Advanced Compilers
CMPSCI 710
Spring 2003
Computing SSA
Emery Berger
University of Massachusetts, Amherst

More SSA
- Last time
  - dominance
  - SSA form
- Today
  - Computing SSA form

Criteria for Inserting $\phi$ Functions
- Brute-force: Insert one $\phi$ function for each variable at every join point (point in CFG with more than one predecessor)
- Wasteful
- When should we insert $\phi$ function for a variable $a$ at node $z$ of the CFG?
  - Intuitively: add $\phi$ if there are two definitions of $a$ that can reach the point $z$ through distinct paths

Path Convergence Criterion
[Cytron-Ferrante '89]
Insert $\phi$ function for variable $a$ at node $z$ if all the following conditions are true:
1. There is a block $x$ that defines $a$
2. There is a block $y \neq x$ that defines $a$
3. There are non-empty paths $x \rightarrow z$ and $y \rightarrow z$
4. Paths $x \rightarrow z$ and $y \rightarrow z$ don't have nodes in common other than $z$
5. The node $z$ does not appear within both $x \rightarrow z$ and $y \rightarrow z$ prior to the end, but it may appear in one or the other.

Note: The start node contains an implicit definition of every variable.

Iterated Path-Convergence Criterion
The $\phi$ function itself is a definition of $a$.
Therefore the path-convergence criterion is a set of equations that must be satisfied.

while there are nodes $x, y, z$ satisfying conditions 1-5 and $z$ does not contain a $\phi$ function for $a$
do insert $a \leftarrow \phi(a_0, a_1, ..., a_n)$ at node $z$

This algorithm is extremely costly, because it requires the examination of every triple of nodes $x, y, z$ and every path from $x$ to $z$ and from $y$ to $z$.
Can we do better?

Dominance Reviewed
- Node $x$ dominates node $w$ if every path from the start node to $w$ must go through $x$
- Node $x$ strictly dominates node $w$ if $x$ dominates $w$ and $x \neq w$
**Dominance Property of SSA Form**

- In SSA form, definitions dominate uses:
  - If \( x \) is used in \( \phi \) function in block \( n \)
    - def of \( x \) dominates every predecessor of \( n \)
  - If \( x \) is used in non-\( \phi \) statement in block \( n \)
    - def of \( x \) dominates \( n \)
- The dominance frontier of node \( x \):
  - nodes \( w \) such that \( x \) dominates predecessor of \( w \), but \( x \) does not strictly dominate \( w \)

**Example**

What is the dominance frontier of node 5?

First we must find all nodes that node 5 strictly dominates.

A node \( w \) is in the dominance frontier of node 5 if 5 dominates a predecessor of \( w \), but 5 does not strictly dominate \( w \) itself. What is the dominance frontier of 5?

**Dominance Frontier Criterion**

**Dominance Frontier Criterion:**
If a node \( x \) contains a definition of variable \( a \), then any node \( z \) in the dominance frontier of \( x \) needs a \( \phi \) function for \( a \).

Can you think of an intuitive explanation for why a node in the dominance frontier of another node must be a join node?
Example

If a node (12) is in the dominance frontier of another node (5), then there must be at least two paths converging to (12).

These paths must be non-intersecting, and one of them (5,7,12) must contain a node strictly dominated by (5).

Dominator Tree

To compute the dominance frontiers, we first compute the dominator tree of the CFG.

There is an edge from node x to node y in the dominator tree if node x immediately dominates node y.

I.e., x dominates y if y is not dominated by any other dominator of y.

Dominator trees can be computed using the Lengauer-Tarjan algorithm in \(O(E \alpha(E,N))\) time.

Example: Dominator Tree

Control Flow Graph

Local Dominance Frontier

Cytron-Ferrante define the local dominance frontier of a node \(n\) as:

\[
DF_{\text{local}}[n] = \text{successors of } n \text{ in the CFG that are not strictly dominated by } n
\]

Example: Local Dominance Frontier

In the example, what are the local dominance frontiers of nodes 5, 6 and 7?

\[
DF_{\text{local}}[5] = \emptyset
\]

\[
DF_{\text{local}}[6] = \{4,8\}
\]

\[
DF_{\text{local}}[7] = \{8,12\}
\]

Dominance Frontier

Inherited From Its Children

The dominance frontier of a node \(n\) is formed by its local dominance frontier plus nodes that are passed up by the children of \(n\) in the dominator tree.

The contribution of a node \(c\) to its parents' dominance frontier is defined as [Cytron-Ferrante, 1991]:

\[
DF_{\text{up}}[c] = \text{nodes in the dominance frontier of } c \text{ that are not strictly dominated by the immediate dominator of } c
\]
Example: Up Dominance Frontier

In the example, what are the contributions of nodes 6, 7, and 8 to its parent dominance frontier?

First we compute the DF and immediate dominator of each node:

- $DF[6] = \{4, 8\}$, idom(6) = 5
- $DF[7] = \{8, 12\}$, idom(7) = 5
- $DF[8] = \{5, 13\}$, idom(8) = 5

Now, we check for the $DF_{up}$ condition:

- $DF_{up}[6] = \{4\}$
- $DF_{up}[7] = \{12\}$
- $DF_{up}[8] = \{5, 13\}$

$DF_{up}[c] = \text{nodes in } DF[c] \text{ not strictly dominated by } \text{idom}(c)$

Dominance Frontier Inherited From Its Children

The dominance frontier of a node $n$ is formed by its local dominance frontier plus nodes that are passed up by the children of $n$ in the dominator tree. Thus the dominance frontier of a node $n$ is defined as [Cytron-Ferrante 1991]:

$$DF[n] = DF_{local}[n] \cup \bigcup_{c \in DT_{children}[n]} DF_{up}[c]$$

Example: Local Dominance Frontier

What is $DF[5]$?

Remember that:

- $DF_{local}[5] = \emptyset$
- $DF_{up}[6] = \{4\}$
- $DF_{up}[7] = \{12\}$
- $DF_{up}[8] = \{5, 13\}$
- $DT_{children}[5] = \{6, 7, 8\}$

Thus, $DF[5] = \{4, 5, 12, 13\}$

Computing Dominance Frontiers

- Use $DF_{local}$ and $DF_{up}$

```cpp
for X 2 bottom-up traversal of dominance tree
  DF(X) \in Succ(X)
  // local
  if idom(Y) = X then DF(X) \in DF(X) \setminus \{Y\}
  for Z 2 children(X)
    for Y 2 DF(2)
      // up
      if idom(Y) = X then DF(X) \in DF(X) \setminus \{Y\}
```
Join Sets

In order to insert $\phi$-nodes for a variable $x$ that is defined in a set of nodes $S = \{n_1, n_2, \ldots, n_k\}$ we need to compute the iterated set of join nodes of $S$.

Given a set of nodes $S$ of a control flow graph $G$, the set of join nodes of $S$, $J(S)$, is defined as follows:

$$J(S) = \{z \in G \mid \exists \text{ two paths } P_{xz} \text{ and } P_{yz} \text{ in } G \text{ that have } z \text{ as its first common node, } x \in S \text{ and } y \in S\}$$

Iterated Join Sets

Because a $\phi$-node is itself a definition of a variable, once we insert $\phi$-nodes in the join set of $S$, we need to find out the join set of $S \cup J(S)$.

Thus, Cytron-Ferrante define the iterated join set of a set of nodes $S$, $J^+(S)$, as the limit of the sequence:

$$J_i = J(S) \quad J_{i+1} = J(S \cup J_i)$$

Iterated Dominance Frontier

We can extend the concept of dominance frontier to define the dominance frontier of a set of nodes as:

$$DF(S) = \bigcup_{x \in S} DF(x)$$

Now we can define the iterated dominance frontier, $DF^+(S)$, of a set of nodes $S$ as the limit of the sequence:

$$DF = DF(S) \quad DF_{i+1} = DF(S \cup DF_i)$$

Location of $\phi$-Nodes

Given a variable $x$ that is defined in a set of nodes $S = \{n_1, n_2, \ldots, n_k\}$ the set of nodes that must receive $\phi$-nodes for $x$ is $J^+(S)$.

An important result proved by Cytron-Ferrante is that:

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

Thus we are really interested in computing the iterated dominance frontier of a set of nodes.

Algorithms to Compute $\phi$-Node Placement

The algorithm to insert $\phi$-nodes, due to Cytron and Ferrante (1991), computes the dominance frontier of each node in the set $S$ before computing the iterated dominance frontier of the set.

In the worst case, the combination of the dominance frontier of the sets can be quadratic in the number of nodes in the CFG. Thus, Cytron-Ferrante’s algorithm has a complexity $O(N^2)$.

In 1994, Shreedhar and Gao proposed a simple, linear algorithm for the insertion of $\phi$-nodes.

Sreedhar and Gao’s DJ Graph

The DJ graph is used to illustrate the concepts of dominator tree and control flow graph.
Shreedhar-Gao's Dominance Frontier Algorithm

Initialization: \( DF[5] = \emptyset \)

1. \textbf{foreach} \( y \in \text{SubTree}(x) \) \textbf{do}
2. \textbf{if} \((y \rightarrow z \approx J\text{-edge}) \text{ and} \ (z\text{.level} \leq x\text{.level})\) \textbf{then} \( DF[x] = DF[x] \cup z \)

SubTree(5) = \{5, 6, 7, 8\}

There are three edges originating in 5:
\{5 \rightarrow 6, 5 \rightarrow 7, 5 \rightarrow 8\}

but they are all D-edges

What is the \( DF[5] \)?
Shreedhar-Gao’s Dominance Frontier Algorithm

\[
\text{DominanceFrontier}(x) \\
\text{Initialization: } DF[5] = \emptyset \\
0: \text{DF}[x] = \emptyset \\
1: \text{foreach } y \in \text{SubTree}(x) \text{ do} \\
2: \text{if } (y \rightarrow z \rightarrow J\text{-edge}) \text{ and } (z, \text{level} \leq x, \text{level}) \text{ then} \\
3: \text{then } DF[x] = DF[x] \cup z \\
\]

SubTree(5) = \{5, 6, 7, 8\}
There are two edges originating in 7: (7→8, 7→12)
again 8.level > 5.level

A important intuition in Shreedhar-Gao’s algorithm is:

If two nodes \(x\) and \(y\) are in \(S\), and \(y\) is an ancestor of \(x\) in the dominator tree, than if we compute \(DF[x]\) first, we do not need to recompute it when computing \(DF[y]\).

Shreedhar-Gao’s φ-Node Insertion Algorithm

Using the D-J graph, Shreedhar and Gao propose a linear time algorithm to compute the iterated dominance frontier of a set of nodes.

A important intuition in Shreedhar-Gao’s algorithm is:

If two nodes \(x\) and \(y\) are in \(S\), and \(y\) is an ancestor of \(x\) in the dominator tree, than if we compute \(DF[x]\) first, we do not need to recompute it when computing \(DF[y]\).

Dead-Code Elimination in SSA Form

Only one definition for each variable

) if the list of uses of the variable is empty, definition is dead

When a statement \(w \leftarrow x \oplus y\) is eliminated because \(v\) is dead, this statement must be removed from the list of uses of \(x\) and \(y\), which might cause those definitions to become dead.

Thus we need to iterate the dead code elimination algorithm.

Simple Constant Propagation in SSA

If there is a statement \(v \leftarrow c\), where \(c\) is a constant, then all uses of \(v\) can be replaced for \(c\).

A \(\phi\) function of the form \(v \leftarrow \phi(c_1, c_2, \ldots, c_n)\) where all \(c_i\) are identical can be replaced for \(v \leftarrow c\).

Using a work list algorithm in a program in SSA form, we can perform constant propagation in linear time.

In the next slide we assume that \(x, y, z\) are variables and \(a, b, c\) are constants.
Linear Time Optimizations in SSA form

Copy propagation: The statement \( x = a(y) \) or the statement \( x = y \) can be deleted and \( y \) can substitute every use of \( x \).

Constant folding: If we have the statement \( x = a \odot b \), we can evaluate \( c = a \odot b \) at compile time and replace the statement for \( x = c \).

Constant conditions: The conditional if \( a < b \) gotoL1 elseL2 can be replaced for gotoL1 or gotoL2, according to the compile time evaluation of \( a < b \), and the CFG, use lists, adjust accordingly.

Unreachable Code: eliminate unreachable blocks.

Next Time

- Implementing other optimizations with SSA

Single Assignment Form

```
i = 1;
j = 1;
k = 0;
while (k < 100)
{
    if (j < 20)
        j = i;
k = k + 1;
    else
        j = k;
k = k + 2;
}
return j;
```
Example: Dead code elimination

Example: Single Argument $\phi$-Function Elimination

Example: Constant and Copy Propagation

Example: Dead Code Elimination

Example: $\phi$-Function Simplification

Example: Constant Propagation
Example:
Dead Code Elimination

Next Time