Lecture 9
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:
  \[ X = X + c \text{ or } X = X - c, \]
  where \( c \) is either a constant or a loop-invariant variable.

• 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:
    \[ A = c_1 \cdot B + c_2 \]

• 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 \( B \).

Optimizations

1. Strength reduction:
   • Let \( A \) be an induction variable in family of basic induction variable \( B \)
     \( (A = c_1 \cdot B + c_2) \)
     • Create new variable: \( A' \)
     • Initialization in preheader: \( A' = c_1 \cdot B + c_2 \)
     • Track value of \( B \): add after \( B = B + x \): \( A' = A' + x \cdot c_2 \)
     • Replace assignment to \( A \): \( A = A' \)
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 ($A = c_1 \times B + c_2$).
     - Replace a comparison such as
       \[
       \text{if } B > X \text{ goto L1}
       \]
       with
       \[
       \text{if } (A' > c_1 \times X + c_2) \text{ goto L1}
       \]
       (assuming $c_2$ 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:
  
  \[
  k = 0;
  \]
  \[
  \text{for } (i = 0; i < n; i++) \{ \\
  \hspace{1em} k = k + 3; \\
  \hspace{1em} \ldots = m; \\
  \hspace{1em} \text{if } (x < y) \\
  \hspace{2em} k = k + 4; \\
  \hspace{2em} \text{if } (a < b) \\
  \hspace{3em} m = 2 \times k; \\
  \hspace{3em} k = k - 2; \\
  \hspace{3em} \ldots = m;
  \}
  \]

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

Strength Reduction Algorithm

- **Key idea:**
  - For each induction variable $A$, ($A = c_1 \times B + c_2$ at time of definition)
    - variable $A'$ holds expression $c_1 \times B + c_2$ 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 $B$:
    - $A = c_1 \times B + c_2$
    (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 (continued)

- Let D be an induction variable in the family of B (D = c1 * B + c2)
  - 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

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