Code Motion

66.648 Compiler Design Lecture (04/16/97)

Computer Science

Rensselaer Polytechnic

Lecture Outline

- Ordering of optimizations
- Code Motion
- Examples
- Administration

Orderings of Computation

Among three optimizations we have studied (register allocation, constant folding and common sub-expression elimination), the following order is generally preferred.

1) Constant propagation

2) Common sub-expression elimination and

3) Register allocation

Reasons

- Constant propagation and constant folding may change the register interference graph.
- Constant folding may reveal more common sub-expressions.
- Common sub-expression elimination will not introduce any constant folding.

Loops in control flow graphs

1. Find dominators.
2. Find back edges - this causes loops.
3. Dominator tree of a CFG.

Example

a=1;

while (a < 100) {

if (x < y) { b = 2; c=a +b;}

else { b = 4; c= b-1;}

d=a+c;

e=b+c;

a=a+1; }

Example

Algorithm for computing Dominators

Dom(b1)=b1 // b1 is the start node

for each basic block b != b1 { dom(b) = set of all basic blocks {b1,..,b_k} }

change=true;

while (change) { for each basic block b!=b1 {

old_dom=dom(b); change=false; dom(b)={b1,…,b_k};

for each predecessor p of b { dom(b)=dom(b) intersect dom(p);} dom(b)=dom(b)U{b}.

If (dom(b) !=old_dom) change=true;}}

Loop Invariant Code Motion

If an expression evaluates to the same value in each iteration of the loop, then the evaluation can be performed at the start of the loop instead of every iteration by moving the evaluation of the expression prior to the loop entry.

The savings obtained by this loop-invariant code evaluation is proportional to the average number of times the loop gets executed.

Example

```
sum =0;
I=1;
while (I<=m/n) {
sum = sum+I;
I=I+1;}
print sum;
```

Example

Algorithm for Detection of Loop Invariant Computation

Given: Loop L with header H.

1. Compute use-def chains, the inverse def-use chains.
2. Initially mark each statement S in loop L as variant.
3. change = true; while (change) {

change = false;

for each statement Z= X op Y marked variant in loop L {

Algorithm

If each of X and Y satisfies one of these conditions

a) it is a constant

b) it has all its reaching definitions located outside the loop or

c) it has a single reaching definition located in L which is marked invariant

then mark z= x op y as invariant; change=true;

}

{

Example

```
sum=0; I =1;
while (I<=n) {
t1=5*m;
sum = sum+I;
delta=delta/n;
I=I+delta;
}
```

Example

Code Motion of Loop Invariant Computation

If statement S is detected to be invariant in loop with header node H, then move S to the preheader node for the same loop.

Legality Constraints:

Code motion of loop-invariant statement S:Z=X op Y is legal if:

1. Statement S dominates all exit nodes of the loop
2. No other loop statements assign value to Z
3. All uses of Z in the loop are reached only by the definition in statement S.

Algorithm for Loop-Invariant Code Motion

1. Identify all loop invariant computations in loop
2. Restrict set of loop-invariant computations to those that satisfy the three legality constraints outlined above.
3. Order remaining loop-invariant computations so that S1 appears before S2, if S2 is an invariant statement with the use from S1.
4. Move each statement in the remaining set of loop-invariant computations to pre-header of loop L.

Loop Invariant code motion for multiple loops

Perform loop-invariant code motion algorithm on each loop in procedure being compiled, taking care to process inner loops before outer loops.

Legality constraint: Statement S must dominate all exit nodes of the loop.

(will work only for condition statements in the while loop).

Work-around: Convert the while loop to d-while loop.

Comments and Feedback

Project 3 is out. Please start working. PLEASE do not wait for the due date to come.

We are in chapters 9 and 10. Please read that chapter and read the relevant portion of Java. Please keep studying this material and work exercises.