impossible (Figure 2). So if $X = Y$ the second term on the RHS of Equation 1 is false. Conversely if the second term is true $X \neq Y$. This leads to a simple formula for strict dominance:

$$X \text{ SDOM } Y \Leftrightarrow \underset{P \in \text{Pred}(Y)}{\forall} X \text{ DOM } P \tag{2}$$

In words: X strictly dominates Y if and only if X dominates all predecessors of Y. A useful corollary is

Lemma 2.1 *Suppose node Z does* not *lie on a path $p : X \xrightarrow{*} Y$, and Z strictly dominates Y. Then Z strictly dominates* all *nodes on p.*

Proof: Since Z strictly dominates Y, it dominates all predecessors of this node (by Equation 2). In particular it dominates the penultimate node of p. Because this node is distinct from Z (Z does not lie on p), Z also dominates it strictly. Repeat this argument inductively to show that Z strictly dominates *every* node in p. $\square$

3 Dominance Frontiers

The *dominance frontier* of a node X, denoted by DF(X), is the set of nodes, Y, such that X dominates some predecessor of Y, but X doesn't strictly dominate Y:

$$\text{DF}(X) = \left\{ Y \,\middle|\, \left(\underset{P \in \text{Pred}(Y)}{\exists} X \text{ DOM } P \right) \wedge X \,\overline{\text{SDOM}}\, Y \right\} \tag{3}$$

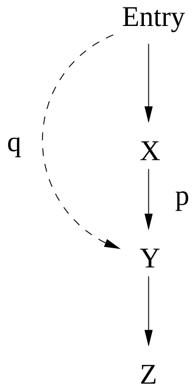


Figure 1: Paths in proof of property 4

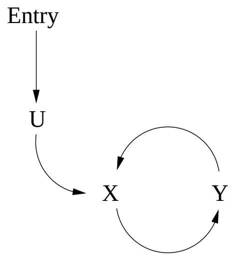


Figure 2: A node X can dominate *some* of its own predecessors (eg $X \, \mathrm{DOM} \, Y$), but it cannot dominate *all* of them (eg $X \, \overline{\mathrm{DOM}} \, U$), because there must be a path from *Entry* to X.

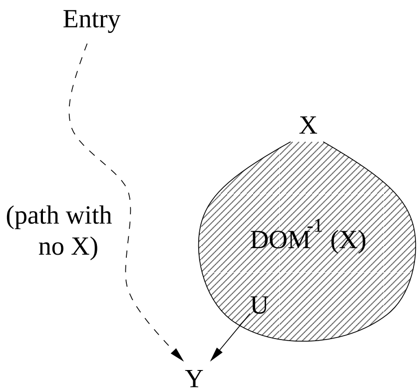


Figure 3: Node Y is in $\mathrm{DF}(X)$ because its predecessor U is dominated by X, but Y is not strictly dominated by X (so, unless $X = Y$, there exists a path from *Entry* to Y with no X on it).

The situation is illustrated in Figure 3. Note that if we follow any path from X, the first node after the startpoint that is *not* strictly dominated by X must belong to $\mathrm{DF}(X)$.

The dominance frontier of a set, S, of nodes is defined as the union of the dominance frontiers of the elements:

$$\mathrm{DF}(S) = \bigcup_{X \in S} \mathrm{DF}(X)$$

The *iterated dominance fronter*, $\mathrm{DF}^+(S)$, of the set of nodes is defined as the limit of the sequence

$$\begin{aligned} \mathrm{DF}_1(S) &= \mathrm{DF}(S) \\ \mathrm{DF}_{i+1}(S) &= \mathrm{DF}(S \cup \mathrm{DF}_i(S)) \end{aligned}$$

The following lemma about iterated dominance frontiers will be important:

Lemma 3.1 *If a path $p : X \xrightarrow{+} Y$ has the property that no node on p strictly dominates its endpoint, Y, then $Y \in DF^+(\{X\})$.*

Proof: Suppose for a contradiction that $Y \notin \mathrm{DF}^+(\{X\})$. Let X' be the last node on the path that *is* in $\mathrm{DF}^+(\{X\})$. If no node on the path is in $\mathrm{DF}^+(\{X\})$, define X' to be X. Clearly no node following X' on p is in $\mathrm{DF}(X')$, otherwise that node would also be in $\mathrm{DF}^+(\{X\})$. So X' strictly dominates the nodes following it, and in particular it strictly dominates Y. This contradicts the premise that no node on p strictly dominates Y, so we can infer $Y \in \mathrm{DF}^+(\{X\})$. □

(In fact the reverse implication also holds, and we can characterize the iterated dominance frontier as follows:

Theorem 3.2 *Node Y is an element of $DF^+(\{X\})$ if and only if there exists a path $p : X \xrightarrow{+} Y$ with the property that no node on p strictly dominates its endpoint, Y.*

To prove "only if" assume $Y \in \mathrm{DF}^+(\{X\})$ and consider Figure 4. Such a path must exist if Y is in $\mathrm{DF}^+(\{X\})$. Suppose there is a node, X', on the path that strictly dominates Y. X' will strictly dominate all nodes in the portion of the path following the last occurrence of X', by Lemma 2.1. Either this occurrence of X' is in the subsequence $(X, X_1, X_2, \ldots)$, or the last member of the subsequence preceding the occurrence strictly dominates X'. In the latter case the transitivity properties of strict dominance mean the preceding member of the sequence strictly dominates all nodes that follow *it* on the path. In either case there is a member of $(X, X_1, X_2, \ldots)$ that strictly dominates all its successors on the path, up to and including Y. So Y cannot be in the dominance frontier of the last member of the subsequence, which contradicts the choice of path. We conclude X' does not exist, so p has the required property[2].)

[2]Note that not *every* path from X to an element of $\mathrm{DF}^+(\{X\})$ has the required property. Figure 5 displays a counter-example.

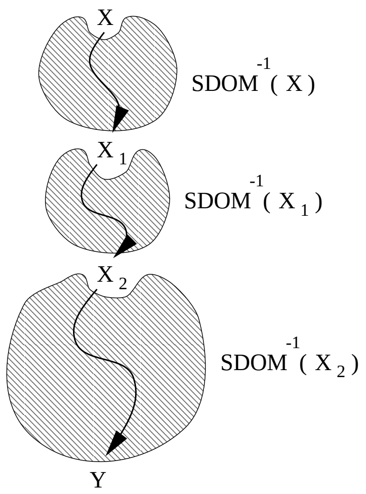


Figure 4: A path to a node on the iterated dominance frontier. Nodes X_1, X_2, Y are in $\mathrm{DF}^+(\{X\})$. Intervening nodes are strictly dominated by nodes on the preceding member of the subsequence (X, X_1, X_2).

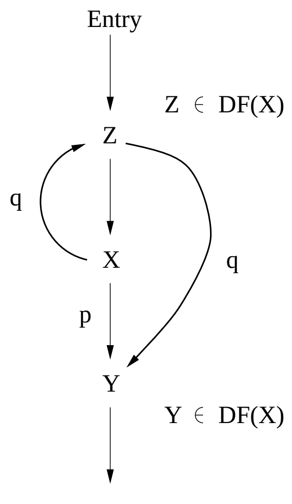


Figure 5: Y belongs to $\mathrm{DF}(X) = \{Y, Z\}$, but there is a path $q : X \to Z \to Y$ including a node, Z, that strictly dominates Y.

4 Join sets

The *join set*, $J(X, Y)$, of nodes X and Y is the set of nodes Z for which there exist two paths $p : X \xrightarrow{+} Z$ and $q : Y \xrightarrow{+} Z$ that are disjoint except at node Z.

Lemma 4.1 *At most one of the two paths p, q includes a node that strictly dominates Z.*

Proof: Suppose a node X' on p strictly dominates Z. Due to the disjointness property, X' does not appear anywhere on q. By Lemma 2.1, X' strictly dominates *every* node in q.

Similarly if a node Y' on q strictly dominates Z, then Y' strictly dominates *every* node in p.

But if both these conditions obtain, we have in particular that $X' \,\mathrm{SDOM}\, Y'$ and $Y' \,\mathrm{SDOM}\, X'$. This contradicts the asymmetry property of dominance. □

Corollary 4.2 *$Z \in DF^{+}(\{X, Y\})$.*

Proof: By the preceding lemma, one of p, q includes no node that strictly dominates Z. Hence, by Lemma 3.1, Z is either an element of $\mathrm{DF}^{+}(\{X\})$ or of $\mathrm{DF}^{+}(\{Y\})$. □

The join of a set, S, of nodes is defined as the union of the join sets of pairs of elements:

$$\mathrm{J}(S) = \bigcup_{X, Y \in S} J(X, Y)$$

The preceding corollary establishes

Lemma 4.3 *$J(S) \subset DF^{+}(S)$.*

Proof: Because $J(X, Y) \subset \mathrm{DF}^{+}(\{X, Y\})$. □

Lemma 4.4 *For any set, S, of nodes containing* Entry, *$DF(S) \subset J(S)$.*

Proof: Consider a node $X \neq \mathit{Entry}$ in S, and any $Y \in \mathrm{DF}(X)$. There is a path from X to Y where all nodes before Y are dominated by X. There is also a path from *Entry* to Y where *none* of the nodes are dominated by X (because there is a path to Y that does not pass through X, so by definition no nodes on this path are dominated by X). Hence the paths are disjoint except at X, and $Y \in J(X, \mathit{Entry})$. □

The *iterated join*, $J^{+}(S)$, of a set of nodes is defined as the limit of the sequence

$$\begin{aligned} J_1(S) &= J(S) \\ J_{i+1}(S) &= J(S \cup J_i) \end{aligned}$$

The importance of the iterated join set is that if S is the set of assignments to some variable, $J^+(S)$ defines the points in the CFG where ϕ terms for that variable must be added in minimal Static Single Assignment form or in Factored Use-Def Chains. We want to prove an equivalence between iterated join sets and iterated dominance frontiers[3].

Theorem 4.5 *If* Entry $\in S$, *then*

$$J^+(S) = DF^+(S).$$

Proof: By Lemma 4.3 and induction on i we show

$$J^+(S) \subset \mathrm{DF}^+(S).$$

The induction step is

$$\begin{aligned} J_{i+1} &= J(S \cup J_i) \;\subset\; J(S \cup \mathrm{DF}^+(S)) \\ &\subset \mathrm{DF}^+(S \cup \mathrm{DF}^+(S)) \;=\; \mathrm{DF}^+(S) \end{aligned}$$

Lemma 4.4 and induction imply

$$\mathrm{DF}^+(S) \subset J^+(S).$$

The induction step is

$$\begin{aligned} \mathrm{DF}_{i+1} &= \mathrm{DF}(S \cup \mathrm{DF}_i) \;\subset\; \mathrm{DF}(S \cup J^+(S)) \\ &\subset J^+(S \cup J^+(S)) \;=\; J^+(S) \end{aligned}$$

□

5 Computation of Dominance Frontiers

Section 3 introduced some basic definitions and lemmas about dominance frontiers, and Section 4 suggested that they may have important applications through their relation to join sets. But to be truly useful, there must be an efficient way to compute them.

For the purposes of this section we assume that the *dominance tree* (see Section 1) has already been computed for the control flow graph. In fact efficient computation of the dominance tree is an interesting and non-trivial problem in itself. For now we just note that a fast, practical algorithm exists[4].

The set of children of node X in the dominance tree is written Children(X). By definition of the dominace tree, this is the set of nodes that X immediately

[3] The remainder of this section is copied from "Efficiently Computing Static Single Assignment Form and the Control Dependence Graph", by Cytron et al, section 4.3.

[4] See "A Fast Algorithm for Finding Dominators in a Flowgraph", T. Lengauer and R. E. Tarjan, TOPLAS, Vol. 1, No. 1, 1979.

dominates. Do not confuse Children(X) with Succ(X)—the set of successors of X in the *control flow graph*. The parent of X in the dominance tree is sometimes written idom(X)—the immediate dominator of X.

The algorithm for computing dominance frontiers is based on the equation

$$\mathrm{DF}(X) = \mathrm{DF}_{\mathrm{local}}(X) \cup \bigcup_{Z \in \mathrm{Children}(X)} \mathrm{DF}_{\mathrm{up}}(Z) \tag{4}$$

where the sets $\mathrm{DF}_{\mathrm{local}}(X)$ and $\mathrm{DF}_{\mathrm{up}}(Z)$ are separate contributions to $\mathrm{DF}(X)$ from the successors and the children of X. They are defined as follows

$$\mathrm{DF}_{\mathrm{local}}(X) \stackrel{\mathrm{def}}{=} \left\{ Y \in \mathrm{Succ}(X) \,\middle|\, X\,\overline{\mathrm{SDOM}}\,Y \right\} \tag{5}$$

$$\mathrm{DF}_{\mathrm{up}}(Z) \stackrel{\mathrm{def}}{=} \left\{ Y \in \mathrm{DF}(Z) \,\middle|\, X\,\overline{\mathrm{SDOM}}\,Y \right\} \tag{6}$$

(to make the second formula self-contained, replace X by idom(Z)).

Proof: Dominance is reflexive, so in equation 5, X dominates a predecessor of Y. So $\mathrm{DF}_{\mathrm{local}}(X) \subset \mathrm{DF}(X)$. Dominance is transitive, so in the equation 6 X dominates any predecessor of Y that Z does; there is at least one such predecessor because $Y \in \mathrm{DF}(Z)$. So $\mathrm{DF}_{\mathrm{up}}(Z) \subset \mathrm{DF}(X)$. Hence the RHS of 4 is contained in $\mathrm{DF}(X)$.

To establish containment of $\mathrm{DF}(X)$ in the RHS, suppose $Y \in \mathrm{DF}(X)$ and P is a predecessor of Y that is dominated by X. We have that $X\,\overline{\mathrm{SDOM}}\,Y$. By consideration of the dominance tree, $P\,\mathrm{DOM}\,X$ implies 1) $P = X$ or 2) there exists some Z such that $X\,\mathrm{IDOM}\,Z$ and $Z\,\mathrm{DOM}\,P$. If 1), then $Y \in \mathrm{DF}_{\mathrm{local}}(X)$. If 2), Z cannot strictly dominate Y, otherwise X would, by the transitivity properties of strict dominance. So Z dominates a predecessor of Y but does not strictly dominate Y, ie $Y \in \mathrm{DF}(Z)$. Since $X\,\overline{\mathrm{SDOM}}\,Y$, Y is a member of $\mathrm{DF}_{\mathrm{up}}(Z)$. □

This establishes the validity of 4. The strict dominance tests in 5 and 6 can be strengthened to immediate dominance tests as follows:

$$\mathrm{DF}_{\mathrm{local}}(X) = \left\{ Y \in \mathrm{Succ}(X) \,\middle|\, X\,\overline{\mathrm{IDOM}}\,Y \right\} \tag{7}$$

$$\mathrm{DF}_{\mathrm{up}}(Z) = \left\{ Y \in \mathrm{DF}(Z) \,\middle|\, X\,\overline{\mathrm{IDOM}}\,Y \right\} \tag{8}$$

This simplifies computations, because immediate dominators will be encoded directly in any sensible representation of the dominance tree.

Proof: To prove 7 we show that if $Y \in \mathrm{Succ}(X)$

$$X\,\mathrm{SDOM}\,Y \Leftrightarrow X\,\mathrm{IDOM}\,Y \tag{9}$$

The $\Leftarrow$ part follows trivially from the definition of IDOM. For the $\Rightarrow$ part suppose $X\,\mathrm{SDOM}\,Y$. Then by consideration of the dominance tree, either 1) $X\,\mathrm{IDOM}\,Y$ or 2) there exists some V such that $X\,\mathrm{IDOM}\,V$ and $V\,\mathrm{SDOM}\,Y$ (ie, either Y is an immediate child of X, or there is some intervening node in the dominance tree). But if V strictly dominates Y it dominates its predecessor

X, and and V DOM X contradicts X IDOM V, due to the antisymmetry of the dominance. So 2) is impossible and we must have X IDOM V.

To prove 8 we show that $Y \in \mathrm{DF}(Z)$ also leads to Equation 9. Again the $\Leftarrow$ part is trivial. The proof that X SDOM $Y \Rightarrow X$ IDOM Y is sketched in Figure 6. □

Equations 4, 7 and 8 immediately give an algorithm for computing the dominance frontiers by post-order (bottom up) traversal of the dominance tree.

6 Reduction to Minimal SSA form

A program is in Static Single Assignment (SSA) form if the program text contains only a single assignment to each variable appearing in the program. Converting a general program to SSA generally involves adding assignments of ϕ-*expressions* to some variables. In a node X that has n CFG predecessors, any ϕ-expression must have n arguments. In a program execution where control enters a node from predecessor i, the run-time value of a ϕ-expression is the value of its i-th argument.

To reach Minimal Static Single Assignment we place a trivial ϕ-term

```
V = φ(V, ..., V)
```

at the entry of any node of the control flow graph belonging to $\mathrm{J}^+(S)$, where S is the set of nodes containing an assignment to V—it includes *Entry* because we can assume each variable gets a default, possibly undefined, value at the start of the program. According to Theorem 4.5 the set of nodes requiring ϕ-terms is identical to $\mathrm{DF}^+(S)$. After placing ϕ-terms, each occurrence of a variable V anywhere in the program is replaced by an occurrence of a new variable V_i, such that the original semantics of the program is unchanged, but the new program text only contains one assignment to V_i.

Lemma 6.1 *Consider a path $p : \mathrm{Entry} \xrightarrow{+} Y$ and assume $Y \notin DF^+(S)$ for some set of nodes, S, that includes* Entry. *Then the last node on p before its endpoint that belongs to $S \cup DF^+(S)$ strictly dominates Y.*

Proof: Similar to proof of Lemma 3.1. Let X' be the last node on the path before the endpoint that belongs to $S \cup \mathrm{DF}^+(S)$. X' exists because *Entry* is in S. No node following X' on p is in $\mathrm{DF}(X')$ (or else it would be in $\mathrm{DF}^+(S)$). The result follows. □

Corollary 6.2 *If a program is in Minimal SSA form and a CFG node Y does* not *have a ϕ-term for V, the last assignment to V before entering Y by any path of program execution occurs in a CFG node that strictly dominates Y.*

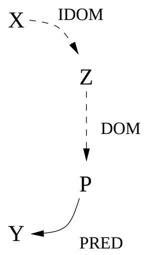
Proof: Immediate from the preceding lemma. Identify S with the set of nodes containing ordinary assignments to V and $\mathrm{DF}^+(S)$ with the set of nodes containing ϕ-terms for V. Since Y does not have a ϕ-term for V it is not in $\mathrm{DF}^+(S)$.

Lemma:

Let X IDOM Z:

and Y ∈ DF (Z):

for some P



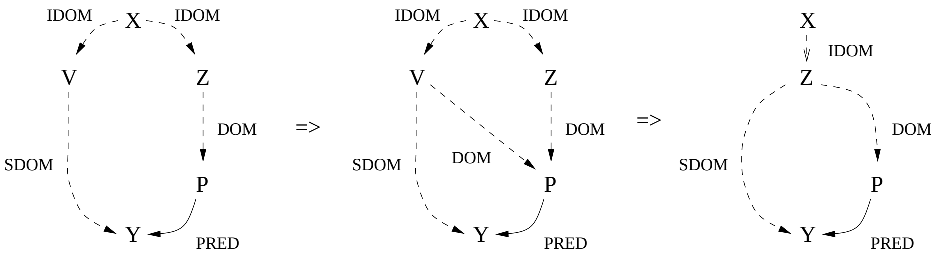
but Z $\overline{\text{SDOM}}$ Y

Then X SDOM Y => X IDOM Y

Proof: Suppose X SDOM Y. There are two possibilities

1) X IDOM Y, or

2) For some V, X IDOM V and V SDOM Y. This implies:



The first inference follows because V SDOM Y implies V dominates all predecessors of Y.
The second inference is because the path X IDOM* P in the dominance tree is unique, so V = Z

We see Z SDOM Y, which contradicts the assumptions. So possibility 1) obtains.

Figure 6: $Y \in \mathrm{DF}(Z)$ and $X\,\mathrm{SDOM}\,Y$ implies $X\,\mathrm{IDOM}\,Y$.

□

This corollary is the basis of an efficient algorithm for renaming variables to convert a program to static single assignment form (Figure 7). Due to the corollary, a pre-order (top down) traversal of the dominance tree achieves the required effect.

```
for each variable V do
    C(V) ← 0
    Version(V) ← 0
end
call SEARCH(Entry)

SEARCH(X) :
    save current state of Version(V) for variables assigned in X

    /* local processing */
    for each statement A in X do
        for each variable V used in an ordinary expression in A do
            replace use of V by V_i, where i = Version(V)
        end
        for each variable V assigned by A do
            C(V) ← C(V) + 1
            Version(V) ← C(V)
            replace def of V by V_i, where i = Version(V)
        end
    end

    /* process φ-terms in successors */
    for each Y ∈ Succ(X) do
        j ← WhichPred(X, Y)
        for each φ-expression in Y do
            replace j-th argument, V, by V_i, where i = Version(V)
        end
    end

    for each Y ∈ Children(X) do
        call SEARCH(Y)
    end

    restore entry state of Version(V) for variables assigned in X
end SEARCH
```

Figure 7: Algorithm for renaming variables in reduction to SSA.