

Compiler Design Notes

Ayush Raina

February 6, 2025

Control Flow Analysis

Why Control Flow Analysis?

Helps to understand structure of Control Flow Graph (CFG), To detect loops in the CFG, Dominator Information, Dominator Frontier Information required for Single Static Assignment (SSA) form. Interval Information used in Data Flow Analysis, Control Dependence Information used in Parallelization.

Dominators

1. We write $(d \text{ dom } n)$ if every path from initial node n_0 to node n passes through node d . A node dominates itself.
2. Node x strictly dominates node y if x dominates y and $x \neq y$.
3. x is immediate dominator of y if x is closest strict dominator of y .

Algorithm to find Dominators

Let us denote the set of Dominators of node n as $D(n) = OUT(n)$. These sets are very helpful to construct the dominator tree quickly.

- Start with n_0 such that $D(n_0) = \{n_0\}$.
- for each $n \in N - \{n_0\}$ do the following:
 - First set $D(n) = N$.
 - Take intersection of all the outsets of the predecessors of n denoted by $IN(n) = \bigcap_{p \in pred(n)} OUT(p)$.
 - $D(n) = IN(n) \cup \{n\}$.
- Repeat until $D(n)$ stabilizes for all n .

Back Edges and Natural Loops

Back Edges: Edges whose head dominates the tail are called back edges. To find the back edges, one easy way to look if opposing edges according to dominator tree is present in the graph.

Natural Loops: Given a back edge (n, d) , the natural loop of the edge is $d \cup$ all the nodes that reach n without going through d . In this case d is the header of the loop and dominates all the nodes in the loop.

Algorithm for finding the natural loop of a back edge

Let the back edge be (n, d) . Then initialize a empty stack. Push n into the stack. Initialize a vector with d in it. Then until the stack is empty, pop the top element and push its predecessors into the stack. If the predecessor is not in the vector, add it to the vector. Continue until the stack is empty. The vector now contains the natural loop.

Depth First Numbering of Nodes in a CFG

We start a $DFS(n)$ from initial node n_0 where n is total number of nodes. We mark a node to n when it is last visit to that node. Last visit means that all the children of that node are visited and now this node will not be visited again. Then we decrement n and continue the process. This way we can number the nodes in a CFG.

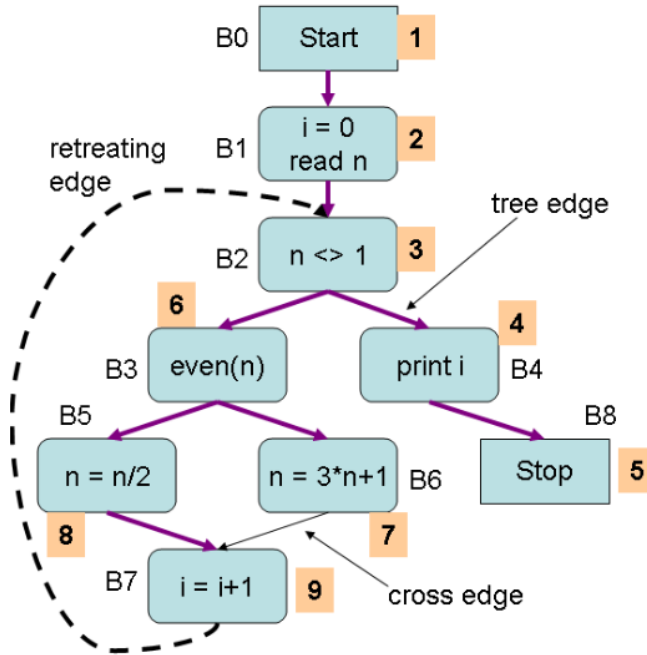


Figure 1: DFS Numbering of Nodes

Reducibility

A flow graph \mathcal{G} is reducible iff every back it can be partitioned into 2 disjoint sets of forward and back edges such that:

- Forward edges form a DAG in which every node is reachable from the initial node.
- All the retreating edges are back edges.
- In an irreducible flow graph, some retreating edges will not be back edges, hence graph of forward edges will contain a cycle.

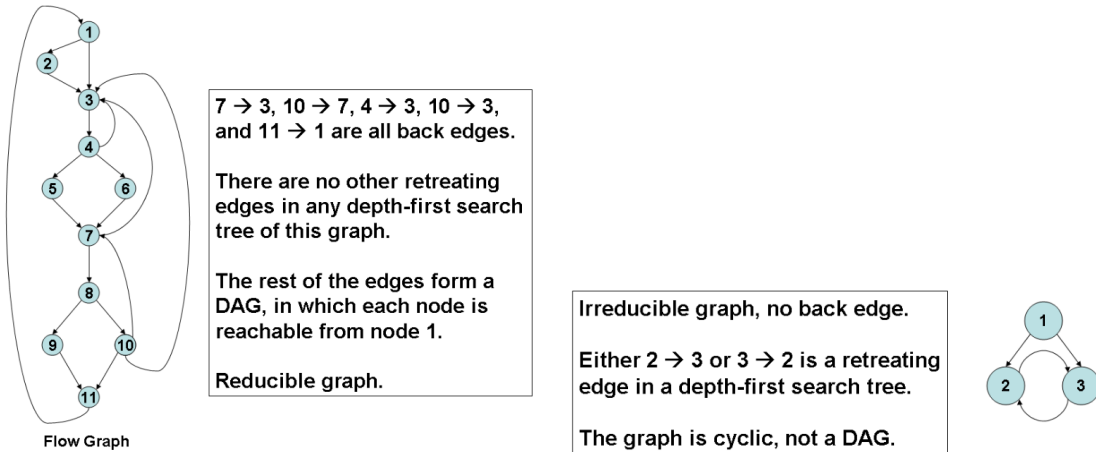


Figure 2: Reducible and Irreducible CFG

Inner Loops

Unless two loops have same header $n \rightarrow d$ (here d is header), they are either disjoint or one is nested inside the other. If natural loop set of one header is subset of the other then there is nesting. Similarly two loops are disjoint if their natural loop sets are disjoint and two loops are identical if their natural loop sets are identical.

If two loops share a header, neither of these may hold and in such a case loops are combined and transformed.

]

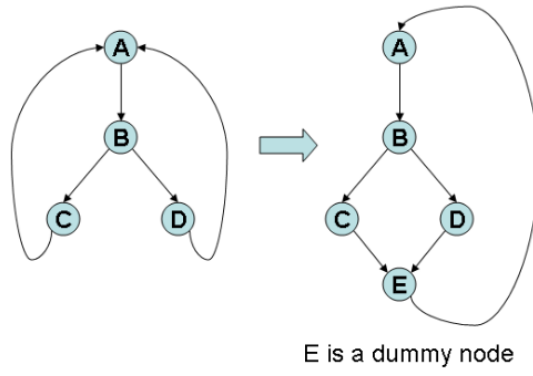


Figure 3: Transformation in case of shared header

Pre Header

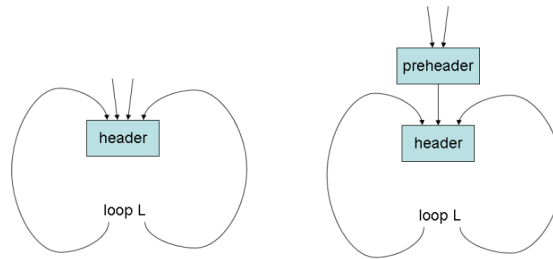


Figure 4: Pre Header

Depth of a CFG and Convergence of DFA Problem

To be continued.