Control Flow Graphs

We will now discuss flow graphs. These are used for global optimizations (as opposed to optimizations local to basic block).

The nodes of a flow graph are basic blocks.

There is an initial node, $s$, in every flow graph. The node $s$ corresponds to the basic block whose leader is the first statement. An artificial single entry node can be created in there are multiple entries (add an arc from the artificial entry to every entry node).

There is an arc from node $n_1$ to node $n_2$ if the basic blocks associated to $n_2$, $B_2$, can immediately follow at execution time the basic block associated with $n_1$, $B_1$.

That is if either
- There is a conditional or unconditional jump from $B_1$ to (the first statement of) $B_2$, or
- $B_2$ immediately follows $B_1$ in the original order of the program and $B_1$ does not end in an unconditional jump.

**Definition.** A flow graph is a triple $G=(N,A,s)$, where $(N,A)$ is a (finite) directed graph, and there is a path from the initial node, $s \in N$, to every node.

Any node unreachable from $s$ can be deleted without loss of generality.

An exit node in a flow graph has no successors.

Flow graphs are usually *sparse*. That is, $|A| = O(|N|)$. In fact, if only binary branching is allowed $|A| \leq 2|N|$. 
### Dominance in Control Flow Graphs

**Definition 1.** A node $x$ in a flow graph $G$ dominates node $y$ (could be the same as $x$) iff every path in $G$ from $s$ to $y$ contains $x$.

$\text{DOM}(x)$ denotes the set of dominatos of $x$.

**Definition 2.** $x$ properly dominates $y$ if $x$ dominates $y$ and $x \neq y$.

**Definition 3.** $x$ directly dominates $y$ if $x$ properly dominates $y$ and any other node $z$ that properly dominates $y$ also properly dominates $x$.

**Lemma 1.** $\text{DOM}(s) = \{s\}$.

**Lemma 2.** The dominance relation of a flow graph $G$ is a partial ordering. That is dominance is reflexive, antisymmetric, and transitive relation.

**Proof:**

It is reflexive because for any node $x$, $x$ dominates $x$.

It is antisymmetric. If $x$ dominates $y$, then $y$ cannot dominate $x$. Assume otherwise. Then in every path from $s$ to $y$, $x$ has to appear before any occurrence of $y$ and $y$ before any occurrence of $x$. Not possible.

It is transitive. If $x$ dominates $y$ and $y$ dominates $z$ then in every path from $s$ to $z$, $y$ has to appear before $z$ and $x$ before $y$.

**Lemma 3.** The initial node $s$ of a flow graph $G$ dominates all nodes of $G$. 
Lemma 4. The dominators of a node form a chain.

Proof Suppose that two nodes $x$ and $y$ dominate node $z$. Then there is a path from $s$ to $z$ where both $x$ and $y$ appear only once (if not “cut and paste” the path). Assume that $x$ appears first. Then if $x$ does not dominate $y$ there is a path from $s$ to $y$ that does not include $x$ contradicting the assumption that $x$ dominates $z$.

Lemma 5. Every node except $s$ has a unique direct dominator.

Proof. The dominators form a chain. The last node in the chain is the direct dominator (which clearly always exist and is unique).

Lemma 6. A graph of dominators can be created from the nodes $N$ of $G$. There is an arc from $x$ to $y$ iff $x$ directly dominates $y$. This graph is a tree.

Algorithm DOM: Finding Dominators in A Flow Graph

Input: A Flow Graph $G = (N,A,s)$.

Output: The sets DOM($x$) for each $x \in N$.

\[
\begin{align*}
\text{DOM}(s) & := \{s\} \\
\text{forall } n \text { in } N - \{s\} \text{ do } & \text{DOM}(n) := N \text{ od} \\
\text{while} \text{ changes to any } \text{DOM}(n) \text{ occur } & \text{do} \\
\quad \text{forall } n \text { in } N - \{s\} \text{ do } & \\
\quad \quad \text{DOM}(n) := \{n\} \cup \bigcap \text{DOM}(p) \\
\quad \quad \text{od} & \\
\quad \text{od} & \\
\end{align*}
\]
Intervals

The notion of loop in the flow graph is introduced through the concept of interval.

Notice that notions such as cycle and strongly connected component are not appropriate. The former is too fine and the latter too coarse. With cycles loops are not necessarily properly nested or disjoint; and with strongly connected components, there is no nesting.

**Definition 4.** The interval with node $h$ as header, denoted $I(h)$, is the subset of nodes of $G$ obtained as follows:

$$I(h) := \{ h \}$$

**while** $\exists$ node $m$ such that $m \notin I(h)$ and $m \neq s$ and all arcs entering $m$ leave nodes in $I(h)$ **do**

$$I(h) := I(h) + \{ m \}$$

**od**

$I(h)$ is unique and does not depend on the order of selection in the while loop.

**Lemma 7.** The subgraph generated by $I(h)$ is a flow graph.

**Lemma 8.**

(a) Every arc entering a node of the interval $I(h)$ from the outside enters the header $h$.

(b) $h$ dominates every node in $I(h)$
(c) every cycle in $I(h)$ includes $h$

**Proof.**

(a) Consider a node $m \neq h$ that is also an entry node of $I(h)$. Then $m$ could not have been added to $I(h)$

(b) Consider a node $m$ in $I(h)$ not dominated by $h$. Then $m$ could not have been added to $I(h)$.

(c) Suppose there is a cycle in $I(h)$ that does not include $h$. Then no node in the cycle could have been added to $I(h)$, because before any such node could be added the preceding node in the cycle would have to be added.

**Algorithm INT: Partitioning a Flow Graph Into Intervals**

*Input:*
1. A Flow Graph Represented by successor lists.

*Output:*
A set $L$ of disjoint intervals whose union is $N$.

*Intermediate:*
A set of potential header nodes, $H$. 
\( H := \{s\} \)
\( L := \emptyset \)

**while** \( H \neq \emptyset \) **do**

- \( h \) from \( H \)
- Compute \( I(h) \)
- \( L := L + I(h) \)
- \( H := H + \{ \text{nodes with predecessors in } I(h) \text{ but that are not in } H \text{ or in one of the intervals in } L \} \)

**od**

**Theorem 1.** The preceeding algorithm partitions the set of nodes of \( G \).

**Proof** If a node is added to an interval it is not added again. Also, since all nodes are accessible from \( s \), every node is added to \( H \) or to one interval.
**Reducibility**

**Definition 5.** If $G$ is a flow graph, then the derived flow graph of $G$, $I(G)$ is:

(a) The nodes of $I(G)$ are the intervals of $G$

(b) The initial node of $I(G)$ is $I(s)$

(c) There is an arc from node $I(h)$ to $I(k)$ in $I(G)$ if there is any arc from a node in $I(h)$ to node $k$ in $G$.

**Definition 6.** The sequence $G=G_0, G_1, ..., G_k$ is called the derived sequence for $G$ iff $G_{i+1} = I(G_i)$ for $0 \leq i < k$, $G_{k-1} \neq G_k$, $I(G_k) = G_k$. $G_k$ is called the limit flow graph of $G$.

**Definition 7.** A flow graph is *reducible* iff its limit flow graph is a single node with no arc. Otherwise it is called *irreducible*.

Example of irreducible flow graph:
**Definition 7' (From the text).** A flow graph is reducible iff we can partition the edges into two disjoint groups, often called the *forward* edges (not to be confused with the forward edges in a DFST) and the back edges, with the following two properties:

1. The forward edges form an acyclic graph in which every node can be reached from the initial node of G.
2. The back edges consist only of edges whose heads dominate their tails.

**Definition 8.** Interval order for a reducible flow graph is defined recursively as follows:

1. If I(G) is a single node, then an interval order is an order in which nodes may be added to the lone interval G.
2. If G is reducible and I(G) is not a single node, then an interval order is formed by:
   (a) Find an interval order for I(G)
   (b) In the order of (a) substitute for each node of I(G) the nodes of G that make up the corresponding interval, themselves in interval order.
Lemma 9. Interval order topsorts the dominance relation of a reducible flow graph G.

Proof Let $G=G_0, G_1, ..., G_k$ be the derived sequence of G.

We show by induction that if a node $x$ properly dominates $y$, then $x$ precedes $y$ in any interval order of the nodes of $G_i$.

For $i=k$ it is obviously true because there is only one node in $G_k$.

Assume true for $G_{i+1}$ and consider two nodes $x$ and $y$ in $G_i$.

If $x$ and $y$ are in the same interval of $G_i$, then clearly $x$ precedes $y$ in any interval order ($y$ will not be added to the interval until $x$ has been added).

If $x$ and $y$ are in different intervals of $G_i$, then they will be in different nodes, say $X$ and $Y$, of $G_{i+1}$. $X$ has to dominate $Y$ otherwise there would be in $G_i$ a path to $y$ that does not include $x$. By inductive hypothesis $X$ precedes $Y$ in any interval order of $G_{i+1}$. 
**Depth-First Spanning Trees**

**Definition 9.** A depth-first spanning tree of a flow graph G is an ordered spanning tree grown by the DFS algorithm.

**Algorithm DFS: Depth First Search computation of spanning tree**

*Input:* A Flow Graph Represented by successor lists. Nodes are numbered from 1 to n.

*Output:* A Depth First Spanning tree, T, and a numbering of the nodes indicating the reverse of the order in which each node was visited by the algorithm.

DFS(x):
- Mark x "visited"
- while SUC(x) ≠ ∅ do
  - y from SUC(x) /* SUC(x) is an ordered list (left-to-right) of nodes */
  - if y is marked "unvisited" then
    - add (x,y) to T
    - DFS(y)
  - fi
- od
- order(x) = i
- i=i-1

/*** Main Program Follows ****/
T=∅
mark all nodes of G as "unvisited"
i=number of nodes of G
DFS(s)
**Definition 10.** The arcs in G that are not in its depth first spanning tree fall into three categories:

(a) Arcs that go from ancestors to descendants are called forward (advancing in the text) arcs.

(b) From descendants to ancestors or from a node to itself are called back (retreating in the text) arcs.

(c) The other arcs are called cross arcs.

**Observations:**

1. Arc \((x,y)\) is a back arc iff \(\text{order}(x) \geq \text{order}(y)\).
2. Cross arcs go from right to left
3. Every cycle of G contains at least one back arc.