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 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.

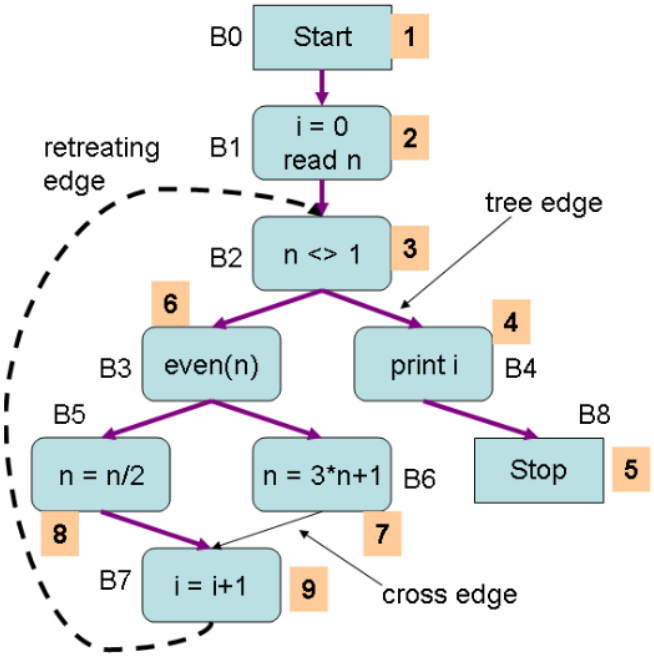

Figure 1: DFS Numbering of Nodes

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.

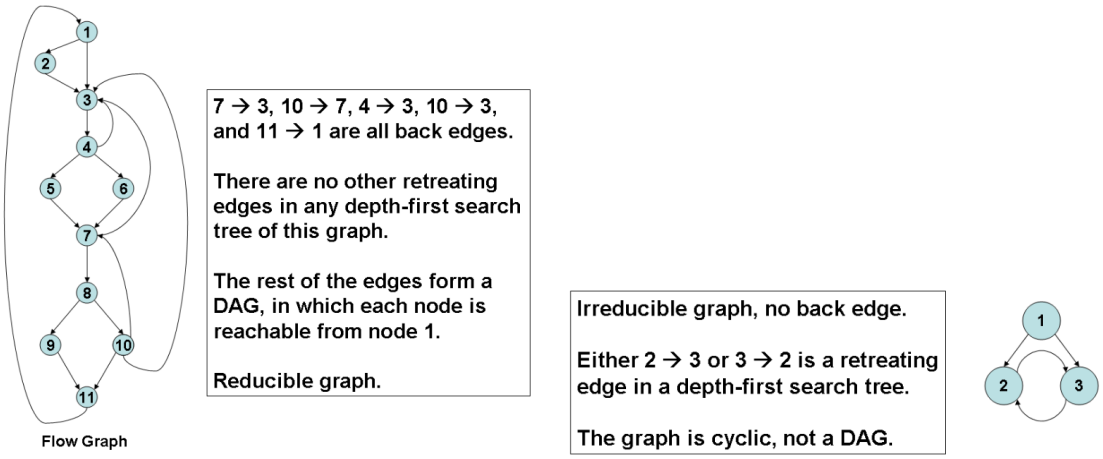

Figure 2: Reducible and Irreducible CFG

Inner Loops

Unless two loops have same header $n \longrightarrow d$ (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.

]

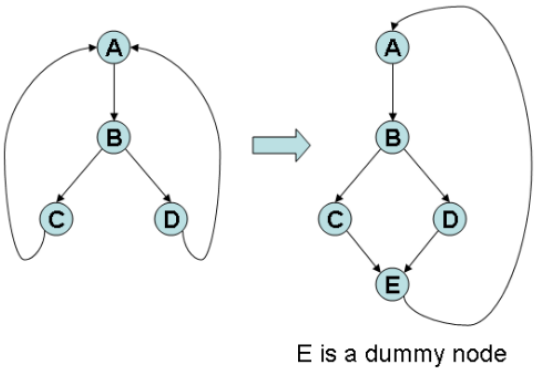

Figure 3: Transformation in case of shared header

Pre Header

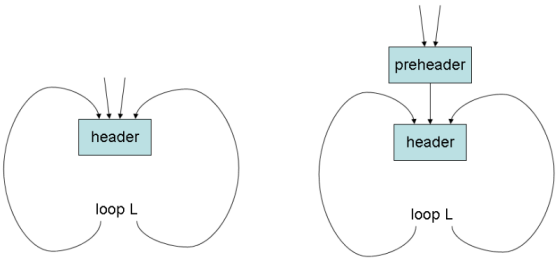

Figure 4: Pre Header

Depth of a CFG and Convergence of DFA Problem

To be continued.

Intervals

Intervals have a header node which dominates all the other nodes in that interval. Given a flow graph $\mathcal{G}$ with initial node n_0.The interval with header n denoted by $I(n)$ is the set defined as:

1. $n \in I(n)$.
2. If $\exists\ m$ such that $Pred(m) \subset I(n)$ then $m \in I(n)$.
3. Nothing else is in $I(n)$.

Constructing $I(n)$:
1. Start with $I(n) = \{n\}$.
2. While there exists a node m such that $Pred(m) \subset I(n)$, do $I(n) = I(n) \cup \{m\}$.

Paritioning a flow graph into disjoint intervals

1. Mark all the nodes as unvisited.
2. Construct $I(n_0)$ and mark all the nodes in $I(n_0)$ as visited.
3. while there is an unvisited node m with atleast one predecessor p which is selected, construct $I(m)$ and mark all the nodes in $I(m)$ as visited and repeat the process.

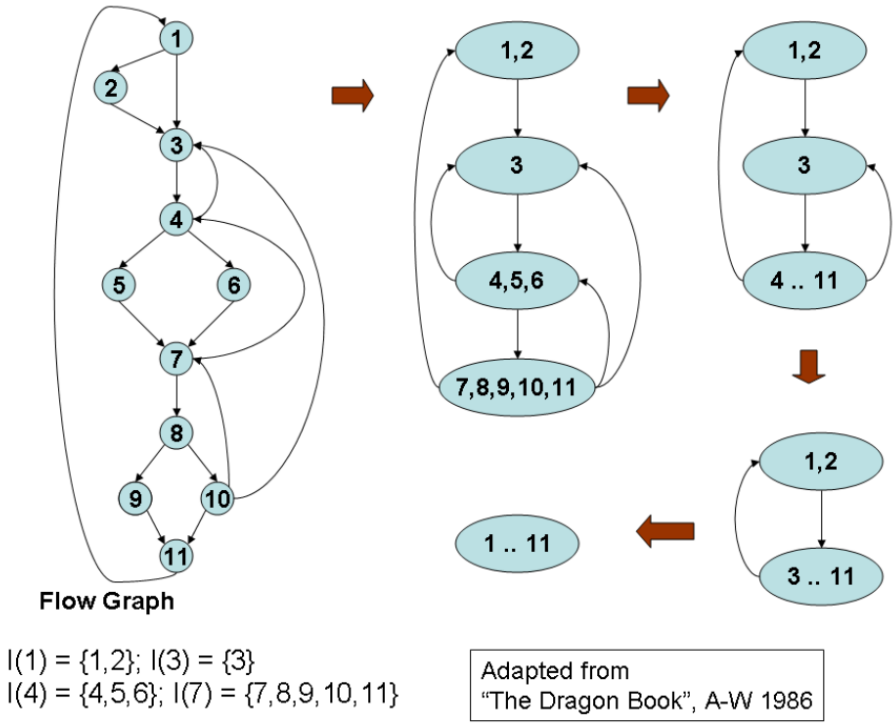

Figure 5: Intervals

In 1st step we got disjoint intervals. Now in the above figure we repeat same procedure further on the interval graph until we reach single node.

Interval Graphs

1. Interval graph is denoted by $I(\mathcal{G})$. Intervals correspond to nodes and interval containing n_0 is the initial node of $I(\mathcal{G})$.
2. If there is an edge from a node in $I(m)$ to the header of interval $I(n)$ then there is an edge from $I(m)$ to $I(n)$ in $I(\mathcal{G})$.
3. We make intervals in interval graph until we reach a limit flow graph which cannot be reduced further.
4. A flow graph is reducible iff its limit flow graph is a single node.

Node Splitting (Extra)

If we reach limit flow graph which is not a single node, we can still proceed further if we split one or more nodes.
1. If a node has k predecessors then we may replace n by k nodes $n_1, n_2, \ldots, n_k$ such that i^{th} predecessor of n becomes predecessor of n_i and all the successors of n become successors of $n_i \forall i \in [1, k]$.
2. After splitting the nodes, we continue reduction process and do node splitting wherever required to reach a single node limit flow graph, however success is not guaranteed.

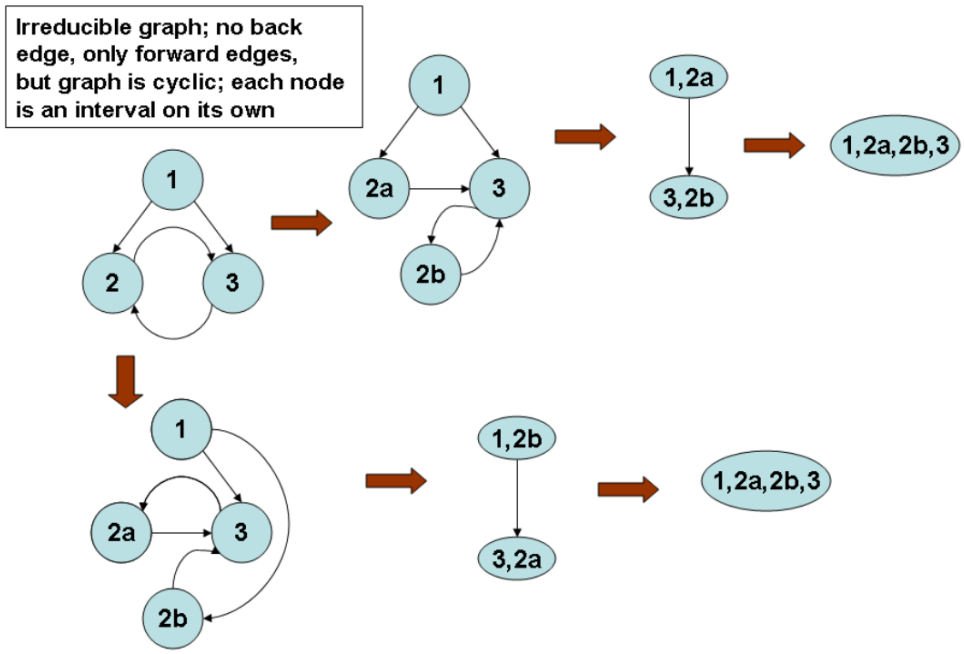

Figure 6: Node Splitting

T1-T2 Transformations and Graph Reduction

T1 Transformation: If n is a node with a loop i.e $(n \to n)$ edge exists then delete that edge.
T2 Transformation: If there is a node n, not the initial node with a unique predecessor m then combine n into m.

By Applying T1,T2 transformations in any order we may reach limit flow graph and node splitting may be required wherever necessary in order to reach a single node limit flow graph.

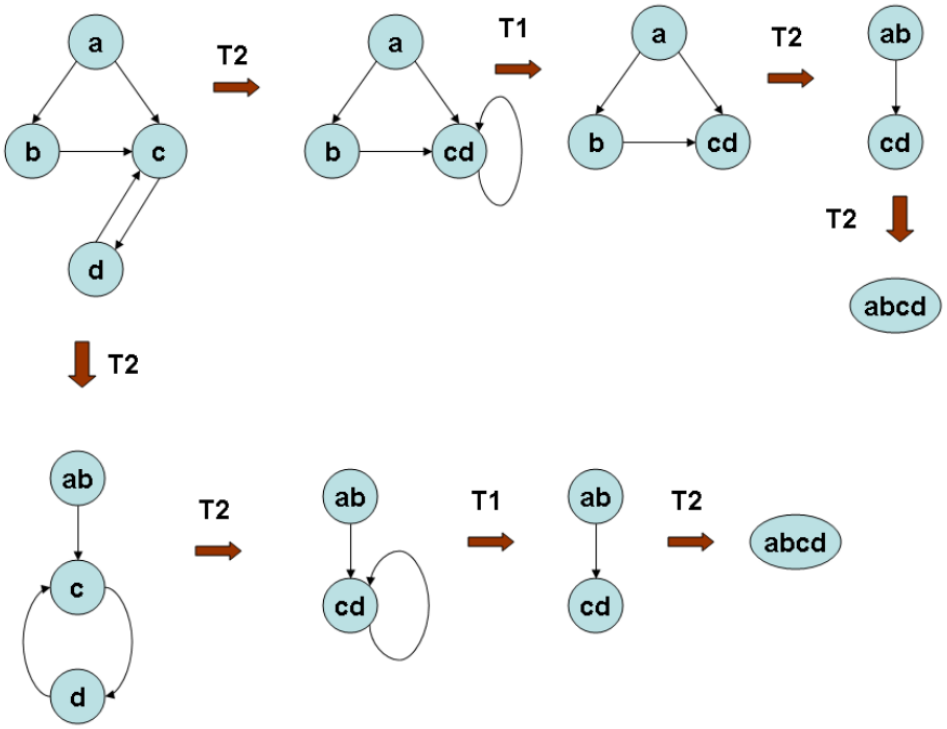

Figure 7: T1-T2 Transformations

One more example of a bigger CFG can be found in slides available on course webpage.

Regions

A set of nodes N that includes a header and it dominates all other nodes in N is called region and all these edges between nodes in N are in the region except (possibly) the edges that enter the header.

All intervals are regions but all regions are not intervals. A region may have multiple exits but an interval has only one exit. As we reduce a flow graph G using $T1 - T2$ transformations, at each step the node obtained is a region.

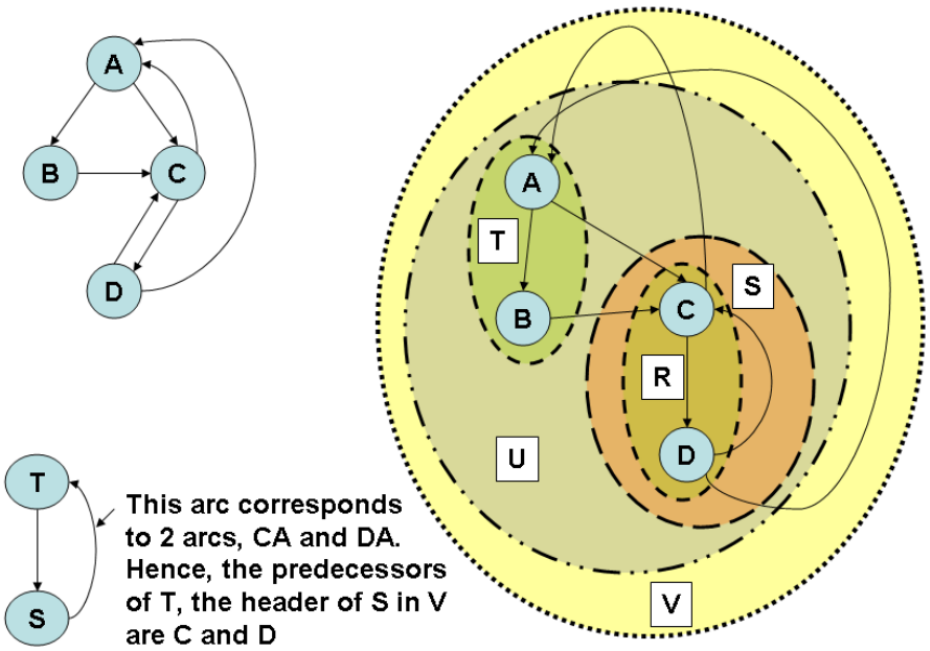

Figure 8: R,S,T,U,V are different regions marked

End of Control Flow Analysis