
COMP 515: Advanced Compilation for Vector and Parallel Processors

Vivek Sarkar
Department of Computer Science
Rice University
vsarkar@rice.edu



Announcements

- Schedule for take-home final exam
 - Will be handed at last class, on April 21st
 - Due by 5pm on April 29th
 - I've blocked off 2:30pm - 4:30pm on April 16th (Thurs) for one-on-one discussions of Homework 2
 - Please contact Bob Garcia to schedule a time
 - Homework for next lecture
 - Exercise 11.2
-

Interprocedural Analysis and Optimization

Chapter 11 (contd)

Constant Propagation

```
SUBROUTINE S(A,B,N,IS,I1)
  REAL A(*), B(*)
  DO I = 0, N-1
S0:      A(IS*I+I1) = A(IS*I+I1) + B(I+1)
  ENDDO
END
```

- If $IS=0$ the loop around S0 is a reduction
 - If $IS \neq 0$ the loop can be vectorized
 - **CONST(p)**: set of variables with known constant values on every invocation of p
 - Knowledge of **CONST(p)** useful for interprocedural constant propagation
-

Overview: Interprocedural Analysis

- Examples of Interprocedural problems
- Classification of Interprocedural problems
- Solve two Interprocedural problems
 - Side-effect Analysis
 - Alias Analysis



Interprocedural Problem Classification

- **May and Must problems**
 - MOD, REF and USE are 'May' problems
 - KILL is a 'Must' problem
 - **Flow sensitive and flow insensitive problems**
 - Flow sensitive: control flow info included in analysis
 - Flow insensitive: control flow info is (conservatively) ignored
 - **May and Must classification can apply to call graph edges as well**
-

Overview: Interprocedural Analysis

- Examples of Interprocedural problems
- Classification of Interprocedural problems
- Solve two Interprocedural problems
 - Side-effect Analysis
 - Alias Analysis



Flow Insensitive Side-effect Analysis

- **Assumptions**
 - No procedure nesting
 - All parameters passed by reference
 - Size of the parameter list bounded by a constant,
- We will formulate and solve MOD(s) problem

Solving MOD

$$MOD(s) = DMOD(s) \cup \bigcup_{x \in DMOD(s)} ALIAS(p, x)$$

- **DMOD(s)**: set of variables which are directly modified as side-effect of call at s (ignoring aliases)

$$DMOD(s) = \{v \mid s \Rightarrow p, v \xrightarrow{s} w, w \in GMOD(p)\}$$

- **GMOD(p)**: set of global variables and formal parameters w of p that are modified, either directly or indirectly as a result of invocation of p
 - Global variables are modeled as special “parameters” in this formulation
-

Example: DMOD and GMOD

```
S0:      CALL P(A,B,C)

...

SUBROUTINE P(X,Y,Z)
  INTEGER X,Y,Z
  X = X*Z
  Y = Y*Z

END
```

- $GMOD(P) = \{X, Y\}$
 - $DMOD(S0) = \{A, B\}$
-

Solving GMOD

- $GMOD(p)$ contains two types of variables
 - Variables explicitly modified in body of P : This constitutes the set $IMOD(p)$
 - Variables modified as a side-effect of some procedure invoked in p
 - Global variables are viewed as parameters to a called procedure

$$GMOD(p) = IMOD(p) \cup \bigcup_{s=(p,q)} \{z \mid z \xrightarrow{s} w, w \in GMOD(q)\}$$

Solving GMOD

- The previous iterative method may take a long time to converge
 - Problem with recursive calls

```
SUBROUTINE P(F0,F1,F2,...,Fn)
  INTEGER X,F0,F1,F2,...,Fn
  ...
S0:   F0 = <some expr>
  ...
S1:   CALL P(F1,F2,...,Fn,X)
  ...
END
```

Solving GMOD

- Decompose $GMOD(p)$ differently to get an efficient solution
- Key: Treat side-effects to global variables and reference formal parameters separately

$$GMOD(p) = \boxed{IMOD^+(p)} \cup \boxed{\bigcup_{s=(p,q)} GMOD(q) \cap \neg LOCAL}$$

Solving for IMOD+

- $x \in IMOD^+(p)$ if
 - $x \in IMOD(p)$ or
 - $x \xrightarrow{s} z, z \in GMOD(q), s = (p, q)$ and x is a formal parameter of p
- **Formally defined**

$$IMOD^+(p) = IMOD(p) \cup \bigcup_{s=(p,q)} \{z \mid z \xrightarrow{s} w, w \in RMOD(q)\}$$

- **RMOD(p):** set of formal parameters in p that may be modified in p , either directly or by assignment to a reference formal parameter of q as a side effect of a call of q in p
-

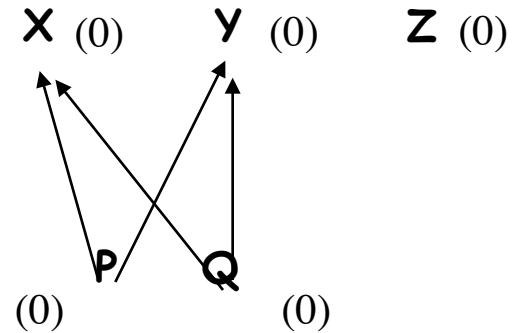
Solving for RMOD

- RMOD(p) construction needs binding graph $G_B=(N_B, E_B)$
 - One vertex for each formal parameter of each procedure
 - Directed edge from formal parameter, f_1 of p to formal parameter, f_2 of q if there exists a call site $s=(p,q)$ in p such that f_1 is bound to f_2
 - Use a marking algorithm to solve RMOD
-

Solving for RMOD

```
SUBROUTINE A(X,Y,Z)
  INTEGER X,Y,Z
  X = Y + Z
  Y = Z + 1
END
```

```
SUBROUTINE B(P,Q)
  INTEGER P,Q,I
  I = 2
  CALL A(P,Q,I)
  CALL A(Q,P,I)
END
```

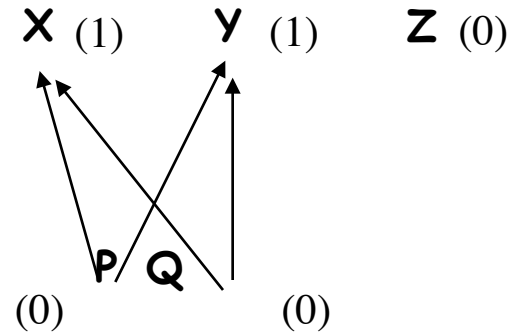


- $IMOD(A) = \{X, Y\}$
- $IMOD(B) = \{I\}$

Solving for RMOD

```
SUBROUTINE A(X,Y,Z)
  INTEGER X,Y,Z
  X = Y + Z
  Y = Z + 1
END
```

```
SUBROUTINE B(P,Q)
  INTEGER P,Q,I
  I = 2
  CALL A(P,Q,I)
  CALL A(Q,P,I)
END
```

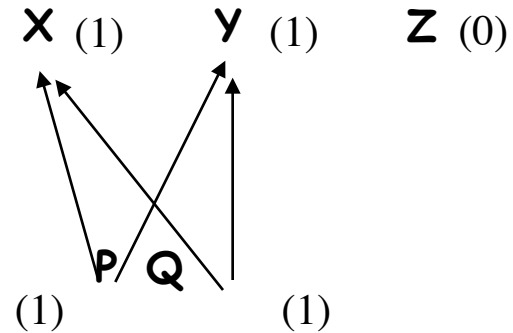


- $IMOD(A) = \{X, Y\}$
 - $IMOD(B) = \{I\}$
 - $Worklist = \{X, Y\}$
-

Solving for RMOD

```
SUBROUTINE A(X,Y,Z)
  INTEGER X,Y,Z
  X = Y + Z
  Y = Z + 1
END
```

```
SUBROUTINE B(P,Q)
  INTEGER P,Q,I
  I = 2
  CALL A(P,Q,I)
  CALL A(Q,P,I)
END
```



- $\text{RMOD}(A) = \{X, Y\}$
- $\text{RMOD}(B) = \{P, Q\}$
- Complexity:

$$O(N_B + E_B)$$

$$N_B \leq \mu N \quad E_B \leq \mu E$$

$$O(N + E)$$

Solving for IMOD⁺

- After gathering RMOD(p) for all procedures, update RMOD(p) to IMOD⁺(p) using this equation

$$IMOD^+(p) = IMOD(p) \cup \bigcup_{s=(p,q)} \{z \mid z \xrightarrow{s} w, w \in RMOD(q)\}$$

- This can be done in $O(NV+E)$ time
-

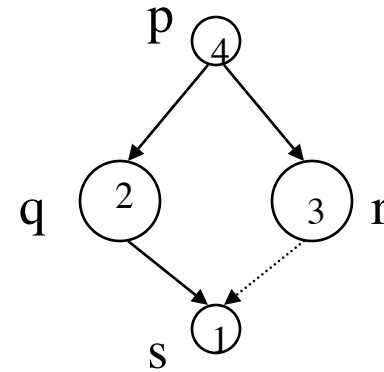
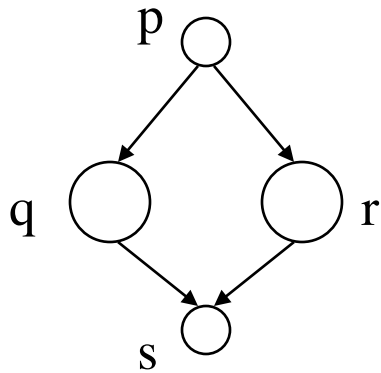
Solving for GMOD

- After gathering $IMOD^+(p)$ for all procedures, calculate $GMOD(p)$ according to the following equation

$$GMOD(p) = IMOD^+(p) \cup \bigcup_{s=(p,q)} GMOD(q) \cap \neg LOCAL$$

- This can be solved using a DFS algorithm based on Tarjan's SCR algorithm on the Call Graph
-

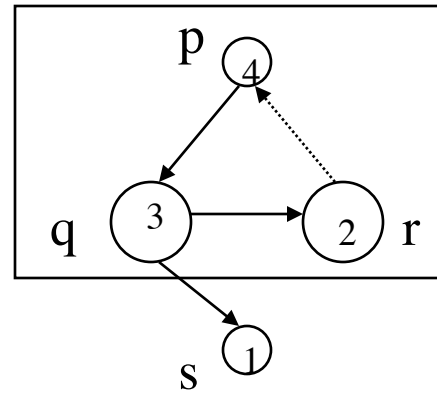
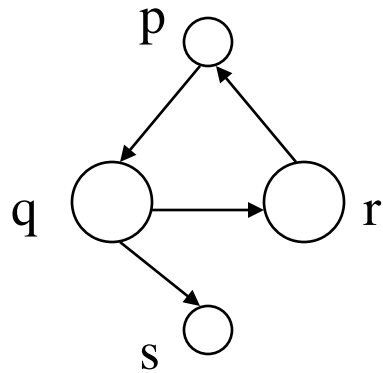
Solving for GMOD



Initialize $GMOD(p)$ to $IMOD^+(p)$ on discovery

Update $GMOD(p)$ computation while backing up

Solving for GMOD



Initialize $GMOD(p)$ to $IMOD^+(p)$ on discovery

Update $GMOD(p)$ computation while backing up

For each node u in a SCR update $GMOD(u)$ in a cycle

$O((N+E)V)$ Algorithm

Overview: Interprocedural Analysis

- Examples of Interprocedural problems
- Classification of Interprocedural problems
- Solve two Interprocedural problems
 - Side-effect Analysis
 - Alias Analysis



Alias Analysis

- Recall definition of $MOD(s)$

$$MOD(s) = DMOD(s) \cup \bigcup_{x \in DMOD(s)} ALIAS(p, x)$$

- Task is to
 - Compute $ALIAS(p, x)$
 - Update $DMOD$ to MOD using $ALIAS(p, x)$

Update DMOD to MOD

```
SUBROUTINE P
  INTEGER A
S0:   CALL S(A,A)
END
```

```
SUBROUTINE S(X,Y)
  INTEGER X,Y
S1:   CALL Q(X)
END
```

```
SUBROUTINE Q(Z)
  INTEGER Z
  Z = 0
END
```

$G\text{MOD}(Q)=\{Z\}$

$D\text{MOD}(S1)=\{X\}$

$M\text{OD}(S1)=\{X,Y\}$

Naïve Update of DMOD to MOD

```
procedure findMod(N,E,n,IMOD+,LOCAL)
  for each call site s do begin
    MOD[s]:=DMOD[s];
    for each x in DMOD[s] do
      for each v in ALIAS[p,x] do
        MOD[s]:=MOD[s] ∪ {v};
      end
    end
  end
end findMod
```

- $O(EV^2)$ algorithm; need to do better
-

Update of DMOD to MOD

- Key Observations

- Two global variables can never be aliases of each other. Global variables can only be aliased to a formal parameter
- $ALIAS(p,x)$ contains less than μ entries if x is global
- $ALIAS(p,f)$ for formal parameter f can contain any global variable or other formal parameters ($O(V)$)

Update of DMOD to MOD

- Break down update into two parts
 - If x is global we need to add less than μ elements but need to do it $O(V)$ times
 - If x is a formal parameter we need to add $O(V)$ elements but need to do it only $O(\mu)$ times
 - This gives $O(\frac{V}{\mu}) = O(V)$ update time per call, implying $O(VE)$ for all calls
-

Computing Aliases

- Compute $A(f)$: the set of global variables a formal parameter f can be aliased to
 - Use binding graph [with cycles reduced to single nodes]
 - Compute $\text{Alias}(p, g)$ for all global variables $g \rightarrow f_0 \rightarrow f_1 \rightarrow \dots \rightarrow f_n = f$
 - Expand $A(f)$ to $\text{Alias}(p, f)$ for formal parameters
 - Find alias introduction sites (eg. `CALL P(X,X)`)
 - Propagate aliases
-

Summary

- Some Interprocedural problems and their application in parallelization and vectorization
 - Classified Interprocedural problems
 - Solved the modification side-effect problem and the alias side-effect problem
-