Lazy Array Data-Flow Dependence Analysis

Anil Kumar
Bhaskar reddy
Narasimham.M
Ranjith kumar

Computer Science and Engineering
IIT Bombay

November 10, 2005

Outline

- Introduction
- Representation and Definitions
- Algorithm for finding Value based dependencies
- Non Affine fragment

Introduction

- Dependencies
- There is a flow dependence from an array access A(I) to an array access B(I') iff
  - A is executed with iteration vector I,
  - B is executed with iteration vector I',
  - A(I) writes to the same location as is read by B(I'),
  - A(I) is executed before B(I')
  - there is no write to the location read by B(I') between the execution of A(I) and B(I').
- Memory based Vs Value based Dependencis

QUAST/LWT

```
    INTEGER rs, p, q, i
    DO rs = 1, nrs
      DO q = 1, np
        DO i = 1, mb
S0:       XRSIQ(i,q) = 0
        END DO
      END DO
      DO p = 1, np
        DO q = 1, p
          ...
          DO i = 1, mb
S1:         XRSIQ(i,q) = XRSIQ(i,q) + ...
S2:         XRSIQ(i,p) = XRSIQ(i,p) + ...
          END DO
        END DO
      END DO
      ...
    END DO
```

(a) Fragment of subroutine OLDA from TRFD

for the statement S_2 in Figure 1(a) is $\mathrm{Src}(S_2[p,q,i]) =$

$$\left[\begin{array}{llll} \text{if} & q=p & \text{then} & S_1[p,q,i] \\ \text{elseif} & q\geq 2 & \text{then} & S_2[p,q-1,i] \\ \text{else} & & \text{then} & S_0[p,i] \end{array}\right. .$$

QUAST Contd...

```
    DO i = 0, M
      DO j = 0, N
S1:     A(2*i+j) = ...
      END DO
    END DO
S2: ... = A(k)
```

$$
Src(S_2) = \left[\begin{array}{l}
\texttt{if} \quad (-k + 2M + N \geq 0) \\
\texttt{then} \quad \texttt{if} \quad (k - 2M \geq 0) \\
\qquad\qquad \texttt{then} \quad S_1[M, k - 2M] \\
\qquad\qquad \texttt{else} \quad \texttt{if} \quad ((-k + N + 2(k \div 2)) \geq 0) \\
\qquad\qquad\qquad\qquad \texttt{then} \quad S_1[k \div 2, k - 2(k \div 2)] \\
\qquad\qquad\qquad\qquad \texttt{else} \quad \perp \\
\texttt{else} \quad \perp
\end{array} \right.
$$

Figure 6: Program and source function represented as a quast

Dependence Relation Representation

```
    INTEGER rs, p, q, i
    DO rs = 1, nrs
      DO q = 1, np
        DO i = 1, mb
S0:       XRSIQ(i,q) = 0
        END DO
      END DO
      DO p = 1, np
        DO q = 1, p
          ...
          DO i = 1, mb
S1:         XRSIQ(i,q) = XRSIQ(i,q) + ...
S2:         XRSIQ(i,p) = XRSIQ(i,p) + ...
          END DO
        END DO
      END DO
      ...
    END DO
```

(a) Fragment of subroutine OLDA from TRFD

$$\begin{array}{lll} S_1[p,q,i] & \rightarrow S_2[p,q,i] & \mid 1 \leq p = q \leq \mathrm{np} \wedge 1 \leq i \leq \mathrm{mb} \\ S_2[p,q-1,i] & \rightarrow S_2[p,q,i] & \mid 2 \leq q < p \leq \mathrm{np} \wedge 1 \leq i \leq \mathrm{mb} \\ S_0[p,i] & \rightarrow S_2[p,1,i] & \mid 2 \leq p \leq \mathrm{np} \wedge 1 \leq i \leq \mathrm{mb} \end{array} \tag{1}$$

Similarly, the source function for S_1 is:

$$\begin{array}{lll} S_2[p,q-1,i] & \rightarrow S_1[p,q,i] & \mid 2 \leq p = q \leq \mathrm{np} \wedge 1 \leq i \leq \mathrm{mb} \\ S_2[p-1,q,i] & \rightarrow S_1[p,q,i] & \mid 2 \leq p \leq \mathrm{np} \wedge q = p-1 \wedge 1 \leq i \leq \mathrm{mb} \\ S_1[p-1,q,i] & \rightarrow S_1[p,q,i] & \mid p \leq \mathrm{np} \wedge 1 \leq q \leq p-2 \wedge 1 \leq i \leq \mathrm{mb} \\ S_0[1,i] & \rightarrow S_1[1,1,i] & \mid 1 \leq i \leq \mathrm{mb} \end{array} \tag{2}$$

Dependence Relation Representation

```
      DO i = 0, M
        DO j = 0, N
S1:       A(2*i+j) = ...
        END DO
      END DO
S2: ... = A(k)
```

$$S_1[M, k{-}2M] \rightarrow S_2 \mid 2M \leq k \leq 2M{+}N \ \wedge \ M \geq 0$$

$$S_1[i, k{-}2i] \rightarrow S_2 \mid$$
$$2i{+}\alpha = k \ \wedge \ 0 \leq \alpha \leq 1 \ \wedge \ 0 \leq k \leq 2M+1 \ \wedge \ N \geq 1$$

$$S_1[i, 0] \rightarrow S_2 \mid$$
$$2i{+}\alpha = k \ \wedge \ 0 \leq \alpha \leq 1 \ \wedge \ 0 \leq k = 2\delta \leq 2M \ \wedge \ N = 0$$

Vectors And Statement Instances

- statement instance : The smallest unit of computation.
- Representation : W[**w**,**s**].
- W: statement of the program
- **w** : vector of loop variable values
- **s** : symbolic constant

Sequencing Predicate

W[**w,s**]$\ll$ R[**r,s**] if and only if
$w[1..n] \ll r[1..n] \vee w[1..n] = r[1..n] \wedge W \ll R$
where n is the no.of common loops surrounding W,R

Value based dependence definition And represention

$$\forall \mathbf{r}, \mathbf{s} : (V[\mathbf{v}, \mathbf{s}] \rightarrow R.A[\mathbf{r}, \mathbf{s}]) \in DepRel(\mathbf{w}, \mathbf{r}, \mathbf{s}) \Leftrightarrow$$
$$V[\mathbf{v}, \mathbf{s}] = \max_{\ll}(W[\mathbf{w}, \mathbf{s}] | \ \mathbf{w} \in [W, \mathbf{s}] \wedge \mathbf{r} \in [R, \mathbf{s}] \wedge$$
$$\mathrm{Arr}(W.B) = \mathrm{Arr}(R.A) \wedge W.\mathrm{B}(\mathbf{w}, \mathbf{s}) = R.\mathrm{A}(\mathbf{r}, \mathbf{s}) \wedge$$
$$W[\mathbf{w}, \mathbf{s}] \ll R[\mathbf{r}, \mathbf{s}])$$

$$DepRel = \left[\begin{array}{l} W_1[\mathbf{w}, \mathbf{s}] \rightarrow R.A[\mathbf{r}, \mathbf{s}] | \ DepRel_1(\mathbf{w}, \mathbf{r}, \mathbf{s}) \\ \ldots \\ W_m[\mathbf{w}, \mathbf{s}] \rightarrow R.A[\mathbf{r}, \mathbf{s}] | \ DepRel_m(\mathbf{w}, \mathbf{r}, \mathbf{s}) \end{array} \right.$$

where each $DepRel_i$ is a conjunction of constraints and

$$\bigcup_{i=1}^{m} \pi_{\mathbf{r},\mathbf{s}}(DepRel_i(\mathbf{w}, \mathbf{r}, \mathbf{s})) \subseteq [R, \mathbf{s}].$$

Algorithm for Value Based Dependence

Basic idea: To start searching for candidate writes in lexicographically close proximity of a read statement for which dependence is being computed.

1: **INPUT**: $R.A$: read reference surrounded by n loops with variables $\mathbf{r} = (r_1, \ldots, r_n)$.
   $\mathbf{s}$ is a vector of symbolic constants.
2: **OUTPUT**: Dependence relation for the read reference $R.A$.
   That is, $\{W[\mathbf{v},\mathbf{s}] \rightarrow R[\mathbf{r},\mathbf{s}]\} \in DepRel \Leftrightarrow W[\mathbf{v},\mathbf{s}] = \max_{\ll} (\, W[\mathbf{w},\mathbf{s}] \mid \mathbf{w} \in [W,\mathbf{s}] \wedge \mathbf{r} \in [R,\mathbf{s}] \wedge$
   $\mathrm{Arr}(W.B) = \mathrm{Arr}(R.A) \wedge W.\mathbf{B}(\mathbf{w},\mathbf{s}) = R.\mathbf{A}(\mathbf{r},\mathbf{s}) \wedge W[\mathbf{w},\mathbf{s}] \ll R[\mathbf{r},\mathbf{s}]\,)$
4: Relation $DepRel := \{\emptyset\}$; Relation $WrMax$
5: Dnf $NotCovered(\mathbf{r},\mathbf{s}) := \mathrm{IsExecuted}(R[\mathbf{r},\mathbf{s}])$
6: Integer $FixLoops := n$
7: Statement $W := R$
8: Boolean $SingleWrite :=$ True; Boolean $LessFlag :=$ False

Algorithm Contd...

While ($NotCovered$ is feasible) do
  W := statement preceding statement W
  Statement W is surrounded by m loops with variables $\mathbf{w} = (w_1, ..., w_m)$
  (* Here unfixed zone consists of loops with depths from $FixLoops + 1$ to n. *)
  If (W is assignment statement and it writes to $\mathsf{Arr}(R.A)$) then
    (* Find source function for instances of reference $R.A[\mathbf{r}, \mathbf{s}]$ which are $NotCovered(\mathbf{r}, \mathbf{s})$ *)
    Dnf $SameCell(\mathbf{w}, \mathbf{r}, \mathbf{s}) := NotCovered(\mathbf{r}, \mathbf{s}) \wedge R.\mathbf{A}(\mathbf{r}, \mathbf{s}) = W.\mathbf{B}(\mathbf{w}, \mathbf{s}) \wedge \mathsf{IsExecuted}(W[\mathbf{w}, \mathbf{s}])$
    Conjunct $Wsub(\mathbf{w}, \mathbf{r}) := \mathbf{w}[1..FixLoops] = \mathbf{r}[1..FixLoops] \wedge (LessFlag \Rightarrow w_{FixLoops+1} < r_{FixLoops+1})$
    Dnf $DepProb(\mathbf{w}, \mathbf{r}, \mathbf{s}) := SameCell(\mathbf{w}, \mathbf{r}, \mathbf{s}) \wedge Wsub(\mathbf{w}, \mathbf{r})$
    Relation $Cmax := \mathsf{RelMax1}_{\ll}(W[\mathbf{w}, \mathbf{s}] \rightarrow R.A[\mathbf{r}, \mathbf{s}] \mid DepProb(\mathbf{w}, \mathbf{r}, \mathbf{s}))$
    If ($SingleWrite$) then
      $DepRel := DepRel \cup Cmax$
      $NotCovered := NotCovered \wedge \neg\mathsf{range}(Cmax)$
    Else
      $WrMax := \mathsf{RelMax2}_{\ll}(WrMax, Cmax)$
    EndIf

Algorithm Contd...

```
ElseIf (statement W is EndDo or Do i=) then (* Enter loop body through its end *)
  If (SingleWrite) then
    If (statement W is Do i=) then
      FixLoops := FixLoops − 1;  LessFlag := True
      W := EndDo stmt for loop with header W
    Else (* statement W is EndDo *)
      LessFlag := False
    EndIf
    WrMax := {∅};  SingleWrite := False
    StopLoop := Do i= stmt of the loop whose EndDo stmt is W
  ElseIf (¬SingleWrite ∧ W = StopLoop) then
    DepRel := DepRel ∪ WrMax
    NotCovered := NotCovered ∧ ¬range(WrMax)
    SingleWrite := True
  EndIf
```

Algorithm Contd...

50: ElseIf (statement W is entry to the subroutine) then
51: $DepRel := DepRel \cup \{Entry \rightarrow R.A[\mathbf{r},\mathbf{s}] \mid NotCovered(\mathbf{r},\mathbf{s})\}$
52: Break out of While loop 10
55: ElseIf (statement W is `EndIf` or `Else` or `If (...) then`) then
56: (∗ Do nothing ∗)
60: EndIf
61:EndDo
62:Return ($DepRel$)

Example for affine fragment

```
      INTEGER rs, p, q, i
      DO rs = 1, nrs
        DO q = 1, np
          DO i = 1, mb
S0:         XRSIQ(i,q) = 0
          END DO
        END DO
        DO p = 1, np
          DO q = 1, p
            ...
            DO i = 1, mb
S1:           XRSIQ(i,q) = XRSIQ(i,q) + ...
S2:           XRSIQ(i,p) = XRSIQ(i,p) + ...
            END DO
          END DO
        END DO
        ...
      END DO
```

(a) Fragment of subroutine OLDA from TRFD

$C_0 : S_0 \to S_1$	$C_1 : S_1 \to S_1$	$C_2 : S_2 \to S_1$
$q_w = q_r$	$q_w = q_r$	$p_w = q_r$
$i_w = i_r$	$i_w = i_r$	$i_w = i_r$
$1 \le q_r \le p_r$	$1 \le q_r \le p_w, p_r$	$1 \le q_w \le q_r \le p_r$
$p_r \le \mathrm{np}$	$p_w, p_r \le \mathrm{np}$	$p_r \le \mathrm{np}$
$1 \le i_r \le \mathrm{mb}$	$1 \le i_r \le \mathrm{mb}$	$1 \le i_r \le \mathrm{mb}$

Example Contd...

The algorithm breaks a set of write statament instances into a sum of disjoint subsets $\omega_2(r), ..., \omega_{ns}(r)$ such that for any r such that R[r] is executed.

$$\omega_{ns}(r) \ll ... \ll \omega_2(r) \ll R(r)$$

Example Contd...

$$S_1[rs_r, p_r, q_r, i_r] \mid 1 \le rs_r \le nrs \wedge 1 \le q_r \le p_r \le np \wedge 1 \le i_r \le mb$$

$$\begin{aligned}
\omega_2 = {} & S_2[rs_r, p_r, q_r, i_w] \mid 1 \le i_w < i_r \\
& S_1[rs_r, p_r, q_r, i_w] \mid 1 \le i_w < i_r \\
\omega_3 = {} & S_2[rs_r, p_r, q_w, i_w] \mid 1 \le q_w < q_r \ \wedge \ 1 \le i_w \le \mathrm{mb} \\
& S_1[rs_r, p_r, q_w, i_w] \mid 1 \le q_w < q_r \ \wedge \ 1 \le i_w \le \mathrm{mb} \\
\omega_4 = {} & S_2[rs_r, p_w, q_w, i_w] \mid 1 \le q_w \le p_w < p_r \ \wedge \ 1 \le i_w \le \mathrm{mb} \\
& S_1[rs_r, p_w, q_w, i_w] \mid 1 \le q_w \le p_w < p_r \ \wedge \ 1 \le i_w \le \mathrm{mb} \\
\omega_5 = {} & S_0[rs_r, q_w, i_w] \mid 1 \le q_w \le \mathrm{np} \ \wedge \ 1 \le i_w \le \mathrm{mb} \\
\omega_6 = {} & S_2[rs_w, p_w, q_w, i_w] \mid 1 \le rs_w < rs_r \ \wedge \\
& \quad 1 \le q_w \le p_w \le \mathrm{np} \ \wedge \ 1 \le i_w \le \mathrm{mb} \\
& S_1[rs_w, p_w, q_w, i_w] \mid 1 \le rs_w < rs_r \ \wedge \\
& \quad 1 \le q_w \le p_w \le \mathrm{np} \ \wedge \ 1 \le i_w \le \mathrm{mb} \\
& S_0[rs_w, q_w, i_w] \mid 1 \le rs_w < rs_r \ \wedge \\
& \quad 1 \le q_w \le \mathrm{np} \ \wedge \ 1 \le i_w \le \mathrm{mb}
\end{aligned}$$

Example Contd...

$NotCovered(\mathtt{r}, \mathtt{s}) = (1 \le q_r \le p_r \le \mathtt{np} \wedge 1 \le i_r \le \mathtt{mb})$.

ω_2: $\omega_2 \wedge C_1$ and $\omega_2 \wedge C_2$ have no solutions. So ω_2 doesn't contribute to dependence.

ω_3: $C_1 \wedge \omega_3$ is not feasible, but $C_2 \wedge \omega_3 = (1 \le q_w < p_w = q_r = p_r \le \mathtt{np} \wedge 1 \le i_w = i_r \le \mathtt{mb})$. Computing $\mathrm{RelMax1}_{\ll}(S_2[p_w, q_w, i_w] \to S_1[p_r, q_r, i_r] \mid C_2 \wedge \omega_3)$ we get

$$S_2[p_r, q_r - 1, i_r] \to S_1[p_r, q_r, i_r] \mid \\ 2 \le p_r = q_r \le \mathtt{np} \wedge 1 \le i_r \le \mathtt{mb}$$

Example Contd...

Now we cover area $2 \le p_r = q_r \le \mathrm{np} \wedge 1 \le i_r \le \mathrm{mb}$ and therefore $NotCovered = (p_r = q_r = 1 \wedge 1 \le i_r \le \mathrm{mb}) \vee (1 \le q_r < p_r \le \mathrm{np} \wedge 1 \le i_r \le \mathrm{mb})$.

ω_4: $(C_2 \wedge \omega_4 \wedge NotCovered) = (1 \le q_w \le p_w = q_r < p_r \le \mathrm{np} \wedge 1 \le i_w = i_r \le \mathrm{mb})$. Maximum of this is $S_2[q_r, q_r, i_r] \mid 1 \le q_r < p_r \le \mathrm{np} \wedge 1 \le i_r \le \mathrm{mb}$.
$(C_1 \wedge \omega_4 \wedge NotCovered) = (1 \le q_w = q_r \le p_w < p_r \le \mathrm{np} \wedge 1 \le i_w = i_r \le \mathrm{mb})$ leading to maximum $S_1[p_r - 1, q_r, i_r] \mid 1 \le q_r < p_r \le \mathrm{np} \wedge 1 \le i_r \le \mathrm{mb}$.
Then we use $\mathsf{RelMax2}_{\ll}$ to compute $\max_{\ll}$ of two source functions (Appendix A.2). The result is

$$
\begin{array}{l}
S_2[p_r - 1, q_r, i_r] \rightarrow S_1[p_r, q_r, i_r] \mid \\
\quad 2 \le p_r \le \mathrm{np} \wedge q_r = p_r - 1 \wedge 1 \le i_r \le \mathrm{mb} \\
S_1[p_r - 1, q_r, i_r] \rightarrow S_1[p_r, q_r, i_r] \mid \\
\quad p_r \le \mathrm{np} \wedge 1 \le q_r \le p_r - 2 \wedge 1 \le i_r \le \mathrm{mb}
\end{array} \tag{7}
$$

$NotCovered = (p_r = q_r = 1 \wedge 1 \le i_r \le \mathrm{mb})$.

Example Contd...

ω_5: $(C_0 \wedge \omega_5 \wedge NotCovered) = (q_w = q_r = p_r = 1 \wedge 1 \leq i_w = i_r \leq \text{mb})$. This easily computes to dependence relation $S_0[1, i_r] \rightarrow S_1[1, 1, i_r] \mid 1 \leq i_r \leq \text{mb}$. Finally $NotCovered = \text{False}$.

After ω_5 step all the read instances of S_1 are covered and we don't have to compute dependences for ω_6 and any writes which textually precede S_0. The resulting source function for S_1 is given in (2).

Non Affine Fragments

```
    DO i = 1, n
      DO j = 1, n
        x = F(i,j)
S0:     IF (x) THEN
S1:       A(j) = ...
        ELSE
S2:       A(j) = ...
        ENDIF
S3:     ... = A(j)
      END DO
    END DO
```

(a) Non-affine IF condition

Non Affine Fragments

```
    DO i = 1, n
      DO j = 1, n
        x = F(i,j)
S1:     A(x) = ...
S2:     ...  = A(x)
      END DO
    END DO
```

(b) Non-affine subscript function

Fixed and Unfixed Zones

- **Unfixed zone:** Around statement S of depth d (denoted by Unfixed(S,d)) is a loop nest which consists of statements belonging to d innermost loops surrounding S.
- **Fixed zone:** Statements not belonging to UnFixed(S,d) affect.

Example

```
      DO i = 1, n
        DO j = 1, n
          x = F(i,j)
S0:       IF (x) THEN
S1:         A(j) = ...
          ELSE
S2:         A(j) = ...
          ENDIF
S3:       ... = A(j)
        END DO
      END DO
```

$$S_1[i,j] \to S_3[i,j] \mid 1 \leq i,j \leq n \wedge x$$
$$S_2[i,j] \to S_3[i,j] \mid 1 \leq i,j \leq n \wedge \neg x$$

$$S_1[i,j] \to S_3[i,j] \mid 1 \leq i,j \leq n \wedge x(i,j)$$
$$S_2[i,j] \to S_3[i,j] \mid 1 \leq i,j \leq n \wedge \neg x(i,j).$$

Example

- To expand the search space only if non covered read instances remain.

- Compute the Upper bound on iteration space
- Computing the Lower and Upper bound on dependence

Conclusion

- Computes exact value based(data-flow) dependences for affine program fragments
- And good approximations of value based dependences for non affine program fragments
- Independent of program size
- Depends on how many writes reach the read and on how complicated the dependence relation is.

References

- Vadim Maslov,**Lazy Array Data-Flow Dependence Analysis**, Proceedings of the 21st ACM SIGPLAN-SIGACT symposium on Principles of programming languages, p.311-325