Value Dependence Analysis

Announcement
– Need to make up November 14th lecture, November 30th class will be 2:10-3:50

Last time
– Data dependences for loops

Today
– Value dependence analysis

Versions of the Dependence Problem

Is there a data/memory dependence?
– in general this question is undecidable, for example indirect memory references (ie. x[i][j]) throw a wrench into things
– equivalent to integer programming when the following assumptions are made:
  – structured procedure (i.e. reducible)
  – subscripts, loop bounds, and conditions are affine functions of the loop variables and loop-independent variables
  – loop steps are constant
– Input: IP problem
– Output: yes or no, is there a solution

Are integer programming problems in P or NP?
Versions of the Dependence Problem (cont...)

What is the distance or direction vector of the dependences?
- may require an exponential number of calls to a dependence testing algorithm that only returns yes/no
- Input: IP problem
- Output: distance or direction vector for dependences
- Example outputs: (1,0), (<), (<,=), (<=,>), (>), (0,3)
- Which one of the above dependence vectors is not legal?

What is the dependence relation?
- mapping from one iteration space to another
- Input: Presburger formula (i.e. affine constraints, existential and universal quantifiers, logical operators)
- Output: simplified presburger formula representing dependence relation
- Example input: \{ [i,j] -> [i’,j’] | 1 <= i,j,i’,j’<=10 &
    \[i=i’-1 \& j=j’ \& i<i’ \& j<j’\] \}
- Example output: \{ [i,j] -> [i+1,j] | 1 <= i,j <= 10 \}

Example

Sample code
```plaintext
do i = 1, 6
  do j = 1, 5
    A(2i, j) = A(i, j-1)
  enddo
enddo
```

Dependence
- 2i - i_2 = 0, j_1 = j_2 - 1, solution: YES

Distance/Direction Vector
- (i_1, j_1) + (d_1, d_2) = (i_2, j_2), d_1 = 1, d_2 = ?, d = (<,1)

Dependence Relation
- \{ [i, j] -> [2i, j+1] | 1<=i<=3 \& 1<=j<=4 \}
We are done right?

Sample code

```plaintext
do i = 1, 6
do j = 1, 5
    A(j) = A(j-1) + A(j) + A(j+1)
endo
dendo
```

Memory Deps:

- distance/direction vectors: (0,1), (<,-1), (<,0), (+,1)
- dependence relations:
  `{[i,j] -> [i,j+1] : 1 <= i <= 6 && 1 <= j <= 4}`,  
  `{[i,j] -> [i',j-1] : 1 <= i < i' <= 6 && 2 <= j <= 5}`,  
  `{[i,j] -> [i',j] : 1 <= i < i' <= 6 && 1 <= j <= 5}`,  
  `{[i,j] -> [i',j+1] : 1 <= i < i' <= 6 && 1 <= j <= 4}`

Can we remove all anti, output, and transitive flow dependences?

---

Improve Precision with Array Expansion

Sample code

```plaintext
do i = 1, 6
do j = 1, 5
    A(i,j) = A(i,j-1) + A(i-1,j) + A(i-1,j+1)
endo
dendo
```

Memory Deps:

- distance/direction vectors: (0,1), (1,0), (1,-1)
- dependence relations:
  `{[i,j] -> [i,j+1] : 1 <= i <= 6 && 1 <= j <= 4}`,  
  `{[i,j] -> [i+1,j] : 1 <= i <= 5 && 1 <= j <= 5}`,  
  `{[i,j] -> [i+1,j-1] : 1 <= i <= 5 && 2 <= j <= 5}`

Are these also value dependences?
Value Dependence Analysis

Get rid of false dependences without doing array expansion

Memory Dependence versus Value Dependence [Pugh & Wonnocott 93]
- There is a flow dependence from array access $A(I)$ to $A(J)$ iff
  - $A$ is executed with iteration vector $I$
  - $B$ is executed with iteration vector $J$
  - $A(I)$ writes to the same location as is read by $B(J)$
  - $A(I)$ is executed before $B(J)$
  - there is no write to the location read by $B(J)$ between the execution of $A(I)$ and $B(J)$

Build Dependence Relation for Memory Dependence

Sample code
\[
\text{do } i = 1,6 \\
\text{do } j = 1,5 \\
\quad A(j) = A(j-1) + A(j) + A(j+1) \\
\text{enddo} \\
\text{enddo}
\]

$A(j)$ write to $A(j-1)$ read (+,1):
\[
\{(i,j) \rightarrow [i',j'] : 1 \leq i < i' \leq 6 \land 1 \leq j < 6 \} \text{ access same location}
\]

Simplified:
\[
\{(i,j) \rightarrow [i',j+1] : 1 \leq i < i' \leq 6 \land 1 \leq j \leq 4 \}
\]
union
\[
\{(i,j) \rightarrow [i,j+1] : 1 \leq i \leq 6 \land 1 \leq j \leq 4 \}
\]
Build Dependence Relation for Value Dependence

Sample code

```latex
\begin{verbatim}
do i = 1,6
do j = 1,5
  A(j) = A(j-1)+A(j)+A(j+1)
enddo
enddo
\end{verbatim}
```

$A(j)$ write to $A(j-1)$ read, (0,1):

\[
\{[i,j]\rightarrow [i',j'] : 1\leq i \leq 6 \&\& 1\leq j \leq 5 \land \text{loop bounds}
\land (i<i' \lor (i=i' \&\& j<j')) \land \text{write before read}
\land j=j'-1 \land \text{access same location}
\land \neg\exists ([i',j']): 1\leq i' \leq 6 \&\& 1\leq j' \leq 5 \land \text{no intervening}
\land (i,j)\subset(i',j')\subset(i'',j'') \land (j'\neq j''=1) \}
\]

\text{Simplified:}
\[
\{[i,j]\rightarrow [i,j+1] : 1\leq i \leq 6 \&\& 1\leq j \leq 4}\}
\]

Omega Calculator Manipulates Relations

>./omega/bin/oc

```
# Omega Calculator v1.2 (based on Omega Library 1.2, August, 2000):
R := \{[i,j]\rightarrow [i',j'] : 1\leq i,i' \leq 6 \&\& 1\leq j,j' \leq 5
\land (i<i' \lor (i=i' \&\& j<j')) \land (j=j'-1)
\land \exists ([i',j']): 1\leq i' \leq 6 \&\& 1\leq j' \leq 5
\land (i,j)\subset(i',j')\subset(i'',j'') \}
```

\text{range R;}
\[
\{[i,j]: 1\leq i \leq 6 \&\& 2\leq j \leq 5}\}
\]

\text{domain R;}
\[
\{[i,j]: 1\leq i \leq 6 \&\& 1\leq j \leq 4}\}
\]
Petit Analyses Tiny Programs

```bash
>../omega/bin/petit -Rsimple.petit.out
  -b simple.t

>more simple.petit.out

flow 1: Entry --> 6: a(i-1, j+1)
{[-1, In2] -> [0, In2-1] : 1 <= In2 <= N} union
{[In1, N] -> [In1+1, N-1] : 0 <= In1 <= N-2}
flow 1: Entry --> 6: a(i-1, j+1)
{[In1, In2] -> [In1+1, In2-1] : -1 <= In1 <= N-2 && 1 <= In2 <= N}
output 1: Entry --> 6: a(i, j)
{[In1, In1] -> [In1, In2] : 0 <= In1 < N && 0 <= In2 < N}
flow 1: Entry --> 4: N
{ -> TRUE }
flow 1: Entry --> 5: N
{ -> [i] : 0 <= i < N}
flow 6: a(i, j) --> 6: a(i-1, j+1) {1, -1}
{[i, j] -> [i+1, j-1] : 0 <= i <= N-2 && 1 <= j < N}
exact dd: {
{i, -1}}
flow 6: a(i, j) --> 8: Exit
{[i, j] -> [i, j] : 0 <= i < N && 0 <= j < N}
```

simple.t
-------
integer i, j, N
real a(0:99, 0:99)
for i=0, (N-1) do
  for j=0, (N-1) do
    a(i, j) = a(i-1, j+1)
  endfor
endfor

Concepts

Data Dependence Problems
– is there a dependence?
– what is the dependence distance or direction vector?
– what is the dependence relation?

Value Dependence Analysis gets the same results as full array expansion

Omega tools manipulate data dependence relations
Next Time

Lecture
- Loop transformations