Lecture 8

Induction Variables and Strength Reduction

I Overview of optimization

II Algorithm to find induction variables

Reference: Muchnick 14.1

Example

```
FOR i = 0 to 100
  A[i] = 0;
```

```
    i = 0
L2: IF i>=100 GOTO L1
     t1 = 4 * i
     t2 = &A + t1
     *t2 = 0
     i = i+1
     GOTO L2
L1:
```

Definitions

1. A basic induction variable is a variable X
   - whose only definitions within the loop are assignments of the form X = X+c or X = X-c, where c is either a constant or a loop-invariant variable.

2. An induction variable is
   - a basic induction variable
   - a variable defined once within the loop, whose value is a linear function of some basic induction variable at the time of the definition.
     $A = c_1 * B + c_2$

3. The FAMILY of a basic induction variable B
   - the set of induction variables A such that each time A is assigned in the loop, the value of A is a linear function of B.

Optimizations

1. Strength reduction:

Let A be an induction variable in family of basic induction variable B ($A = c_1 *B + c_2$)

- Create new variable: `A'`
- Initialization in preheader: `A'=` c_1`*B +` c_2`;`
- Track value of B: add after `B=B+x: A'=A'+x*`c_1`;`
- Replace assignment to A: `A=A'`

Optimizations (cont.)

2. Optimizing non-basic induction variables

- copy propagation
- dead code elimination

3. Optimizing basic induction variables

Eliminate basic induction variables used only for

- calculating other induction variables and loop tests

Algorithm

- Select an induction variable A in the family of B, preferably with simple constants ($A = c_1 * B + c_2$).
- Replace a comparison such as

```
if B > X goto L1
```

  by

  `if (A' > c1 X + c2) goto L1,` assuming c_1 is positive

- if B is live at any exit from the loop, recompute it from A'
  - After the exit, $B = (A' - c_2) / c_1$

II. Basic Induction Variables

- **A BASIC induction variable in a loop L**
  - a variable X whose only definitions within L are assignments of the form X = X+c or X = X-c, where c is either a constant or a loop-invariant variable.
- **Algorithm:** can be detected by scanning L
- Example:

```
k = 0;
for (i = 0; i < n; i++) {
   k = k + 3;
   ... = m
   if (x < y)
      k = k + 4;
   if (a < b)
      m = 2 * k
   k = k - 2
   ... = m
```

Each iteration may execute a different number of increments/decrements!!

Strength Reduction Algorithm

- **Key idea**
  - For each induction variable A, ($A = c_1B+c_2$ at time of definition)
    - variable A' holds expression c_1B+c_2 at all times
    - replace definition of A with A=A' only when executed
- **Result**
  - Program is correct
  - Definition of A does not need to refer to B

Finding Induction Variable Families

- **Let B be a basic induction variable**
  - Find all induction variables A in family of B:
    - $A = c_1 * B + c_2$
      (where B refers to the value of B at time of definition)
- **Conditions**
  - If A has a single assignment in the loop L,
    and assignment is one of:

```
A = B * c   A = c * B
A = B / c   (assuming A is real)
A = B + c   A = c + B
A = B - c
A = c - B
```

  - OR, ... (next page)

Finding Induction Variable Families (cont)

- Let D be an induction variable in the family of B
  ($D = c_1 * B + c_2$)
  - If A has a single assignment in the loop L,
    and assignment is one of:
    ```
    A = D * c   A = c * D
    A = D / c   (assuming A is real)
    A = D + c   A = c + D
    A = D - c
    A = c - D
    ```
  - No definition of D outside L reaches the assignment to A
  - Between the lone point of assignment to D in L
    and the assignment to A,
    there are no definitions of B

Conclusions

- **Precise definitions of induction variables**
- **Systematic idenfication of induction variables**
- **Strength reduction**
- **Clean up:**
  - eliminating basic induction variables
    - used in other induction variable calculations
    - replacement of loop tests
  - eliminating other induction variables
    - standard optimizations