

Induction Variables

Announcements

- HW1 due Monday
- No office hours Thursday
- No class Friday

Last Time

- Code Motion

Today

- Induction variables

Induction variables

Induction variable identification

- Induction variables
 - Variables whose values form an arithmetic progression
 - Useful for strength reduction, induction variable elimination, and loop transformations

Why bother?

- Automatic parallelization, . . .

Simple approach

- Search for statements of the form, $i = i + c$
- Examine ud-chains to make sure there are no other defs of i in the loop
- Does not catch all induction variables. Examples?

Induction Variable Identification

Types of Induction Variables

- **Basic** induction variables (eg. loop index)
 - Variables that are defined once in a loop by a statement of the form, $i=i+c$ (or $i=i-c$), where c is a constant integer
- **Derived** induction variables
 - Variables that are defined once in a loop as a linear function of another induction variable
 - $k = j + c_1$ or
 - $k = c_2 * j$ where c_1 and c_2 are loop invariant

Example Induction Variables

```
s = 0;
for (i=0; i<N; i++)
    s += a[i];
```

Induction Variable Triples

Each induction variable k is associated with a triple (i, c_1, c_2)

- i is a basic induction variable
- c_1 and c_2 are constants such that $k = c_1 + c_2 * i$ when k is defined
- k belongs to the family of i

Basic induction variables

- their triple is $(i, 0, 1)$
- $i = 0 + 1 * i$ when i is defined

Algorithm for Identifying Induction Variables

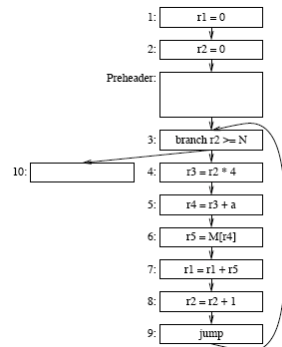
Input: A loop L consisting of 3-address instructions, ud-chains, and loop-invariant information.

Output: A set of induction variables, each with an associated triple.

Algorithm:

1. For each stmt in L that matches the pattern $i = i+c$ or $i=i-c$ create the triple $(i, 0, 1)$.
2. Derived induction variables: For each stmt of L ,
 - If the stmt is of the form $k=j+c_1$ or $k=j*c_2$
 - and j is an induction variable with the triple (x, a, b)
 - and c_1 and c_2 are loop invariant
 - and k is only defined once in the loop
 - and if j is a derived induction variable belonging to the family of i then
 - the only def of j that reaches k must be in L
 - and no def of i must occur on any path between the definition of j and k
 - then create the triple $(x, a+c_1, b)$ for $k=j+c_1$ or $(x, a*c_2, b*c_2)$ for $k=j*c_2$

Example: Induction Variable Detection



Picture from Prof David Walker's CS320 slides

Algorithm for Strength Reduction

Input: A loop L consisting of 3-address instructions and induction variable triples.

Output: A modified loop with a new preheader.

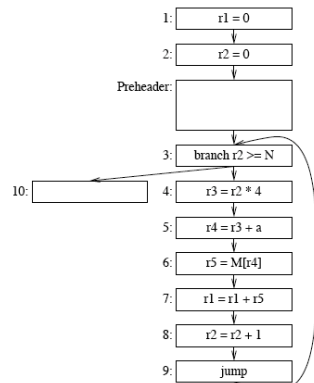
Algorithm:

1. For each derived induction variable j with triple (i, a, b)
 - create a new j'
 - put computation $t=b*c$ in preheader
 - after each definition of i in L , where $i = i+c$ insert $j' = j' + t$
 - replace the definition of j with $j=j'$
 - initialize j' at the end of the preheader to $j' = a+b*i$

Note:

- j' also has triple (i, a, b)
- multiplication has been moved out of the loop

Example: Strength Reduction



Picture from Prof David Walker's CS320 slides
CS553 Lecture

Induction Variables

9

Algorithm for Induction Variable Elimination

Input: A loop L consisting of 3-address instructions, ud-chains, loop-invariant information, and live-variable information.

Output: A revised loop.

Algorithm:

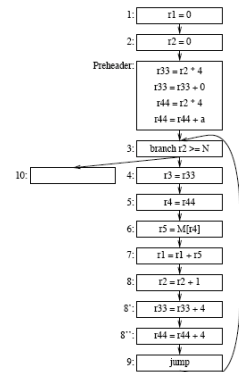
- For each basic induction variable **i**
 - If only uses are to compute other induction variables in its family and in conditional branches
 - Use a triple **(j, c, d)** in family, preferably with $c = 0$
 - Modify each conditional involving **i** so that **b** is used instead
 - if $i \text{ rel op } x \text{ goto } L\#$ becomes if $j \text{ rel op } y \text{ goto } L\#$ with $y = c + d * x$
 - Delete all assignments to the eliminated induction variable
- Apply copy propagation followed by dead code elimination to eliminate copies introduced by strength-reduction.
- Remove any induction variable definitions where the induction variable is only used and defined within that definition.

CS553 Lecture

Induction Variables

10

Example: Induction Variable Elimination



Picture from Prof David Walker's CS320 slides

CS553 Lecture

Induction Variables

11

Summary

Induction variable detection uses

- strength reduction and induction variable elimination
- data dependence analysis, which can then be used for parallelization

Strength reduction

- removes multiplications
- the definition for some derived induction variables no longer depend directly on a basic induction variable

Induction variable elimination

- removes unnecessary induction variables

CS553 Lecture

Induction Variables

12

Next Time

Reading

- Ch 19 through 19.2

Lecture

- SSA