Automatic Data Dependence Analysis by Deductive Verification

Asmae Heydari Tabar

joint work with:
Richard Bubel and Reiner Hähnle

Data Dependence

```
m = a;   //Write m
  ⋮
b = m;   //Read m
```

Data Dependence

```
m = a;   //Write m
  ⋮
b = m;   //Read m
```

Data Dependence

```
m = a;   //Write m
  ⋮
b = m;   //Read m
```

Data Dependence

Read after Write (RaW)

```
m = a;   //Write m
  ⋮
b = m;   //Read m
```

Data Dependence

Read after Write (RaW)

```
m = a;   //Write m
  ⋮
b = m;   //Read m
```

Write after Read (WaR)

```
b = m;   //Read m
  ⋮
m = a;   //Write m
```

Data Dependence

Read after Write (RaW)

```
m = a;   //Write m
  ⋮
b = m;   //Read m
```

Write after Read (WaR)

```
b = m;   //Read m
  ⋮
m = a;   //Write m
```

Write after Write (WaW)

```
m = a;   //Write m
  ⋮
m = b;   //Write m
```

Data Dependence

Read after Write (RaW)

```
m = a;   //Write m
  ⋮
b = m;   //Read m
```

Write after Read (WaR)

```
b = m;   //Read m
  ⋮
m = a;   //Write m
```

Write after Write (WaW)

```
m = a;   //Write m
  ⋮
m = b;   //Write m
```

Showing noRaW, noWaR, or noWaW $\Rightarrow$ Opportunity for parallelization

Data Dependence

Read after Write (RaW)

```
m = a;   //Write m
  ⋮
b = m;   //Read m
```

Write after Read (WaR)

```
b = m;   //Read m
  ⋮
m = a;   //Write m
```

Write after Write (WaW)

```
m = a;   //Write m
  ⋮
m = b;   //Write m
```

Showing noRaW, noWaR, or noWaW $\Rrightarrow$ Opportunity for parallelization

Analysis

Static $\rightsquigarrow$ Missing Parallelization Opportunities
Dynamic $\rightsquigarrow$ Wrong Behavior

Data Dependence

Read after Write (RaW)

```
m = a;   //Write m
  ⋮
b = m;   //Read m
```

Write after Read (WaR)

```
b = m;   //Read m
  ⋮
m = a;   //Write m
```

Write after Write (WaW)

```
m = a;   //Write m
  ⋮
m = b;   //Write m
```

Showing noRaW, noWaR, or noWaW $\Rightarrow$ Opportunity for parallelization

Analysis

Static $\rightsquigarrow$ Missing Parallelization Opportunities
Dynamic $\rightsquigarrow$ Wrong Behavior
Deductive Verification

Program Logic

Access Updates

$$\Longrightarrow \texttt{[a = m; m = b;]post}$$

Program Logic

Access Updates

$$\Longrightarrow [\ \texttt{a = m; m = b;}\ ]\,\texttt{post}$$

$$\Longrightarrow \{\ \texttt{a} := \texttt{m}\ \|\qquad\qquad\ \}[\ \texttt{m = b;}\ ](\texttt{post} \wedge\qquad\qquad)$$

Program Logic

Access Updates

$$\Longrightarrow [\ \texttt{a = m; m = b;}\ ]\,\texttt{post}$$

$$\Longrightarrow \{\ \texttt{a} := \texttt{m} \parallel \texttt{R(m); W(a)}\ \}[\ \texttt{m = b;}\ ](\texttt{post} \wedge \texttt{noRaW(m)})$$

Program Logic

Access Updates

$$\Longrightarrow [\ \texttt{a = m; m = b;}\ ]\,\texttt{post}$$

$$\Longrightarrow \{\ \texttt{a} := \texttt{m} \parallel \texttt{R(m); W(a)}\ \}[\ \texttt{m = b;}\ ](\texttt{post} \wedge \texttt{noRaW(m)})$$

Trace Semantics

Ghost Variable: $\langle$ (read, m) $\circ$ (write, a) $\rangle$

Program Logic

Access Updates

$$\Longrightarrow [\ \texttt{a = m; m = b;}\ ]\,\texttt{post}$$

$$\Longrightarrow \{\ \texttt{a} := \texttt{m} \parallel \texttt{R(m); W(a)}\ \}[\ \texttt{m = b;}\ ](\texttt{post} \wedge \texttt{noRaW(m)})$$

Trace Semantics

Ghost Variable: $\langle$ (read, m) $\circ$ (write, a) $\rangle$

noRaW(m)

Ghost Variable: $\langle \cdots \circ$ (write, x) $\circ \cdots \circ$ (read, y) $\circ \cdots \rangle$

$$x \cap y \cap m = \emptyset$$

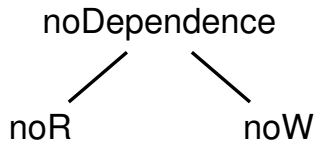
Loop Invariant Generation

Abstraction

Data Dependence	Memory Location
noDependence	$a[l_0 \,..\, h_0]$

Abstraction

Data Dependence

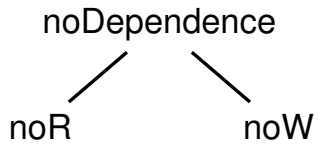


Memory Location

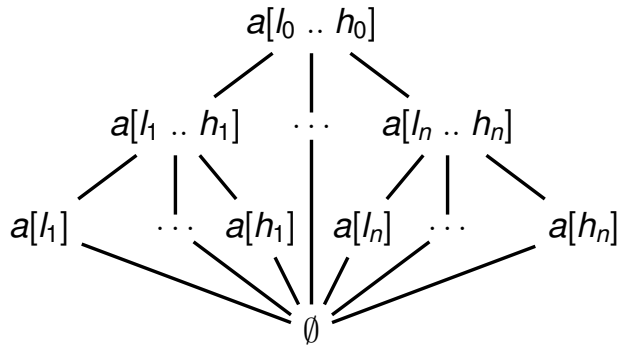
$a[l_0 .. h_0]$

Abstraction

Data Dependence

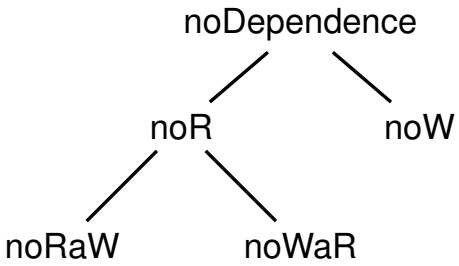


Memory Location

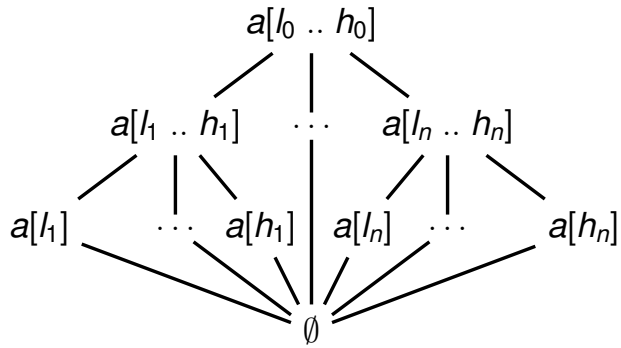


Abstraction

Data Dependence

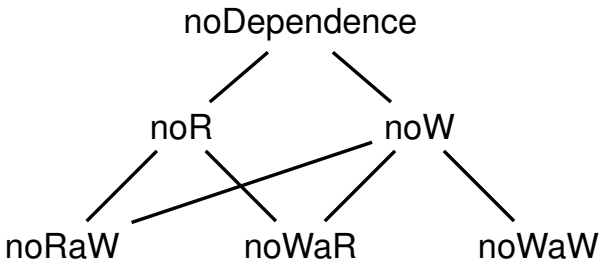


Memory Location

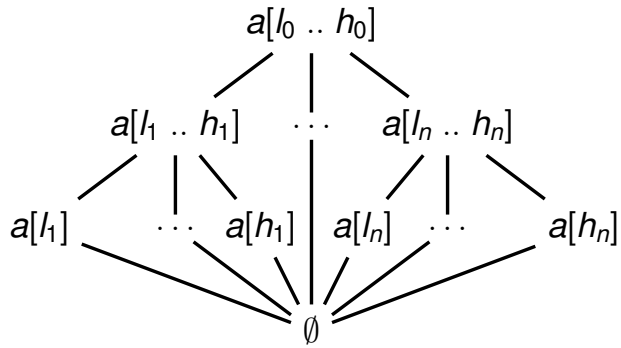


Abstraction

Data Dependence

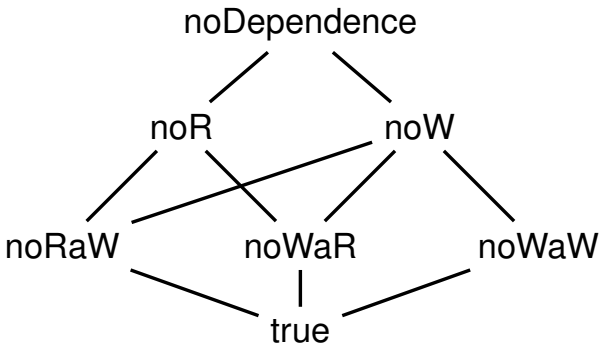


Memory Location

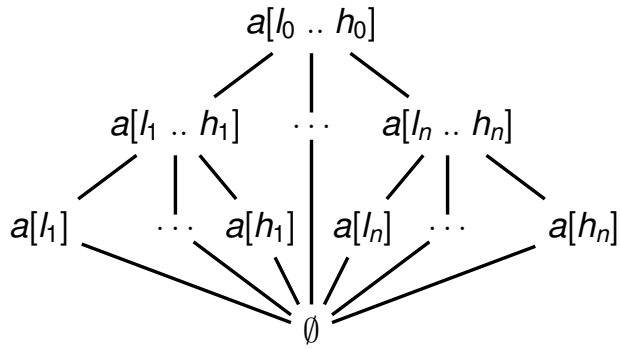


Abstraction

Data Dependence



Memory Location



Predicate Abstraction for Loop Invariant Generation

1. Initial set of predicates

Predicate Abstraction for Loop Invariant Generation

1. Initial set of predicates
2. Refine it while symbolically executing the loop

Predicate Abstraction for Loop Invariant Generation

1. Initial set of predicates
2. Refine it while symbolically executing the loop
3. Terminates with reaching a fixed point

Predicate Abstraction for Loop Invariant Generation

1. Initial set of predicates
2. Refine it while symbolically executing the loop
3. Terminates with reaching a fixed point
4. Loop invariant is a Boolean combination of the predicates

Predicate Abstraction for Loop Invariant Generation

1. Initial set of predicates
2. Refine it while symbolically executing the loop
3. Terminates with reaching a fixed point
4. Loop invariant is a Boolean combination of the predicates

Refining while Symbolically Executing

- Unwind

Refining while Symbolically Executing

- Unwind $\rightsquigarrow$ Ending up with updates on modality

Refining while Symbolically Executing

- Unwind $\rightsquigarrow$ Ending up with updates on modality
- Shift Update

Refining while Symbolically Executing

- Unwind $\rightsquigarrow$ Ending up with updates on modality
- Shift Update

$$\frac{\{\mathtt{R}^{-1}(\mathtt{m})\}\Gamma,\ \mathtt{Read}(\mathtt{m},0) \Longrightarrow \phi, \{\mathtt{R}^{-1}(\mathtt{m})\}\Delta}{\Gamma \Longrightarrow \{\mathtt{R}(\mathtt{m})\}\phi, \Delta}$$

$\rightsquigarrow$ **After** $\{\mathtt{R}(\mathtt{m})\}$

$\rightsquigarrow$ **Before** $\{\mathtt{R}(\mathtt{m})\}$

Refining while Symbolically Executing

- Unwind $\rightsquigarrow$ Ending up with updates on modality
- Shift Update

$$\frac{\{\mathtt{R}^{-1}(\mathtt{m})\}\Gamma,\ \mathtt{Read}(\mathtt{m},0) \Longrightarrow \phi, \{\mathtt{R}^{-1}(\mathtt{m})\}\Delta}{\Gamma \Longrightarrow \{\mathtt{R}(\mathtt{m})\}\phi, \Delta}$$

$\rightsquigarrow$ **After** $\{\mathtt{R}(\mathtt{m})\}$

$\rightsquigarrow$ **Before** $\{\mathtt{R}(\mathtt{m})\}$

Refining while Symbolically Executing

- Unwind $\rightsquigarrow$ Ending up with updates on modality
- Shift Update

$$\frac{\{\mathtt{R}^{-1}(\mathtt{m})\}\Gamma,\ \mathtt{Read(m,0)} \Longrightarrow \phi, \{\mathtt{R}^{-1}(\mathtt{m})\}\Delta}{\Gamma \Longrightarrow \{\mathtt{R}(\mathtt{m})\}\phi, \Delta}$$

$\rightsquigarrow$ **After** $\{\mathtt{R}(\mathtt{m})\}$

$\rightsquigarrow$ **Before** $\{\mathtt{R}(\mathtt{m})\}$

Refining while Symbolically Executing

- Unwind ⇝ Ending up with updates on modality
- Shift Update

$$\frac{\{\mathtt{R}^{-1}(\mathtt{m})\}\Gamma,\ \mathtt{Read}(\mathtt{m},0) \Longrightarrow \phi, \{\mathtt{R}^{-1}(\mathtt{m})\}\Delta}{\Gamma \Longrightarrow \{\mathtt{R}(\mathtt{m})\}\phi, \Delta}$$

⇝ **After** $\{\mathtt{R}(\mathtt{m})\}$

⇝ **Before** $\{\mathtt{R}(\mathtt{m})\}$

- Evaluate each predicate in the new sequent

Refining while Symbolically Executing

- Unwind $\rightsquigarrow$ Ending up with updates on modality
- Shift Update

$$\frac{\{\mathtt{R}^{-1}(\mathtt{m})\}\Gamma,\ \mathtt{Read}(\mathtt{m},0) \Longrightarrow \phi, \{\mathtt{R}^{-1}(\mathtt{m})\}\Delta}{\Gamma \Longrightarrow \{\mathtt{R}(\mathtt{m})\}\phi, \Delta}$$

$\rightsquigarrow$ **After** $\{\mathtt{R}(\mathtt{m})\}$

$\rightsquigarrow$ **Before** $\{\mathtt{R}(\mathtt{m})\}$

- Evaluate each predicate in the new sequent
- If not proven, weaken

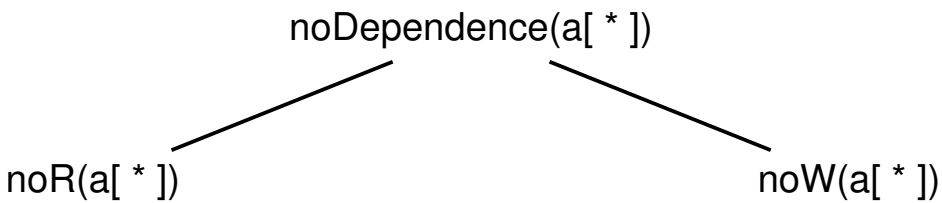
Example

```
int i = 0;
while(i < n - 1) {
   a[i] = a[i+1];
   i++;
}
```

noDependence(a[*])

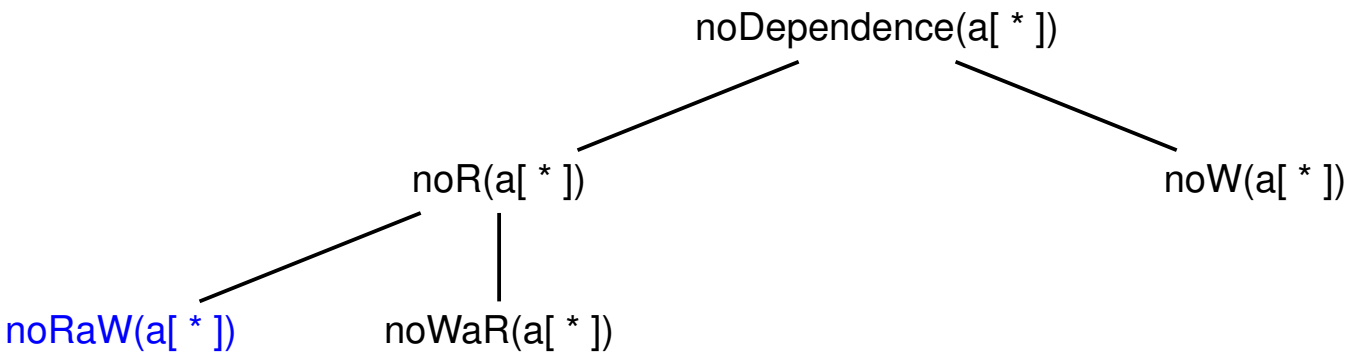
Example

```
int i = 0;
while(i < n - 1) {
   a[i] = a[i+1];
   i++;
}
```



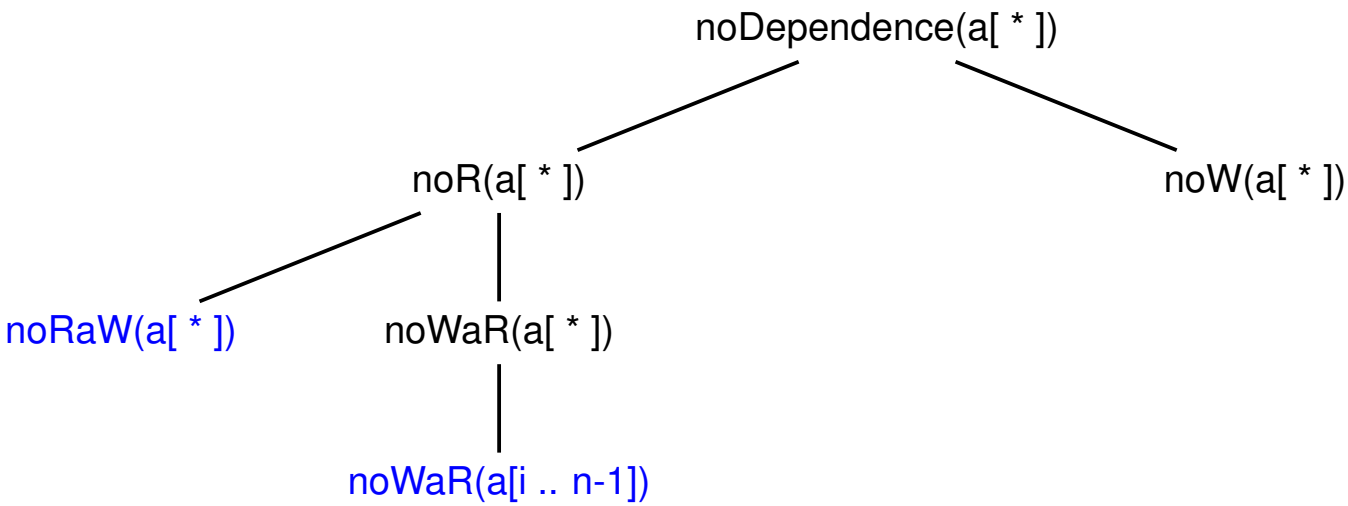
Example

```
int i = 0;
while(i < n - 1) {
   a[i] = a[i+1];
   i++;
}
```



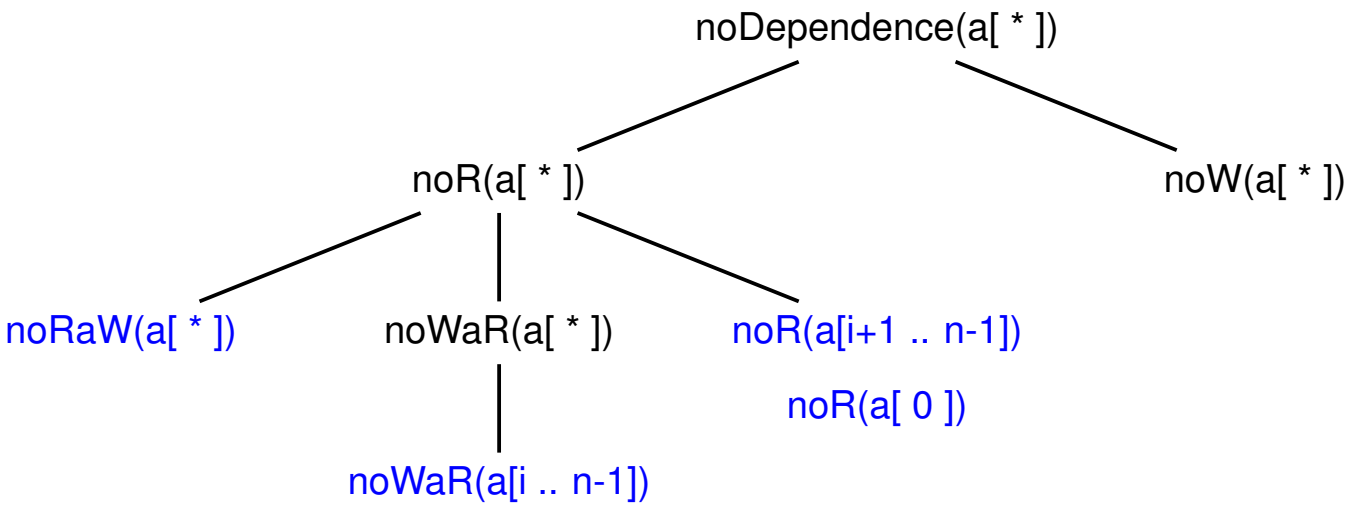
Example

```
int i = 0;
while(i < n - 1) {
   a[i] = a[i+1];
   i++;
}
```



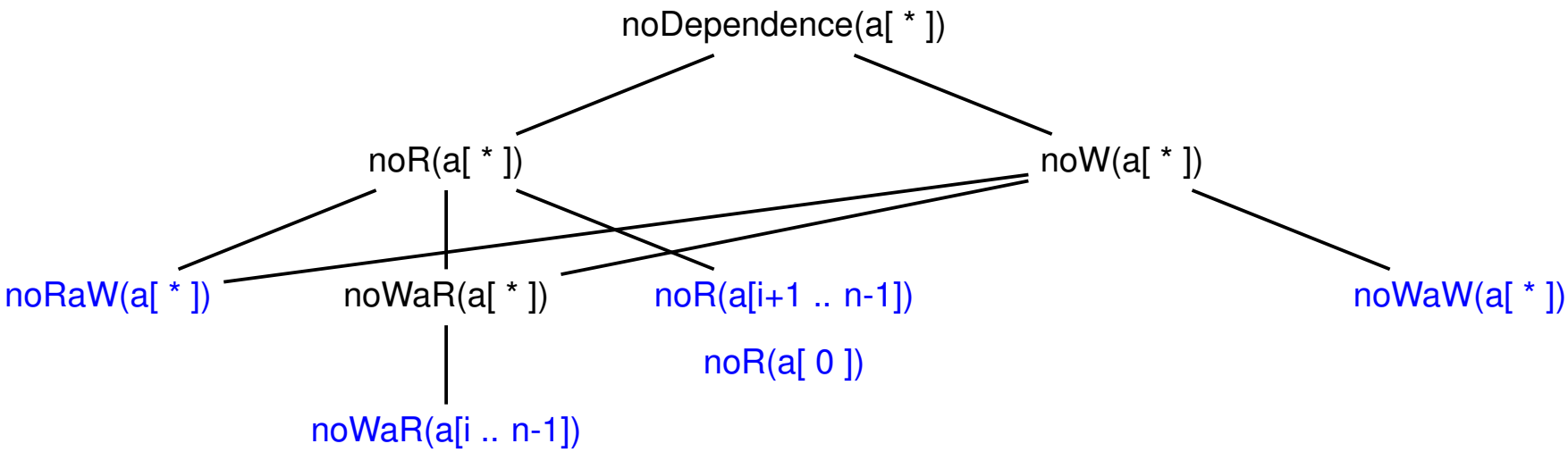
Example

```
int i = 0;
while(i < n - 1) {
   a[i] = a[i+1];
   i++;
}
```



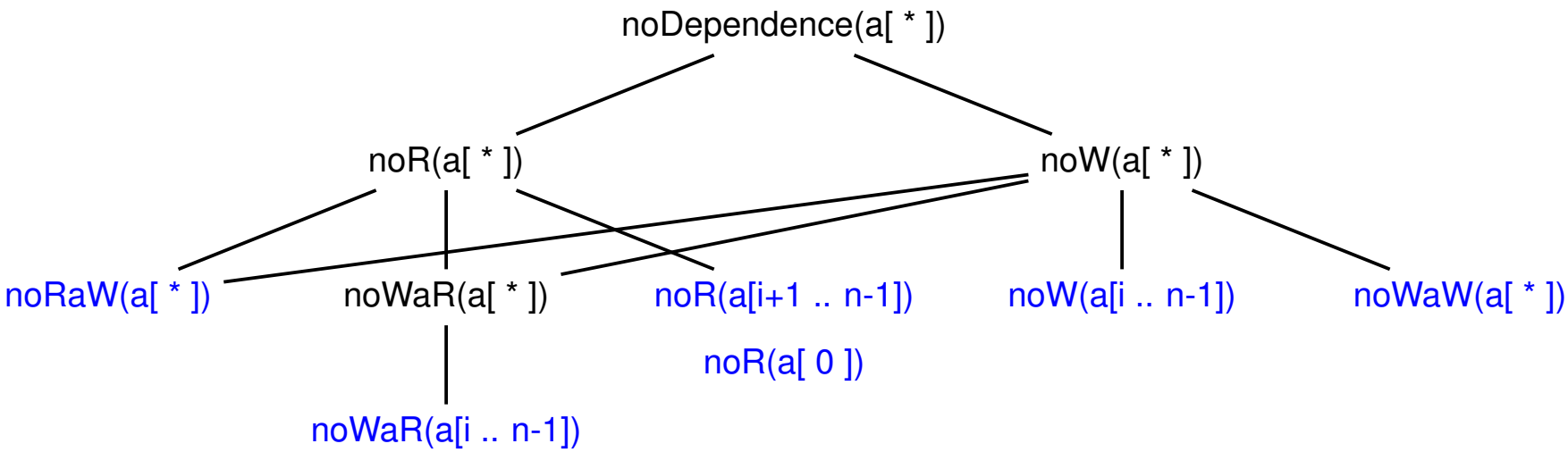
Example

```
int i = 0;
while(i < n - 1) {
   a[i] = a[i+1];
   i++;
}
```



Example

```
int i = 0;
while(i < n - 1) {
   a[i] = a[i+1];
   i++;
}
```



Invariant Generation for Nested Loops

What to do with the Inner Loop?

- Generate invariant for the inner loop
- Replace the inner loop with its invariant
- Apply the Use Case of the loop invariant rule

What to do with the Inner Loop?

- Generate invariant for the inner loop
- Replace the inner loop with its invariant
- Apply the Use Case of the loop invariant rule

$$\frac{\begin{array}{l}\text{Invariant Initially Valid}\\ \text{Body Preserves Invariant}\\ \text{Use Case}\end{array}}{\Gamma \Longrightarrow \{u\}[\texttt{while (}b\texttt{) \{}s\texttt{\} }r\texttt{;}]\phi, \Delta}$$

What to do with the Inner Loop?

- Generate invariant for the inner loop
- Replace the inner loop with its invariant
- Apply the Use Case of the loop invariant rule

$$\frac{\begin{array}{l}\text{Invariant Initially Valid}\\ \text{Body Preserves Invariant}\\ \text{Use Case}\end{array}}{\Gamma \Longrightarrow \{u\}[\texttt{while (}b\texttt{) \{}s\texttt{\} }r\texttt{;}]\phi, \Delta}$$

Example

```
int i = 0;
int j = 0;
while(i < n - 1) {
   while(j < m - 1) {
      a[i][j] = a[i+1][j+1];
      j++;
   }
   j = 0;
   i++;
}
```

Example

```
int i = 0;
int j = 0;
while(i < n - 1) {
   while(j < m - 1) {
      a[i][j] = a[i+1][j+1];
      j++;
   }
   j = 0;
   i++;
}
```

Loop Invariant:

$$
\begin{aligned}
&\text{noW(a[i .. n-1][j .. m-1])} \quad \wedge \\
&\text{noR(a[i+1 .. n-1] [j+1 .. m-1])} \quad \wedge \\
&\text{noR(a[0][0])} \quad \wedge \\
&\text{noWaW(a[*][*])} \quad \wedge \\
&\text{noRaW(a[*][*])} \quad \wedge \\
&0 \leq i \leq n - 1 \quad \wedge \\
&0 \leq j \leq m - 1
\end{aligned}
$$

There is more ...

- No assumption about nested loop

There is more ...

- No assumption about nested loop
- Inter-iteration dependences

There is more ...

- No assumption about nested loop
- Inter-iteration dependences
- Conditionals, aliasing, and function calls

There is more ...

- No assumption about nested loop
- Inter-iteration dependences
- Conditionals, aliasing, and function calls
- Precise result for PolyBench-c-4.2.1

Take Away

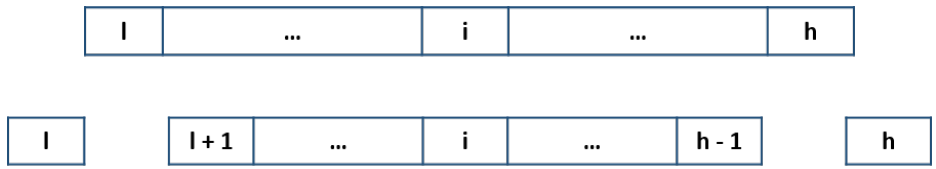
- Data dependence analysis with high precision
- Weakest Pre Cond. $\rightsquigarrow$ Strongest Post Cond. with Shift Update
- Use cases with an event after another event

Backup

Array Abstraction Heuristic

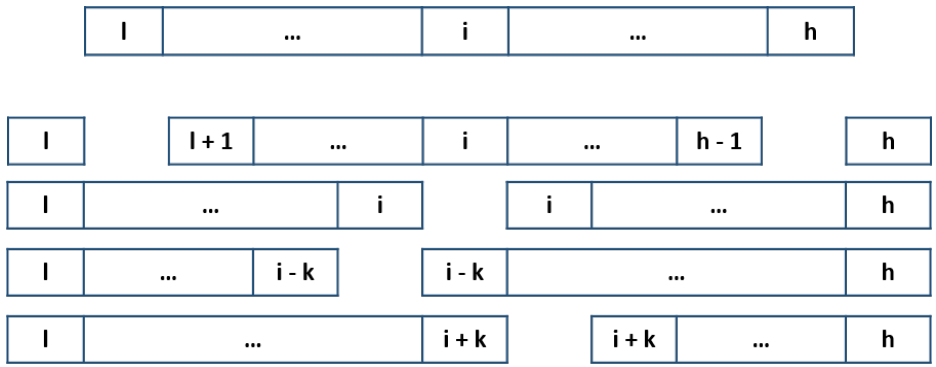
Backup

Array Abstraction Heuristic



Backup

Array Abstraction Heuristic



Backup

Shift Update

$$\frac{\{u'\}\Gamma,\ \mathit{link}(u,\ u') \Longrightarrow \phi,\ \{u'\}\Delta}{\Gamma \Longrightarrow \{u\}\ \phi, \Delta}$$

⇝ Evaluated after $\{u\}$

⇝ Evaluated before $\{u\}$

Backup

Shift Update

$$\frac{\{u'\}\Gamma,\ \mathit{link}(u,\ u') \Longrightarrow \phi,\ \{u'\}\Delta}{\Gamma \Longrightarrow \{u\}\ \phi, \Delta}$$

⇝ Evaluated after $\{u\}$

⇝ Evaluated before $\{u\}$

Example

$$\frac{\overbrace{\{\mathtt{i} := \mathtt{i}'\}}^{u'}\mathtt{i} \doteq 0,\ \overbrace{\mathtt{i} \doteq \mathtt{i}' + 1}^{\mathit{link}(u,\ u')} \Longrightarrow [\ldots]\, i > 0,\ \overbrace{\{\mathtt{i} := \mathtt{i}'\}}^{u'}\Delta}{\mathtt{i} \doteq 0 \Longrightarrow \{\underbrace{\mathtt{i} := \mathtt{i} + 1}_{u}\}[\ldots]\, i > 0, \Delta}$$

Backup

Shift Update

$$\frac{\{u'\}\Gamma,\ \mathit{link}(u,\ u') \Longrightarrow \phi,\ \{u'\}\Delta}{\Gamma \Longrightarrow \{u\}\ \phi, \Delta}$$

⇝ Evaluated after $\{u\}$

⇝ Evaluated before $\{u\}$

Example

$$\frac{\overbrace{\{\mathtt{i} := \mathtt{i}'\}}^{u'}\mathtt{i} \doteq 0,\ \overbrace{\mathtt{i} \doteq \mathtt{i}' + 1}^{\mathit{link}(u,\ u')} \Longrightarrow [\ldots]\, i > 0,\ \overbrace{\{\mathtt{i} := \mathtt{i}'\}}^{u'}\Delta}{\mathtt{i} \doteq 0 \Longrightarrow \{\underbrace{\mathtt{i} := \mathtt{i} + 1}_{u}\}[\ldots]\, i > 0, \Delta}$$

Backup

Shift Update

$$\frac{\{u'\}\Gamma,\ \mathit{link}(u,\ u') \Longrightarrow \phi,\ \{u'\}\Delta}{\Gamma \Longrightarrow \{u\}\ \phi, \Delta}$$

⇝ Evaluated after $\{u\}$

⇝ Evaluated before $\{u\}$

Example

$$\frac{\overbrace{\{\mathtt{i} := \mathtt{i}'\}}^{u'}\mathtt{i} \doteq 0,\ \overbrace{\mathtt{i} \doteq \mathtt{i}' + 1}^{\mathit{link}(u,\ u')} \Longrightarrow [\ldots]\, i > 0,\ \overbrace{\{\mathtt{i} := \mathtt{i}'\}\Delta}^{u'}}{\mathtt{i} \doteq 0 \Longrightarrow \{\underbrace{\mathtt{i} := \mathtt{i} + 1}_{u}\}[\ldots]\, i > 0, \Delta}$$

Backup

Shift Read Access Update

$$\frac{\{\mathtt{R}^{-1}(m)\}\Gamma, \mathtt{Read(m,\ 0)} \Longrightarrow \phi, \{\mathtt{R}^{-1}(m)\}\Delta}{\Gamma \Longrightarrow \{\mathtt{R}(m)\}\phi, \Delta}$$

- ⇝ Evaluated at ghost variable: ..., (read, m)
- ⇝ Evaluated at ghost variable: ...

Backup

Shift Read Access Update

$$\frac{\{\mathtt{R}^{-1}(m)\}\Gamma, \mathtt{Read(m,\ 0)} \Longrightarrow \phi, \{\mathtt{R}^{-1}(m)\}\Delta}{\Gamma \Longrightarrow \{\mathtt{R}(m)\}\phi, \Delta}$$

- ⇝ Evaluated at ghost variable: ..., (read, m)
- ⇝ Evaluated at ghost variable: ...

Backup

Shift Read Access Update

$$\frac{\{\mathtt{R}^{-1}(m)\}\Gamma, \mathtt{Read(m,\ 0)} \Longrightarrow \phi, \{\mathtt{R}^{-1}(m)\}\Delta}{\Gamma \Longrightarrow \{\mathtt{R}(m)\}\phi, \Delta}$$

0
↓

- ⇝ Evaluated at ghost variable: ..., (read, m)
- ⇝ Evaluated at ghost variable: ...

Backup

Shift Read Access Update

$$\frac{\{\mathtt{R}^{-1}(m)\}\Gamma, \mathtt{Read(m,\ 0)} \Longrightarrow \phi, \{\mathtt{R}^{-1}(m)\}\Delta}{\Gamma \Longrightarrow \{\mathtt{R}(m)\}\phi, \Delta}$$

- ⇝ Evaluated at ghost variable: ..., (read, m)
- ⇝ Evaluated at ghost variable: ...

Example

$$\frac{\{\mathtt{R}^{-1}(b)\}\Gamma,\ \mathtt{Read(b,\ 0)} \Longrightarrow \phi, \quad \{}{\Gamma \Longrightarrow \{\mathtt{R}(a),\ \mathtt{R}(b)\}\phi, \Delta}$$

Backup

Shift Read Access Update

$$\frac{\{\mathtt{R}^{-1}(m)\}\Gamma, \mathtt{Read(m,\ 0)} \Longrightarrow \phi, \{\mathtt{R}^{-1}(m)\}\Delta}{\Gamma \Longrightarrow \{\mathtt{R}(m)\}\phi, \Delta}$$

- ⇝ Evaluated at ghost variable: ..., (read, m)
- ⇝ Evaluated at ghost variable: ...

Example

$$\frac{\{\mathtt{R}^{-1}(a)\}\{\mathtt{R}^{-1}(b)\}\Gamma,\ \mathtt{Read(b,\ 1), Read(a,\ 0)} \Longrightarrow \phi, \{\mathtt{R}^{-1}(a)\}\{\mathtt{R}^{-1}(b)\}\Delta}{\Gamma \Longrightarrow \{\mathtt{R}(a),\ \mathtt{R}(b)\}\phi, \Delta}$$