Induction Variables and Strength Reduction

I. Overview of optimization

II. Algorithm to find induction variables

Definitions

- 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. (e.g., i)

- An **induction variable** is
  - a basic induction variable, or
  - a variable defined once within the loop, whose value is a linear function of some basic induction variable at the time of the definition:
    \[ A = c_1 \times B + c_2 \]
    (e.g., t1, t2)

- The **FAMILY** of a basic induction variable B is
  - 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. (e.g., t1, t2 in family of i)

Optimizations

1. **Strength reduction:**
   - A is an induction variable in family of **basic induction variable** B \( (A = c_1 \times B + c_2) \)
     - Create new variable: \( A' \)
     - Initialization in preheader: \( A' = c_1 \times B + c_2 \)
     - Track value of B: add after \( B = B + x \): \( A' = A' + x \times c_1 \)
     - Replace assignment to A: \( A = A' \)

Example

```plaintext
FOR i = 0 to 100
  A[i] = 0;
i = 0
L2: IF i>=100 GOTO L1
t1 = 4 * i
  t1' = 0
t2 = &A + t1
  t2' = &A + 400
  t2' = &A
  t3' = &A + 400
  GOTO L2
  t2' = t2' + 4
  i = i+1
  t1' = t1' + 4
  GOTO L2
L1:
```

Induction variables:

- \( t1' = 4i \)
- \( t2' = 4i + &A \)

L2: IF \( t2' >= t3' \) GOTO L1

- \( t2' = 0 \)
- \( t2' = t2' + 4 \)
- \( t2' = t2' + 4 \)
Optimizations (continued)

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 (\(A = c_1 \cdot B + c_2\)).
     - Replace a comparison such as
       
       \[
       \text{if } B > X \text{ goto L1}
       \]

       with
       
       \[
       \text{if } (A' > c_1 \cdot X + c_2) \text{ goto L1}
       \]

       (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:

```c
k = 0;
for (i = 0; i < n; i++) {
    k = k + 3;
    \dots = m;
    if (x < y)
        k = k + 4;
    if (a < b)
        m = 2 * k;
    \dots = m;
}
```

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

Strength Reduction Algorithm

- Key idea:
  - For each induction variable \(A\), \((A = c_1 \cdot B + c_2\) at time of definition)
    - variable \(A'\) holds expression \(c_1 \cdot B + c_2\) at all times
    - replace definition of \(A\) with \(A'\) only when executed
      \(m\) is only updated when appropriate
- 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 \(B\):
    - \(A = c_1 \cdot B + c_2\)
      (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:
      \[
      A = B * c \quad A = c * B
      \]
      \[
      A = B / c \quad A = B + c \quad A = B - c
      \]
      \[
      A = c + B \quad A = c - B
      \]
    - OR, ... (next page)
Finding Induction Variable Families (continued)

Let D be an induction variable in the family of B \(D = c_1B + c_2\).

Rule 1: If A has a single assignment in the loop L, and assignment is one of:

- \(A = D \times c\)
- \(A = c \times D\)
- \(A = D / c\) (assuming A is real)
- \(A = D + c\)
- \(A = c + D\)
- \(A = D - c\)
- \(A = c - D\)

Rule 2: No definition of D outside L reaches the assignment to A

Rule 3: Between the lone point of assignment to D in L and the assignment to A, there are no definitions of B

Examples

L2: IF i>=100 GOTO L1
    t2 = t1 + 10
    i = i + 1
    t1 = 4 * i
    goto L2
L1:

L2: IF i>=100 GOTO L1
    t1 = 4 * i
    i = i + 1
    t2 = t1 + 10
    i = i + 1
    goto L2
L1:

Summary

- Precise definitions of induction variables
- Systematic identification 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

Wednesday's Class

- Partial Redundancy Elimination  [ALSU 9.5-9.5.2]