Interprocedural Data Flow Analysis

Uday Khedker

Department of Computer Science and Engineering,
Indian Institute of Technology, Bombay

October 2015
Part 1

About These Slides
Copyright

These slides constitute the lecture notes for CS618 Program Analysis course at IIT Bombay and have been made available as teaching material accompanying the book:


Apart from the above book, some slides are based on the material from the following books


*These slides are being made available under GNU FDL v1.2 or later purely for academic or research use.*
Outline

• Issues in interprocedural analysis
• Functional approach
• Value context based approach
Part 2

Issues in Interprocedural Analysis
Interprocedural Analysis: Overview

- Extends the scope of data flow analysis across procedure boundaries
  Incorporates the effects of
  - procedure calls in the caller procedures, and
  - calling contexts in the callee procedures

- Approaches:
  - Generic: Call strings approach, functional approach
  - Problem specific: Alias analysis, Points-to analysis, Partial redundancy elimination, Constant propagation
Inherited and Synthesized Data Flow Information

<table>
<thead>
<tr>
<th>Data Flow Information</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>x</strong></td>
<td>Inherited by procedure $r$ from call site $c_i$ in procedure $s$</td>
</tr>
<tr>
<td><strong>y</strong></td>
<td>Inherited by procedure $r$ from call site $c_j$ in procedure $t$</td>
</tr>
<tr>
<td><strong>$x'$</strong></td>
<td>Synthesized by procedure $r$ in $s$ at call site procedure $c_i$</td>
</tr>
<tr>
<td><strong>$y'$</strong></td>
<td>Synthesized by procedure $r$ in $t$ at call site procedure $c_j$</td>
</tr>
</tbody>
</table>
Inherited and Synthesized Data Flow Information

• Example of uses of inherited data flow information

  Answering questions about formal parameters and global variables:
  ▶ Which variables are constant?
  ▶ Which variables aliased with each other?
  ▶ Which locations can a pointer variable point to?

• Examples of uses of synthesized data flow information

  Answering questions about side effects of a procedure call:
  ▶ Which variables are defined or used by a called procedure?
    (Could be local/global/formal variables)

• Most of the above questions may have a *May* or *Must* qualifier
Program Representation for Interprocedural Data Flow Analysis: Call Multi-Graph

Supergraphs of procedures
Program Representation for Interprocedural Data Flow Analysis: Call Multi-Graph

Supergraphs of procedures

Call multi-graph
Program Representation for Interprocedural Data Flow Analysis: Call Multi-Graph

Supergraphs of procedures

Call multi-graph
Program Representation for Interprocedural Data Flow Analysis: Call Multi-Graph

Supergraphs of procedures

Call multi-graph
Program Representation for Interprocedural Data Flow Analysis: Call Multi-Graph

Supergraphs of procedures

Call multi-graph
Program Representation for Interprocedural Data Flow Analysis: Call Multi-Graph

Supergraphs of procedures

Call multi-graph

Oct 2015
Program Representation for Interprocedural Data Flow Analysis: Supergraph
Program Representation for Interprocedural Data Flow Analysis: Supergraph

\[ S_{main} \rightarrow a + b \rightarrow C_1 \rightarrow \text{Call p} \]

\[ S_p \rightarrow \quad \rightarrow E_p \rightarrow \text{Call q} \rightarrow C_2 \]

\[ R_1 \rightarrow \quad \rightarrow E_{main} \]

\[ n_1 \rightarrow d = a + b \rightarrow n_2 \rightarrow a = 1 \rightarrow n_4 \rightarrow \text{Call q} \rightarrow n_3 \rightarrow \text{Call p} \rightarrow C_3 \rightarrow \text{Call p} \rightarrow C_4 \rightarrow \text{Call p} \]

\[ R_3 \rightarrow n_3 \rightarrow \text{Call p} \rightarrow C_3 \rightarrow \text{Call p} \rightarrow C_4 \rightarrow \text{Call p} \]

\[ R_4 \rightarrow \quad \rightarrow n_4 \rightarrow \text{Call p} \rightarrow C_4 \rightarrow \text{Call p} \]
Program Representation for Interprocedural Data Flow Analysis: Supergraph
Program Representation for Interprocedural Data Flow Analysis: Supergraph

Program:

```plaintext
main
  a + b
  Call p

C1

R1
  E_{main}

Sp
  Call q

C2

R2
  E_p

S_{main}

C3
  Call p

R3
  n_3

C4
  Call p

R4
  n_4

S_q
  n_1
  d = a + b

n_2
  a = 1
```

Variables:
- $a$
- $b$
- $c$
- $d$
- $n_1$
- $n_2$
- $n_3$
- $n_4$
Program Representation for Interprocedural Data Flow Analysis: Supergraph

\[ S_{\text{main}} \]
\[ a + b \]
\[ C_1 \]
\[ \text{Call p} \]

\[ S_p \]
\[ \text{Call q} \]

\[ C_2 \]
\[ R_1 \]
\[ E_{\text{main}} \]

\[ n_1 \]
\[ d = a + b \]
\[ C_3 \]
\[ \text{Call p} \]

\[ C_4 \]
\[ \text{Call p} \]

\[ R_3 \]
\[ n_3 \]
\[ n_4 \]

\[ a = 1 \]

\[ S_q \]
Program Representation for Interprocedural Data Flow Analysis: Supergraph

\[ S_{\text{main}}: a + b \]
\[ C_1: \text{Call p} \]
\[ R_1: \]
\[ E_{\text{main}}: \]
\[ S_p: \]
\[ C_2: \text{Call q} \]
\[ R_2: \]
\[ E_p: \]
\[ n_1: d = a + b \]
\[ C_3: \text{Call p} \]
\[ R_3: \]
\[ n_3: \]
\[ E_q: \]
\[ S_q: a = 1 \]
\[ n_2: \]
\[ C_4: \text{Call p} \]
\[ R_4: \]
\[ n_4: \]

Oct 2015
Validity of Interprocedural Control Flow Paths

Interprocedurally valid control flow path
Validity of Interprocedural Control Flow Paths

Interprocedurally valid control flow path
Validity of Interprocedural Control Flow Paths

Interprocedurally valid control flow path
Validity of Interprocedural Control Flow Paths

Interprocedurally invalid control flow path
Validity of Interprocedural Control Flow Paths

Interprocedurally invalid control flow path
Validity of Interprocedural Control Flow Paths

Interprocedurally valid control flow path
Soundness, Precision, and Efficiency of Data Flow Analysis

- Data flow analysis uses static representation of programs to compute summary information along paths
Soundness, Precision, and Efficiency of Data Flow Analysis

- Data flow analysis uses static representation of programs to compute summary information along paths

- *Ensuring Soundness.* All valid paths must be covered
Soundness, Precision, and Efficiency of Data Flow Analysis

- Data flow analysis uses static representation of programs to compute summary information along paths.

- *Ensuring Soundness.* All **valid** paths must be covered.

A path which represents legal control flow.
Soundness, Precision, and Efficiency of Data Flow Analysis

• Data flow analysis uses static representation of programs to compute summary information along paths

• Ensuring Soundness. All valid paths must be covered

• Ensuring Precision. Only valid paths should be covered

A path which represents legal control flow
Soundness, Precision, and Efficiency of Data Flow Analysis

- Data flow analysis uses static representation of programs to compute summary information along paths.

- **Ensuring Soundness.** All valid paths must be covered.

- **Ensuring Precision.** Only valid paths should be covered.

Subject to merging data flow values at shared program points without creating invalid paths.
Soundness, Precision, and Efficiency of Data Flow Analysis

- Data flow analysis uses static representation of programs to compute summary information along paths.

  - **Ensuring Soundness.** All valid paths must be covered.

  - **Ensuring Precision.** Only valid paths should be covered.

  - **Ensuring Efficiency.** Only relevant valid paths should be covered.

A path which represents legal control flow

Subject to merging data flow values at shared program points without creating invalid paths.
Soundness, Precision, and Efficiency of Data Flow Analysis

- Data flow analysis uses static representation of programs to compute summary information along paths.

- *Ensuring Soundness*. All valid paths must be covered.

- *Ensuring Precision*. Only valid paths should be covered.

- *Ensuring Efficiency*. Only relevant valid paths should be covered.

A path which represents legal control flow.

Subject to merging data flow values at shared program points without creating invalid paths.

A path which yields information that affects the summary information.
Flow and Context Sensitivity

- Flow sensitive analysis:
  Considers *intraprocedurally* valid paths
Flow and Context Sensitivity

- Flow sensitive analysis:
  Considers *intraprocedurally* valid paths

- Context sensitive analysis:
  Considers *interprocedurally* valid paths
Flow and Context Sensitivity

- Flow sensitive analysis:
  Considers *intraprocedurally* valid paths

- Context sensitive analysis:
  Considers *interprocedurally* valid paths

- For **maximum statically attainable precision**, analysis must be both flow and context sensitive
Flow and Context Sensitivity

- Flow sensitive analysis:
  Considers *intraprocedurally* valid paths

- Context sensitive analysis:
  Considers *interprocedurally* valid paths

- For **maximum statically attainable precision**, analysis must be both flow and context sensitive

MFP computation restricted to valid paths only
Context Sensitivity in Interprocedural Analysis

\[ x' = f_r(x) \]
\[ y' = f_r(y) \]
Context Sensitivity in Interprocedural Analysis

\[ S_s \xrightarrow{X} C_i \xrightarrow{c_i} R_i \xrightarrow{x'} E_s \]

\[ S_t \xrightarrow{y} C_j \xrightarrow{c_j} R_j \xrightarrow{y'} E_t \]

\[ S_r \xrightarrow{x} E_r \xrightarrow{f_r} S_t \]
Context Sensitivity in Interprocedural Analysis

\[ S_s \xrightarrow{x} C_i \xrightarrow{x'} R_i \xrightarrow{x'} E_s \]

\[ S_r \xrightarrow{f_r} S_t \]

\[ C_j \xrightarrow{c_j} R_j \xrightarrow{y'} E_t \]
Context Sensitivity in Interprocedural Analysis

The diagram illustrates the context sensitivity in interprocedural analysis. The states $S_s$, $C_i$, $R_i$, $E_s$ in the source module are connected to $S_r$, $E_r$ in the calling module, and to $S_t$, $C_j$, $R_j$, $E_t$ in the target module. The transitions are labeled with variables $x$, $y$, $x'$, $y'$, and the function $f_r$. The context sensitivity is shown by the different contexts $c_i$ and $c_j$ in the source and target modules, respectively.
Context Sensitivity in Interprocedural Analysis

\[ S_s \rightarrow C_i \rightarrow R_i \rightarrow E_s \]
\[ S_r \rightarrow f_r \rightarrow E_r \]
\[ S_t \rightarrow C_j \rightarrow R_j \rightarrow E_t \]

\[ c_i \rightarrow x \rightarrow x' \rightarrow f_r \rightarrow y \rightarrow y' \rightarrow c_j \]
Increasing Precision in Data Flow Analysis

Flow insensitive
  intraprocedural

Flow sensitive
  intraprocedural

Context insensitive
  flow insensitive

Context insensitive
  flow sensitive

Context sensitive
  flow insensitive

Context sensitive
  flow sensitive
Increasing Precision in Data Flow Analysis

Flow insensitive
Intraprocedural

Flow sensitive
Intraprocedural

Context insensitive
Flow insensitive

Context insensitive
Flow sensitive

Context sensitive
Flow sensitive

Context sensitive

actually, only caller sensitive
Part 3

Classical Functional Approach
**Functional Approach**

\[ x' = f_r(x) \]
Functional Approach

- Compute summary flow functions for each procedure
- Use summary flow functions as the flow function for a call block
Notation for Summary Flow Function

For simplicity forward flow is assumed
Notation for Summary Flow Function

For simplicity forward flow is assumed

- $u_i$: Program points
- $f_i$: Node flow functions
- $\Phi_r(u_i)$: Summary flow functions mapping data flow value from $S_r$ to $u_i$
Notation for Summary Flow Function

For simplicity forward flow is assumed

- $u_i$: Program points
- $f_i$: Node flow functions
- $\Phi_r(u_i)$: Summary flow functions mapping data flow value from $S_r$ to $u_i$
Notation for Summary Flow Function

For simplicity forward flow is assumed

- $u_i$: Program points
- $f_i$: Node flow functions
- $\Phi_r(u_i)$: Summary flow functions mapping data flow value from $S_r$ to $u_i$
Notation for Summary Flow Function

For simplicity forward flow is assumed

- \( u_i \): Program points
- \( f_i \): Node flow functions
- \( \Phi_r(u_i) \): Summary flow functions mapping data flow value from \( S_r \) to \( u_i \)

\[
\Phi_r(u_3) \equiv f_1
\]
Notation for Summary Flow Function

For simplicity forward flow is assumed

- $u_i$: Program points
- $f_i$: Node flow functions
- $\Phi_r(u_i)$: Summary flow functions mapping data flow value from $S_r$ to $u_i$

\[
\begin{align*}
\Phi_r(u_1) & \equiv \phi_{id} \\
\Phi_r(u_2) & \equiv f_1 \\
\Phi_r(u_3) & \equiv f_1 \\
\Phi_r(u_5) & \equiv f_2 \circ f_1 \\
\Phi_r(u_4) & \equiv f_1 \\
\Phi_r(u_8) & \equiv f_4
\end{align*}
\]
Notation for Summary Flow Function

For simplicity forward flow is assumed

- $u_i$: Program points
- $f_i$: Node flow functions
- $\Phi_r(u_i)$: Summary flow functions mapping data flow value from $S_r$ to $u_i$

\[
\begin{align*}
\Phi_r(u_1) &\equiv \phi_{id} \\
\Phi_r(u_2) &\equiv f_1 \\
\Phi_r(u_3) &\equiv f_1 \\
\Phi_r(u_5) &\equiv f_2 \circ f_1 \\
\Phi_r(u_6) &\equiv f_3 \circ f_1 \\
\Phi_r(u_7) &\equiv f_1 \\
\Phi_r(u_8) &\equiv f_3 \circ f_1
\end{align*}
\]
Notation for Summary Flow Function

For simplicity forward flow is assumed

- $u_i$: Program points
- $f_i$: Node flow functions
- $\Phi_r(u_i)$: Summary flow functions mapping data flow value from $S_r$ to $u_i$

\[
\Phi_r(u_1) \equiv \phi_{id} \\
\Phi_r(u_2) \equiv f_1 \\
\Phi_r(u_3) \equiv f_1 \\
\Phi_r(u_5) \equiv f_2 \circ f_1 \\
\Phi_r(u_7) \equiv f_2 \circ f_1 \sqcap f_3 \circ f_1 \\
\Phi_r(u_6) \equiv f_3 \circ f_1 \\
\Phi_r(u_4) \equiv f_1 \\
\Phi_r(u_8) \\
\]

Procedure $r$

Oct 2015

IIT Bombay
For simplicity forward flow is assumed

- $u_i$: Program points
- $f_i$: Node flow functions
- $\Phi_r(u_i)$: Summary flow functions mapping data flow value from $S_r$ to $u_i$

\[
\begin{align*}
\Phi_r(u_1) & \equiv \phi_{id} \\
\Phi_r(u_2) & \equiv f_1 \\
\Phi_r(u_3) & \equiv f_1 \\
\Phi_r(u_5) & \equiv f_2 \circ f_1 \\
\Phi_r(u_7) & \equiv f_2 \circ f_1 \sqcap f_3 \circ f_1 \\
\Phi_r(u_8) & \equiv f_4 \circ (f_2 \circ f_1 \sqcap f_3 \circ f_1)
\end{align*}
\]
Equations for Constructing Summary Flow Functions

For simplicity forward flow is assumed

\[ \Phi_r(I_n) = \begin{cases} 
\phi_{id} & \text{if } n \text{ is } S_r \\
\prod_{p \in \text{pred}(n)} (\Phi_r(O_p)) & \text{otherwise}
\end{cases} \]

\[ \Phi_r(O_n) = \begin{cases} 
\Phi_s(u) \circ \Phi_r(I_n) & \text{if } n \text{ calls procedure } s \\
f_n \circ \Phi_r(I_n) & \text{otherwise}
\end{cases} \]

The summary flow function of a given procedure \( r \)

- is influenced by summary flow functions of the callees of \( r \)
- is not influenced by summary flow functions of the callers of \( r \)

Fixed point computation may be required in the presence of loops or recursion
Constructing Summary Flow Functions Iteratively

\[ r \]

\[ f_1 \]

\[ f_2 \]
Constructing Summary Flow Functions Iteratively

$\Phi_r(u_1) = \phi_{id}$

$\Phi_r(u_2) = f_1$

$\Phi_r(u_3) = f_1$

$\Phi_r(u_4) = f_2 \circ f_1$
Constructing Summary Flow Functions Iteratively

\[ r \]

\[ \Phi_r(u_1) = \phi_{id} \]

\[ \Phi_r(u_2) = f_1 \]

\[ \Phi_r(u_3) = f_1 \cap f_2 \circ f_1 \]

\[ \Phi_r(u_4) = f_2 \circ (f_1 \cap f_2 \circ f_1) \]
Constructing Summary Flow Functions Iteratively

Iteration #3

\[ \Phi_r(u_1) = \phi_{id} \]
\[ \Phi_r(u_2) = f_1 \]
\[ \Phi_r(u_3) = f_1 \sqcap f_2 \circ (f_1 \sqcap f_2 \circ f_1) \]
\[ \Phi_r(u_4) = f_2 \circ (f_1 \sqcap f_2 \circ (f_1 \sqcap f_2 \circ f_1)) \]

Termination is possible only if all function compositions and confluences can be reduced to a finite set of functions.
Lattice of Flow Functions for Live Variables Analysis

Component functions (i.e. for a single variable)

<table>
<thead>
<tr>
<th>Lattice of data flow values</th>
<th>All possible flow functions</th>
<th>Lattice of flow functions</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\top = \emptyset$</td>
<td>$\emptyset$</td>
<td>$\phi_{id}$</td>
</tr>
<tr>
<td>$\bot = {a}$</td>
<td>$\emptyset$</td>
<td>$\phi_{id}$</td>
</tr>
<tr>
<td>$\emptyset$</td>
<td>${a}$</td>
<td>$\phi_{id}$</td>
</tr>
<tr>
<td>${a}$</td>
<td>$\emptyset$</td>
<td>$\phi_{id}$</td>
</tr>
<tr>
<td>${a}$</td>
<td>${a}$</td>
<td>$\phi_{id}$</td>
</tr>
</tbody>
</table>
Reducing Component Flow Functions for Live Variables
Analysis

Let \( \hat{\phi} \in \{\hat{\phi}_T, \hat{\phi}_{id}, \hat{\phi}_\bot\} \) and \( x \in \{1, 0\} \). Then,

- \( \hat{\phi}_T \cap \hat{\phi} = \hat{\phi} \) (because \( 0 + x = x \))
- \( \hat{\phi}_\bot \cap \hat{\phi} = \hat{\phi}_\bot \) (because \( 1 + x = 1 \))
- \( \hat{\phi}_T \circ \hat{\phi} = \hat{\phi}_T \) (because \( \hat{\phi}_T \) is a constant function)
- \( \hat{\phi}_\bot \circ \hat{\phi} = \hat{\phi}_\bot \) (because \( \hat{\phi}_\bot \) is a constant function)
- \( \hat{\phi}_{id} \circ \hat{\phi} = \hat{\phi} \) (because \( \hat{\phi}_{id} \) is the identity function)
Reducing Function Compositions in Bit Vector Frameworks

Kill$_n$ denoted by $K_n$ and Gen$_n$ denoted by $G_n$

$$f_3(x) = f_2(f_1(x))$$
Reducing Function Compositions in Bit Vector Frameworks

Kill$_n$ denoted by $K_n$ and Gen$_n$ denoted by $G_n$

\[
f_3(x) = f_2(f_1(x)) = f_2((x - K_1) \cup G_1)
\]
Reducing Function Compositions in Bit Vector Frameworks

Killₙ denoted by Kₙ and Genₙ denoted by Gₙ

\[
f₃(x) = f₂(f₁(x)) \\
= f₂((x - K₁) \cup G₁) \\
= (((x - K₁) \cup G₁) - K₂) \cup G₂
\]
Reducing Function Compositions in Bit Vector Frameworks

$\text{Kill}_n$ denoted by $K_n$ and $\text{Gen}_n$ denoted by $G_n$

\[
f_3(x) = f_2(f_1(x)) \\
= f_2((x - K_1) \cup G_1) \\
= (((x - K_1) \cup G_1) - K_2) \cup G_2 \\
= (x - (K_1 \cup K_2)) \cup (G_1 - K_2) \cup G_2
\]
Reducing Function Compositions in Bit Vector Frameworks

Kill$_n$ denoted by $K_n$ and Gen$_n$ denoted by $G_n$

\[ f_3(x) = f_2(f_1(x)) \]
\[ = f_2((x - K_1) \cup G_1) \]
\[ = (((x - K_1) \cup G_1) - K_2) \cup G_2 \]
\[ = (x - (K_1 \cup K_2)) \cup (G_1 - K_2) \cup G_2 \]

Hence,

\[ K_3 = K_1 \cup K_2 \]
\[ G_3 = (G_1 - K_2) \cup G_2 \]
Reducing Bit Vector Flow Function Confluences (1)

Kill
 and Gen
 denoted by \( K_n \) and \( G_n \)

- When \( \cap \) is \( \cup \),

\[
    f_3(x) = f_2(x) \cup f_1(x) \\
    = ((x - K_2) \cup G_2) \cup ((x - K_1) \cup G_1) \\
    = (x - (K_1 \cap K_2)) \cup (G_1 \cup G_2)
\]

Hence,

\[
    K_3 = K_1 \cap K_2 \\
    G_3 = G_1 \cup G_2
\]
Reducing Bit Vector Flow Function Confluences (1)

Killₙ denoted by \( Kₙ \) and Genₙ denoted by \( Gₙ \)

- When \( \cap \) is \( \cup \),

\[
f₃(x) = f₂(x) \cup f₁(x) \\
= ((x - K₂) \cup G₂) \cup ((x - K₁) \cup G₁) \\
= (x - (K₁ \cap K₂)) \cup (G₁ \cup G₂)
\]

Hence,

\[
K₃ = K₁ \cap K₂ \\
G₃ = G₁ \cup G₂
\]
Reducing Bit Vector Flow Function Confluences (2)

Kill$_n$ denoted by $K_n$ and Gen$_n$ denoted by $G_n$

- When $\cap$ is $\cap$,

\[
f_3(x) = f_2(x) \cap f_1(x) \\
= \left( (x - K_2) \cup G_2 \right) \cap \left( (x - K_1) \cup G_1 \right) \\
= \left( x - (K_1 \cup K_2) \right) \cup \left( G_1 \cap G_2 \right)
\]

Hence,

\[
K_3 = K_1 \cup K_2 \\
G_3 = G_1 \cap G_2
\]
Kill\(_n\) denoted by \(K_n\) and Gen\(_n\) denoted by \(G_n\)

- When \(\cap\) is \(\cap\),

\[
f_3(x) = f_2(x) \cap f_1(x) \\
= ((x - K_2) \cup G_2) \cap ((x - K_1) \cup G_1) \\
= (x - (K_1 \cup K_2)) \cup (G_1 \cap G_2)
\]

Hence,

\[
K_3 = K_1 \cup K_2 \\
G_3 = G_1 \cap G_2
\]
Lattice of Flow Functions for Live Variables Analysis

Flow functions for two variables

- Product of lattices for independent variables (because of separability)

<table>
<thead>
<tr>
<th>Lattice of data flow values</th>
<th>All possible flow functions</th>
<th>Lattice of flow functions</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\top = \emptyset$</td>
<td>$\emptyset$</td>
<td>$\emptyset$</td>
</tr>
<tr>
<td>${a}$</td>
<td>$\phi_{II}$</td>
<td>${a}$</td>
</tr>
<tr>
<td>${b}$</td>
<td>$\phi_{IT}$</td>
<td>${b}$</td>
</tr>
<tr>
<td>${a, b}$</td>
<td>$\phi_{TT}$</td>
<td>${a, b}$</td>
</tr>
<tr>
<td>${a}$</td>
<td>$\phi_{\bot I}$</td>
<td>${a, b}$</td>
</tr>
<tr>
<td>${a}$</td>
<td>$\phi_{\bot I}$</td>
<td>$\emptyset$</td>
</tr>
<tr>
<td>${a}$</td>
<td>$\phi_{\bot I}$</td>
<td>${a}$</td>
</tr>
<tr>
<td>${a}$</td>
<td>$\phi_{\bot I}$</td>
<td>${a}$</td>
</tr>
<tr>
<td>${a}$</td>
<td>$\phi_{\bot I}$</td>
<td>${a, b}$</td>
</tr>
<tr>
<td>${a}$</td>
<td>$\phi_{\bot I}$</td>
<td>${a}$</td>
</tr>
<tr>
<td>${a}$</td>
<td>$\phi_{\bot I}$</td>
<td>${a}$</td>
</tr>
<tr>
<td>${a}$</td>
<td>$\phi_{\bot I}$</td>
<td>${a, b}$</td>
</tr>
<tr>
<td>${a}$</td>
<td>$\phi_{\bot I}$</td>
<td>${a}$</td>
</tr>
<tr>
<td>${a}$</td>
<td>$\phi_{\bot I}$</td>
<td>${a}$</td>
</tr>
<tr>
<td>${a}$</td>
<td>$\phi_{\bot I}$</td>
<td>${a, b}$</td>
</tr>
<tr>
<td>${a}$</td>
<td>$\phi_{\bot I}$</td>
<td>${a}$</td>
</tr>
<tr>
<td>${a}$</td>
<td>$\phi_{\bot I}$</td>
<td>${a}$</td>
</tr>
<tr>
<td>${a}$</td>
<td>$\phi_{\bot I}$</td>
<td>${a, b}$</td>
</tr>
<tr>
<td>${a}$</td>
<td>$\phi_{\bot I}$</td>
<td>${a}$</td>
</tr>
<tr>
<td>${a}$</td>
<td>$\phi_{\bot I}$</td>
<td>${a}$</td>
</tr>
<tr>
<td>${a}$</td>
<td>$\phi_{\bot I}$</td>
<td>${a, b}$</td>
</tr>
<tr>
<td>${a}$</td>
<td>$\phi_{\bot I}$</td>
<td>${a}$</td>
</tr>
<tr>
<td>${a}$</td>
<td>$\phi_{\bot I}$</td>
<td>${a}$</td>
</tr>
<tr>
<td>${a}$</td>
<td>$\phi_{\bot I}$</td>
<td>${a, b}$</td>
</tr>
<tr>
<td>${a}$</td>
<td>$\phi_{\bot I}$</td>
<td>${a}$</td>
</tr>
<tr>
<td>${a}$</td>
<td>$\phi_{\bot I}$</td>
<td>${a}$</td>
</tr>
<tr>
<td>${a}$</td>
<td>$\phi_{\bot I}$</td>
<td>${a, b}$</td>
</tr>
<tr>
<td>${a}$</td>
<td>$\phi_{\bot I}$</td>
<td>${a}$</td>
</tr>
<tr>
<td>${a}$</td>
<td>$\phi_{\bot I}$</td>
<td>${a}$</td>
</tr>
<tr>
<td>${a}$</td>
<td>$\phi_{\bot I}$</td>
<td>${a, b}$</td>
</tr>
<tr>
<td>${a}$</td>
<td>$\phi_{\bot I}$</td>
<td>${a}$</td>
</tr>
<tr>
<td>${a}$</td>
<td>$\phi_{\bot I}$</td>
<td>${a}$</td>
</tr>
<tr>
<td>${a}$</td>
<td>$\phi_{\bot I}$</td>
<td>${a, b}$</td>
</tr>
<tr>
<td>${a}$</td>
<td>$\phi_{\bot I}$</td>
<td>${a}$</td>
</tr>
<tr>
<td>${a}$</td>
<td>$\phi_{\bot I}$</td>
<td>${a}$</td>
</tr>
<tr>
<td>${a}$</td>
<td>$\phi_{\bot I}$</td>
<td>${a, b}$</td>
</tr>
<tr>
<td>${a}$</td>
<td>$\phi_{\bot I}$</td>
<td>${a}$</td>
</tr>
<tr>
<td>${a}$</td>
<td>$\phi_{\bot I}$</td>
<td>${a}$</td>
</tr>
<tr>
<td>${a}$</td>
<td>$\phi_{\bot I}$</td>
<td>${a, b}$</td>
</tr>
<tr>
<td>${a}$</td>
<td>$\phi_{\bot I}$</td>
<td>${a}$</td>
</tr>
<tr>
<td>${a}$</td>
<td>$\phi_{\bot I}$</td>
<td>${a}$</td>
</tr>
<tr>
<td>${a}$</td>
<td>$\phi_{\bot I}$</td>
<td>${a, b}$</td>
</tr>
<tr>
<td>${a}$</td>
<td>$\phi_{\bot I}$</td>
<td>${a}$</td>
</tr>
<tr>
<td>${a}$</td>
<td>$\phi_{\bot I}$</td>
<td>${a}$</td>
</tr>
<tr>
<td>${a}$</td>
<td>$\phi_{\bot I}$</td>
<td>${a, b}$</td>
</tr>
<tr>
<td>${a}$</td>
<td>$\phi_{\bot I}$</td>
<td>${a}$</td>
</tr>
<tr>
<td>${a}$</td>
<td>$\phi_{\bot I}$</td>
<td>${a}$</td>
</tr>
<tr>
<td>${a}$</td>
<td>$\phi_{\bot I}$</td>
<td>${a, b}$</td>
</tr>
<tr>
<td>${a}$</td>
<td>$\phi_{\bot I}$</td>
<td>${a}$</td>
</tr>
<tr>
<td>${a}$</td>
<td>$\phi_{\bot I}$</td>
<td>${a}$</td>
</tr>
<tr>
<td>${a}$</td>
<td>$\phi_{\bot I}$</td>
<td>${a, b}$</td>
</tr>
<tr>
<td>${a}$</td>
<td>$\phi_{\bot I}$</td>
<td>${a}$</td>
</tr>
<tr>
<td>${a}$</td>
<td>$\phi_{\bot I}$</td>
<td>${a}$</td>
</tr>
<tr>
<td>${a}$</td>
<td>$\phi_{\bot I}$</td>
<td>${a, b}$</td>
</tr>
<tr>
<td>${a}$</td>
<td>$\phi_{\bot I}$</td>
<td>${a}$</td>
</tr>
<tr>
<td>${a}$</td>
<td>$\phi_{\bot I}$</td>
<td>${a}$</td>
</tr>
<tr>
<td>${a}$</td>
<td>$\phi_{\bot I}$</td>
<td>${a, b}$</td>
</tr>
<tr>
<td>${a}$</td>
<td>$\phi_{\bot I}$</td>
<td>${a}$</td>
</tr>
<tr>
<td>${a}$</td>
<td>$\phi_{\bot I}$</td>
<td>${a}$</td>
</tr>
<tr>
<td>${a}$</td>
<td>$\phi_{\bot I}$</td>
<td>${a, b}$</td>
</tr>
<tr>
<td>${a}$</td>
<td>$\phi_{\bot I}$</td>
<td>${a}$</td>
</tr>
<tr>
<td>${a}$</td>
<td>$\phi_{\bot I}$</td>
<td>${a}$</td>
</tr>
<tr>
<td>${a}$</td>
<td>$\phi_{\bot I}$</td>
<td>${a, b}$</td>
</tr>
<tr>
<td>${a}$</td>
<td>$\phi_{\bot I}$</td>
<td>${a}$</td>
</tr>
<tr>
<td>${a}$</td>
<td>$\phi_{\bot I}$</td>
<td>${a}$</td>
</tr>
<tr>
<td>${a}$</td>
<td>$\phi_{\bot I}$</td>
<td>${a, b}$</td>
</tr>
<tr>
<td>${a}$</td>
<td>$\phi_{\bot I}$</td>
<td>${a}$</td>
</tr>
<tr>
<td>${a}$</td>
<td>$\phi_{\bot I}$</td>
<td>${a}$</td>
</tr>
<tr>
<td>${a}$</td>
<td>$\phi_{\bot I}$</td>
<td>${a, b}$</td>
</tr>
<tr>
<td>${a}$</td>
<td>$\phi_{\bot I}$</td>
<td>${a}$</td>
</tr>
<tr>
<td>${a}$</td>
<td>$\phi_{\bot I}$</td>
<td>${a}$</td>
</tr>
<tr>
<td>${a}$</td>
<td>$\phi_{\bot I}$</td>
<td>${a, b}$</td>
</tr>
<tr>
<td>${a}$</td>
<td>$\phi_{\bot I}$</td>
<td>${a}$</td>
</tr>
<tr>
<td>${a}$</td>
<td>$\phi_{\bot I}$</td>
<td>${a}$</td>
</tr>
<tr>
<td>${a}$</td>
<td>$\phi_{\bot I}$</td>
<td>${a, b}$</td>
</tr>
<tr>
<td>${a}$</td>
<td>$\phi_{\bot I}$</td>
<td>${a}$</td>
</tr>
<tr>
<td>${a}$</td>
<td>$\phi_{\bot I}$</td>
<td>${a}$</td>
</tr>
<tr>
<td>${a}$</td>
<td>$\phi_{\bot I}$</td>
<td>${a, b}$</td>
</tr>
<tr>
<td>${a}$</td>
<td>$\phi_{\bot I}$</td>
<td>${a}$</td>
</tr>
<tr>
<td>${a}$</td>
<td>$\phi_{\bot I}$</td>
<td>${a}$</td>
</tr>
<tr>
<td>${a}$</td>
<td>$\phi_{\bot I}$</td>
<td>${a, b}$</td>
</tr>
<tr>
<td>${a}$</td>
<td>$\phi_{\bot I}$</td>
<td>${a}$</td>
</tr>
<tr>
<td>${a}$</td>
<td>$\phi_{\bot I}$</td>
<td>${a}$</td>
</tr>
<tr>
<td>${a}$</td>
<td>$\phi_{\bot I}$</td>
<td>${a, b}$</td>
</tr>
<tr>
<td>${a}$</td>
<td>$\phi_{\bot I}$</td>
<td>${a}$</td>
</tr>
<tr>
<td>${a}$</td>
<td>$\phi_{\bot I}$</td>
<td>${a}$</td>
</tr>
<tr>
<td>${a}$</td>
<td>$\phi_{\bot I}$</td>
<td>${a, b}$</td>
</tr>
<tr>
<td>${a}$</td>
<td>$\phi_{\bot I}$</td>
<td>${a}$</td>
</tr>
<tr>
<td>${a}$</td>
<td>$\phi_{\bot I}$</td>
<td>${a}$</td>
</tr>
<tr>
<td>${a}$</td>
<td>$\phi_{\bot I}$</td>
<td>${a, b}$</td>
</tr>
<tr>
<td>${a}$</td>
<td>$\phi_{\bot I}$</td>
<td>${a}$</td>
</tr>
<tr>
<td>${a}$</td>
<td>$\phi_{\bot I}$</td>
<td>${a}$</td>
</tr>
</tbody>
</table>
An Example of Interprocedural Liveness Analysis

\[
\begin{align*}
S_{\text{main}} & : \begin{align*}
a &= 5; \quad b &= 3 \\
c &= 7; \quad \text{read } d
\end{align*} \\
& \quad \xrightarrow{c_1} \text{Call } p \\
& \quad \xrightarrow{n_1} \begin{align*}
a &= a + 2 \\
e &= c + d
\end{align*} \\
& \quad \xrightarrow{n_2} d = a \times b \\
& \quad \xrightarrow{c_2} \text{Call } q \\
E_{\text{main}} & : \text{print } a + c + e
\end{align*}
\]

\[
\begin{align*}
S_p & : \begin{align*}
b &= 2 \\
\text{if } (b < d)
\end{align*} \\
& \quad \xrightarrow{T} c = a + b \\
& \quad \xrightarrow{F} \text{Call } q \\
E_p & : \text{print } c + d
\end{align*}
\]

\[
\begin{align*}
S_q & : \begin{align*}
a &= 1
\end{align*} \\
& \quad \xrightarrow{c_3} \text{Call } p \\
E_q & : a = a \times b
\end{align*}
\]
## Summary Flow Functions for Interprocedural Liveness Analysis

<table>
<thead>
<tr>
<th>Proc.</th>
<th>Flow Function</th>
<th>Defining Expression</th>
<th>Iteration #1</th>
<th>Changes in iteration #2</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>Gen</td>
<td>Kill</td>
</tr>
<tr>
<td>p</td>
<td>$\Phi_p(E_p)$</td>
<td>$f_{E_p}$</td>
<td>${c, d}$</td>
<td>$\emptyset$</td>
</tr>
<tr>
<td>p</td>
<td>$\Phi_p(n_3)$</td>
<td>$f_{n_3} \circ \Phi_p(E_p)$</td>
<td>${a, b, d}$</td>
<td>${c}$</td>
</tr>
<tr>
<td>p</td>
<td>$\Phi_p(c_4)$</td>
<td>$f_q \circ \Phi_p(E_p) = \phi_T$</td>
<td>$\emptyset$</td>
<td>${a, b, c, d, e}$</td>
</tr>
<tr>
<td>p</td>
<td>$\Phi_p(S_p)$</td>
<td>$f_{S_p} \circ (\Phi_p(n_3) \sqcap \Phi_p(c_4))$</td>
<td>${a, d}$</td>
<td>${b, c}$</td>
</tr>
<tr>
<td></td>
<td>$f_p$</td>
<td>$\Phi_p(S_p)$</td>
<td>${a, d}$</td>
<td>${b, c}$</td>
</tr>
<tr>
<td>q</td>
<td>$\Phi_q(E_q)$</td>
<td>$f_{E_q}$</td>
<td>${a, b}$</td>
<td>${a}$</td>
</tr>
<tr>
<td>q</td>
<td>$\Phi_q(c_3)$</td>
<td>$f_p \circ \Phi_q(E_q)$</td>
<td>${a, d}$</td>
<td>${a, b, c}$</td>
</tr>
<tr>
<td>q</td>
<td>$\Phi_q(S_q)$</td>
<td>$f_{S_q} \circ \Phi_q(c_3)$</td>
<td>${d}$</td>
<td>${a, b, c}$</td>
</tr>
<tr>
<td>q</td>
<td>$f_q$</td>
<td>$\Phi_q(S_q)$</td>
<td>${d}$</td>
<td>${a, b, c}$</td>
</tr>
</tbody>
</table>
Computed Summary Flow Functions

Summary Flow Function

<table>
<thead>
<tr>
<th>Function</th>
<th>Domain</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \Phi_p(E_p) )</td>
<td>( B{p } \cup {c, d} )</td>
</tr>
<tr>
<td>( \Phi_p(n_3) )</td>
<td>( (B{p } - {c}) \cup {a, b, d} )</td>
</tr>
<tr>
<td>( \Phi_p(c_4) )</td>
<td>( (B{p } - {a, b, c}) \cup {d} )</td>
</tr>
<tr>
<td>( \Phi_p(S_p) )</td>
<td>( (B{p } - {b, c}) \cup {a, d} )</td>
</tr>
<tr>
<td>( \Phi_q(E_q) )</td>
<td>( (B{q } - {a}) \cup {a, b} )</td>
</tr>
<tr>
<td>( \Phi_q(c_3) )</td>
<td>( (B{q } - {a, b, c}) \cup {a, d} )</td>
</tr>
<tr>
<td>( \Phi_q(S_q) )</td>
<td>( (B{q } - {a, b, c}) \cup {d} )</td>
</tr>
</tbody>
</table>
Result of Interprocedural Liveness Analysis

<table>
<thead>
<tr>
<th>Data flow variable</th>
<th>Summary flow function</th>
<th>Data flow value</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Procedure $main$, $BI = \emptyset$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$ln_{E_m}$</td>
<td>$\Phi_m(E_m)$</td>
<td>$BL_m \cup {a, c, e}$</td>
</tr>
<tr>
<td>$ln_{c_2}$</td>
<td>$\Phi_m(c_2)$</td>
<td>$(BL_m - {a, b, c}) \cup {d, e}$</td>
</tr>
<tr>
<td>$ln_{n_2}$</td>
<td>$\Phi_m(n_2)$</td>
<td>$(BL_m - {a, b, c, d}) \cup {a, b, e}$</td>
</tr>
<tr>
<td>$ln_{n_1}$</td>
<td>$\Phi_m(n_1)$</td>
<td>$(BL_m - {a, b, c, d, e}) \cup {a, b, c, d}$</td>
</tr>
<tr>
<td>$ln_{c_1}$</td>
<td>$\Phi_m(c_1)$</td>
<td>$(BL_m - {a, b, c, d, e}) \cup {a, d}$</td>
</tr>
<tr>
<td>$ln_{S_m}$</td>
<td>$\Phi_m(S_m)$</td>
<td>$BL_m - {a, b, c, d, e}$</td>
</tr>
</tbody>
</table>
## Result of Interprocedural Liveness Analysis

<table>
<thead>
<tr>
<th>Data flow variable</th>
<th>Summary flow function</th>
<th>Data flow value</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>Name</strong></td>
<td><strong>Definition</strong></td>
</tr>
<tr>
<td>**Procedure p, **</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$ln_{E_p}$</td>
<td>$\Phi_p(E_p)$</td>
<td>$Bl_p \cup {c, d}$</td>
</tr>
<tr>
<td>$ln_{n_3}$</td>
<td>$\Phi_p(n_3)$</td>
<td>$(Bl_p - {c}) \cup {a, b, d}$</td>
</tr>
<tr>
<td>$ln_{c_4}$</td>
<td>$\Phi_p(c_4)$</td>
<td>$(Bl_p - {a, b, c}) \cup {d}$</td>
</tr>
<tr>
<td>$ln_{S_p}$</td>
<td>$\Phi_p(S_p)$</td>
<td>$(Bl_p - {b, c}) \cup {a, d}$</td>
</tr>
<tr>
<td>**Procedure q, **</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$ln_{E_q}$</td>
<td>$\Phi_q(E_q)$</td>
<td>$(Bl_q - {a}) \cup {a, b}$</td>
</tr>
<tr>
<td>$ln_{c_3}$</td>
<td>$\Phi_q(c_3)$</td>
<td>$(Bl_q - {a, b, c}) \cup {a, d}$</td>
</tr>
<tr>
<td>$ln_{S_q}$</td>
<td>$\Phi_q(S_q)$</td>
<td>$(Bl_q - {a, b, c}) \cup {d}$</td>
</tr>
</tbody>
</table>
Context Sensitivity of Interprocedural Liveness Analysis

\( S_{\text{main}} \)

- \( a = 5; b = 3 \newline c = 7; \text{read } d \)

\( S_p \)

- \( b = 2 \)
- \( \text{if } (b < d) \)

\( E_p \)

- \( \text{print } c + d \)

\( S_q \)

- \( a = 1 \)

\( E_q \)

- \( a = a \ast b \)
Context Sensitivity of Interprocedural Liveness Analysis

\( S_{\text{main}} \)

\[
\begin{align*}
\{a, d\} & \quad \text{Call } p \\
\{a, b, c, d\} & \quad \text{Call } q \\
\{a, b, e\} & \\
\{d, e\} & \\
\{a, c, e\} & \\
\{a, c + d\} & \\
\end{align*}
\]

\( S_p \)

\[
\begin{align*}
\{a, b, d, e\} & \\
\{a, b, c, d, e\} & \\
\{d, e\} & \\
\end{align*}
\]

\( S_q \)

\[
\begin{align*}
\{a, d, e\} & \quad \text{Call } p \\
\{a, b, c, d, e\} & \quad \text{Call } q \\
\end{align*}
\]

\( E_{\text{main}} \)

\[
\text{print } a + c + e
\]

\( E_q \)

\[
\begin{align*}
\{a, b, c, d, e\} & \quad \text{Call } p \\
\end{align*}
\]

\( c = a + b \quad \text{if } (b < d) \)

\( d = a \ast b \quad c + d \)

\( a = 1 \)

\( a = 5; b = 3 \quad c = 7; \text{read } d \)

\( b = 3 \)

\( a \in \text{In}_{S_p} \text{ but } e \notin \text{In}_{c_1} \)
Explaining Context Sensitivity

- Flow function of procedure $p$ is identity with respect to variable $e$
Explaining Context Sensitivity

- Flow function of procedure $p$ is identity with respect to variable $e$
Explaining Context Sensitivity

- Flow function of procedure $p$ is identity with respect to variable $e$
- Is $e$ live in the body of procedure $p$?
  - During the analysis: Depends on the calling context
  - After the analysis: Yes (static approximation across all executions)
Explaining Context Sensitivity

• Flow function of procedure $p$ is identity with respect to variable $e$

• Is $e$ live in the body of procedure $p$?
  
  ▶ During the analysis: Depends on the calling context
  ▶ After the analysis: Yes (static approximation across all executions)

• Distinction between caller’s effect on callee and callee’s effect on caller
Perform interprocedural live variables analysis for the following program

```
main()
{
    p();
}
```
```
p()
{
    while (c < 10)
    {
        p();
        a = a*b;
    }
}
```
Tutorial Problem #2: Summary Flow Function for Constant Propagation

\[ b = c + d \]

\[ a = a + b \]

Call \( p \)

\[ a = a - b \]

Print \( c + d \)
Tutorial Problem #2: Summary Flow Function for Constant Propagation

<table>
<thead>
<tr>
<th></th>
<th>Iter. #1</th>
<th>Iter. #2</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\Phi_p(S_p)(\langle v_a, v_b \rangle)$</td>
<td>$\langle v_a, v_b \rangle$</td>
<td>$\langle v_a, v_b \rangle$</td>
</tr>
<tr>
<td>$\Phi_p(n_1)(\langle v_a, v_b \rangle)$</td>
<td>$\langle v_a + v_b, v_b \rangle$</td>
<td>$\langle v_a + v_b, v_b \rangle$</td>
</tr>
<tr>
<td>$\Phi_p(C_1)(\langle v_a, v_b \rangle)$</td>
<td>$\langle \top, \top \rangle$</td>
<td>$\langle v_a + v_b, v_b \rangle$</td>
</tr>
<tr>
<td>$\Phi_p(n_2)(\langle v_a, v_b \rangle)$</td>
<td>$\langle \top, \top \rangle$</td>
<td>$\langle v_a, v_b \rangle$</td>
</tr>
<tr>
<td>$\Phi_p(E_p)(\langle v_a, v_b \rangle)$</td>
<td>$\langle v_a, v_b \rangle$</td>
<td>$\langle v_a, v_b \rangle$</td>
</tr>
<tr>
<td>$f_p(\langle v_a, v_b \rangle)$</td>
<td>$\langle v_a, v_b \rangle$</td>
<td>$\langle v_a, v_b \rangle$</td>
</tr>
</tbody>
</table>
Tutorial Problem #2: Summary Flow Function for Constant Propagation

Will this work always?
Tutorial Problem #3

- Is a*b available on line 18? Line 6?
- Perform available expressions analysis by constructing the summary flow function for procedure p

```
1. main()
2. {
3.     c = a*b;
4.     p();
5.     a = a*b;
6. }
7. p()
8. {
9.     if (...)
10.     { a = a*b;
11.     p();
12.     }
13. else if (...)
14.     { c = a*b;
15.     p();
16.     c = a;
17. }
18. else
19.     ; /* ignore */
20. }
```
Limitations of Functional Approach to Interprocedural Data Flow Analysis

Problems with constructing summary flow functions
Limitations of Functional Approach to Interprocedural Data Flow Analysis

Problems with constructing summary flow functions

- Reducing expressions defining flow functions may not be possible in the presence of dependent parts
- May work for some instances of some problems but not for all
- Hence basic blocks in pointer analysis and constant propagation contain a single statement
Overall Flow Function and Component Function

- Overall flow function $f : L \mapsto L$ is $\langle \hat{h}_1, \hat{h}_2, \ldots, \hat{h}_m \rangle$

- Component function: $\hat{h}_i$ which computes the value of $\hat{x}_i$
Overall Flow Function and Component Function

- Overall flow function $f : L \leftrightarrow L$ is $\langle \hat{h}_1, \hat{h}_2, \ldots, \hat{h}_m \rangle$
- Component function: $\hat{h}_i$ which computes the value of $\hat{x}_i$

| Separable | General Non-Separable |
Overall Flow Function and Component Function

- Overall flow function $f : L \mapsto L$ is $\langle \hat{h}_1, \hat{h}_2, \ldots, \hat{h}_m \rangle$
- Component function: $\hat{h}_i$ which computes the value of $\hat{x}_i$

### Separable

\[
\langle \hat{x}_1, \hat{x}_2, \ldots, \hat{x}_m \rangle \\
\downarrow f \\
\langle \hat{y}_1, \hat{y}_2, \ldots, \hat{y}_m \rangle
\]

### General Non-Separable

\[
\langle \hat{x}_1, \hat{x}_2, \ldots, \hat{x}_m \rangle \\
\downarrow f \\
\langle \hat{y}_1, \hat{y}_2, \ldots, \hat{y}_m \rangle
\]
Overall Flow Function and Component Function

- Overall flow function $f : L \mapsto L$ is $\langle \hat{h}_1, \hat{h}_2, \ldots, \hat{h}_m \rangle$
- Component function: $\hat{h}_i$ which computes the value of $\hat{x}_i$

Separable

\[
\langle \hat{x}_1, \hat{x}_2, \ldots, \hat{x}_m \rangle
\]

\[
\hat{h}_2
\]

\[
\langle \hat{y}_1, \hat{y}_2, \ldots, \hat{y}_m \rangle
\]

General Non-Separable

\[
\langle \hat{x}_1, \hat{x}_2, \ldots, \hat{x}_m \rangle
\]

\[
f
\]

\[
\langle \hat{y}_1, \hat{y}_2, \ldots, \hat{y}_m \rangle
\]
Overall Flow Function and Component Function

- Overall flow function $f : L \mapsto L$ is $\langle \hat{h}_1, \hat{h}_2, \ldots, \hat{h}_m \rangle$
- Component function: $\hat{h}_i$ which computes the value of $\hat{x}_i$

Separable

$$\langle \hat{x}_1, \hat{x}_2, \ldots, \hat{x}_m \rangle$$

$$\hat{h}_2$$

$$\langle \hat{y}_1, \hat{y}_2, \ldots, \hat{y}_m \rangle$$

General Non-Separable

$$\langle \hat{x}_1, \hat{x}_2, \ldots, \hat{x}_m \rangle$$

$$f$$

$$\langle \hat{y}_1, \hat{y}_2, \ldots, \hat{y}_m \rangle$$

$\hat{h} : \hat{L} \mapsto \hat{L}$
**Overall Flow Function and Component Function**

- Overall flow function \( f : L \mapsto L \) is \( \langle \hat{h}_1, \hat{h}_2, \ldots, \hat{h}_m \rangle \)
- Component function: \( \hat{h}_i \) which computes the value of \( \hat{x}_i \)

### Separable

\[
\langle \hat{x}_1, \hat{x}_2, \ldots, \hat{x}_m \rangle \quad \hat{h}_2 \quad \langle \hat{y}_1, \hat{y}_2, \ldots, \hat{y}_m \rangle
\]

### General Non-Separable

\[
\langle \hat{x}_1, \hat{x}_2, \ldots, \hat{x}_m \rangle \quad \hat{h}_2 \quad \langle \hat{y}_1, \hat{y}_2, \ldots, \hat{y}_m \rangle
\]

\[ \hat{h} : \hat{L} \mapsto \hat{L} \]
Overall Flow Function and Component Function

- Overall flow function $f : L \mapsto L$ is $\langle \hat{h}_1, \hat{h}_2, \ldots, \hat{h}_m \rangle$
- Component function: $\hat{h}_i$ which computes the value of $\hat{x}_i$

### Separable

$\langle \hat{x}_1, \hat{x}_2, \ldots, \hat{x}_m \rangle$

$\hat{h}_2$

$\langle \hat{y}_1, \hat{y}_2, \ldots, \hat{y}_m \rangle$

$\hat{h} : \hat{L} \mapsto \hat{L}$

**Example:** All bit vector frameworks

### General Non-Separable

$\langle \hat{x}_1, \hat{x}_2, \ldots, \hat{x}_m \rangle$

$\hat{h}_2$

$\langle \hat{y}_1, \hat{y}_2, \ldots, \hat{y}_m \rangle$

$\hat{h} : L \mapsto \hat{L}$

**Example:** Points-To Analysis
Entity Functions in Points-to Analysis

<table>
<thead>
<tr>
<th>Statement with ( a \in L_locations )</th>
<th>Entity functions</th>
<th>Closed under composition?</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \ldots = \text{null} )</td>
<td>Constant ( \hat{L} \mapsto \hat{L} )</td>
<td>Yes</td>
</tr>
<tr>
<td>( \ldots = &amp; b )</td>
<td>Constant ( \hat{L} \mapsto \hat{L} )</td>
<td>Yes</td>
</tr>
<tr>
<td>( \ldots = b )</td>
<td>Identity ( \hat{L} \mapsto \hat{L} )</td>
<td>Yes</td>
</tr>
<tr>
<td>( \ldots = * b )</td>
<td>?</td>
<td>( L \mapsto \hat{L} ) No</td>
</tr>
</tbody>
</table>
## Entity Functions in Constant Propagation

<table>
<thead>
<tr>
<th>Statement</th>
<th>Entity functions</th>
<th>Closed under composition?</th>
</tr>
</thead>
<tbody>
<tr>
<td>$a = 5$</td>
<td>Constant</td>
<td>$\hat{L} \mapsto \hat{L}$</td>
</tr>
<tr>
<td>$a = b$</td>
<td>Constant</td>
<td>$\hat{L} \mapsto \hat{L}$</td>
</tr>
<tr>
<td>$a = b + 5$</td>
<td>Linear</td>
<td>$\hat{L} \mapsto \hat{L}$</td>
</tr>
<tr>
<td>$a = b + c$</td>
<td>?</td>
<td>$L \mapsto \hat{L}$</td>
</tr>
</tbody>
</table>
Enumeration Based Functional Approach

- Instead of constructing flow functions, remember the mapping $x \mapsto y$ as input output values
- Reuse output value of a flow function when the same input value is encountered again
Enumeration Based Functional Approach

- Instead of constructing flow functions, remember the mapping $x \mapsto y$ as input output values
- Reuse output value of a flow function when the same input value is encountered again

Requires the number of values to be finite
Part 4

IPDFA Using Value Contexts
Value Contexts: Key Ideas

Consider call chains $\sigma_1$ and $\sigma_2$ reaching $S_p$

- Data flow value invariant:
  If the data flow reaching $S_p$ along $\sigma_1$ and $\sigma_2$ are identical, then
Value Contexts: Key Ideas

Consider call chains $\sigma_1$ and $\sigma_2$ reaching $S_p$

- Data flow value invariant:
  If the data flow reaching $S_p$ along $\sigma_1$ and $\sigma_2$ are identical, then
  - the data flow values reaching $E_p$ for the two contexts will also be identical
Value Contexts: Key Ideas

Consider call chains $\sigma_1$ and $\sigma_2$ reaching $S_p$

- Data flow value invariant:
  If the data flow reaching $S_p$ along $\sigma_1$ and $\sigma_2$ are identical, then
    - the data flow values reaching $E_p$ for the two contexts will also be identical

- We can reduce the amount of effort by using
  - Data flow values at $S_p$ as value contexts
  - Maintaining distinct data flow values in $p$ for each value context
Interprocedural Data Flow Analysis Using Value Contexts

- A value context is defined by a particular input data flow value reaching a procedure.

- It is used to enumerate the summary flow functions in terms of \((\text{input} \mapsto \text{output})\) pairs.

- In order to compute these pairs, data flow analysis within a procedure is performed separately for each context (i.e. input data flow value).

- When a new call to a procedure is encountered, the pairs are consulted to decide if the procedure needs to be analysed again:
  - If it was already analysed once for the input value, output can be directly processed.
  - Otherwise, a new context is created and the procedure is analysed for this new context.
Understanding Value Contexts

\[ \sigma_0 \xRightarrow{X_0} \sigma_1 \xRightarrow{X_1} \sigma_2 \xRightarrow{X_2} \sigma_3 \xRightarrow{X_3} \]

\[ S_p \]

\[ C_j \]

Call q

\[ R_i \]

\[ E_p \]

\[ S_q \]

\[ E_q \]
Understanding Value Contexts

Separate contexts are created for each unique data flow value
Understanding Value Contexts

\[ \begin{align*}
S_0 & \quad \sigma_0 \quad x_0 \\
S_1 & \quad \sigma_1 \quad x_1 \\
S_2 & \quad \sigma_2 \quad x_2 \\
S_3 & \quad \sigma_3 \quad x_3
\end{align*} \]

Call \( q \)

\[
\begin{align*}
S_p & \quad C_j \\
R_i & \quad E_p
\end{align*}
\]

\[
\begin{align*}
S_q & \quad E_q
\end{align*}
\]
Understanding Value Contexts

Distinct data flow values are maintained for each context (i.e. each procedure is analysed separately for each context)
Understanding Value Contexts

New contexts are created for data flow values reaching $q$

Context transitions on call sites are recorded globally

$$s_0 \xrightarrow{c_i} s_0, \quad s_1 \xrightarrow{c_i} s_1, \quad s_2 \xrightarrow{c_i} s_2, \quad s_3 \xrightarrow{c_i} s_3$$
Understanding Value Contexts

New contexts are created for data flow values reaching $q$.
Context transitions on call sites are recorded globally.
Understanding Value Contexts

\[ \sigma_0 \xrightarrow{x_0} S_0 \]
\[ \sigma_1 \xrightarrow{x_1} S_1 \]
\[ \sigma_2 \xrightarrow{x_2} S_2 \]
\[ \sigma_3 \xrightarrow{x_3} S_3 \]

\[ S_p \]

\[ C_i \]

\[ R_i \]

\[ E_p \]

\[ S_q \]

\[ E_q \]
Understanding Value Contexts

\[ S_0 \xrightarrow{\sigma_0} X_0 \]
\[ S_1 \xrightarrow{\sigma_1} X_1 \]
\[ S_2 \xrightarrow{\sigma_2} X_2 \]
\[ S_3 \xrightarrow{\sigma_3} X_3 \]

\[ C_i \]

\[ R_i \]

\[ E_p \]

\[ S_4 \xrightarrow{S_0 C_i} X_0 \]
\[ S_5 \xrightarrow{S_1 C_i} X_1 \]
\[ S_6 \xrightarrow{S_3 C_i} X_3 \]
Understanding Value Contexts

Context transitions are consulted to transfer data flow values to calling contexts.

Oct 2015 IIT Bombay
Understanding Value Contexts

Context transitions are consulted to transfer data flow values to calling contexts.

Oct 2015 IIT Bombay
Understanding Value Contexts

$$\begin{align*}
S_0 &\rightarrow \frac{\sigma_0}{x_0} \\
S_1 &\rightarrow \frac{\sigma_1}{x_1} \\
S_2 &\rightarrow \frac{\sigma_2}{x_2} \\
S_3 &\rightarrow \frac{\sigma_3}{x_3} \\
S_4 &\rightarrow \frac{S_0 C_i}{x_0} \\
S_5 &\rightarrow \frac{S_1 C_i}{x_1} \\
S_6 &\rightarrow \frac{S_3 C_i}{x_3}
\end{align*}$$

$$\begin{align*}
S_0 &\rightarrow \frac{S_1}{x_1} \\
S_1 &\rightarrow \frac{S_2}{x_1} \\
S_2 &\rightarrow \frac{S_3}{x_3}
\end{align*}$$

$$\begin{align*}
S_0 &\rightarrow \frac{S_1}{y_1} \\
S_1 &\rightarrow \frac{S_2}{y_1} \\
S_2 &\rightarrow \frac{S_3}{y_3}
\end{align*}$$

$$\begin{align*}
S_0 &\rightarrow \frac{S_1}{y_0} \\
S_1 &\rightarrow \frac{S_2}{y_0} \\
S_2 &\rightarrow \frac{S_3}{y_3}
\end{align*}$$
Understanding Value Contexts

\[
\begin{align*}
S_0 & \rightarrow \sigma_0 \frac{x_0}{\sigma_i} \\
S_1 & \rightarrow \sigma_1 \frac{x_1}{\sigma_i} \\
S_2 & \rightarrow \sigma_2 \frac{x_2}{\sigma_i} \\
S_3 & \rightarrow \sigma_3 \frac{x_3}{\sigma_i} \\
S_4 & \rightarrow S_0 C_i \frac{x_0}{\sigma_i} \\
S_5 & \rightarrow S_1 C_i \frac{x_1}{\sigma_i} \\
S_6 & \rightarrow S_2 C_i \frac{x_3}{\sigma_i} \\
S_p & \rightarrow \omega \\
C_i & \rightarrow S_0 \frac{x_0}{y_0} \\
R_i & \rightarrow S_1 \frac{x_1}{y_1} \\
E_p & \rightarrow S_2 \frac{x_2}{y_1} \\
S_q & \rightarrow S_3 \frac{x_3}{y_1} \\
E_q & \rightarrow S_4 \frac{y_0}{y_1} \\
S_0 & \rightarrow S_5 \frac{y_1}{y_1} \\
S_1 & \rightarrow S_6 \frac{y_1}{y_1} \\
S_2 & \rightarrow S_3 \frac{y_1}{y_1} \\
S_3 & \rightarrow S_0 \frac{y_0}{y_1} \\
\end{align*}
\]
Understanding Value Contexts

\[ \sigma_0 \xrightarrow{X_0} S_0 \]
\[ \sigma_1 \xrightarrow{X_1} S_1 \]
\[ \sigma_2 \xrightarrow{X_2} S_2 \]
\[ \sigma_3 \xrightarrow{X_3} S_3 \]

\[ S_0 \xrightarrow{C_i} S_4 \]
\[ S_1 \xrightarrow{C_i} S_5 \]
\[ S_2 \xrightarrow{C_i} S_6 \]
\[ S_3 \xrightarrow{C_i} S_7 \]
Defining Value Contexts

- The set of value contexts is $VC = Procs \times L$

  A value context $X = \langle proc, entryValue \rangle \in VC$

  where $proc \in Procs$ and $entryValue \in L$
Defining Value Contexts

- The set of value contexts is $VC = Procs \times L$
  
  A value context $X = \langle proc, entryValue \rangle \in VC$
  
  where $proc \in Procs$ and $entryValue \in L$

- Supporting functions ($CS$ is the set of call sites)
  
  $\triangleright\ \exitValue : VC \mapsto L$

  $\triangleright\ \text{transitions} : (VC \times CS) \mapsto VC$
Defining Value Contexts

- The set of value contexts is \( VC = Procs \times L \)

A value context \( X = \langle proc, entryValue \rangle \in VC \)
where \( proc \in Procs \) and \( entryValue \in L \)

- Supporting functions (\( CS \) is the set of call sites)
  - \( exitValue : VC \mapsto L \)
    eg. \( exitValue(X) = v \)
  - \( transitions : (VC \times CS) \mapsto VC \)
    eg. \( X \xrightarrow{c_i} Y \)
Interprocedural Data Flow Analysis Using Value Contexts

- The method works with a collection of control flow graphs
  
  No need of supergraph
  
  - No need to distinguish between $C_i$ and $R_i$
  - No need of call $(C_i \rightarrow S_p)$ and return $(E_p \rightarrow E_i)$ edges

- Maintain a work list $WL$ of entries $\langle context, node \rangle$
  (in reverse post order of nodes within a procedure for forward flows)

- Notation:

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\langle p, v \rangle$</td>
<td>Context for procedure $p$ with data flow value $v$</td>
</tr>
<tr>
<td>$X</td>
<td>m$</td>
</tr>
<tr>
<td>$X.v$</td>
<td>Data flow value in context $X$ is $v$</td>
</tr>
<tr>
<td>$Out_m[X]$</td>
<td>Data flow value of context $X$ in $Out_m$</td>
</tr>
<tr>
<td>$X \overset{C_i}{\rightarrow} Y$</td>
<td>Transition from context $X$ to context $Y$ at call site $C_i$</td>
</tr>
</tbody>
</table>
Interprocedural Data Flow Analysis Using Value Contexts: An Overview

- Select $X|n$ from $WL$. Compute $ln_n$. 
Interprocedural Data Flow Analysis Using Value Contexts: An Overview

- Select $X|n$ from $WL$. Compute $In_n$.
  - If $n = C_i$ calling procedure $p$
  - If $n = E_p$
  - If $n$ is some other node
Interprocedural Data Flow Analysis Using Value Contexts: An Overview

- Select $X|n$ from $WL$. Compute $ln_n$.
  - If $n = C_i$ calling procedure $p$
    Propagate $ln_n$ to appropriate value context of the callee procedure $p$
  - If $n = E_p$
  - If $n$ is some other node
Interprocedural Data Flow Analysis Using Value Contexts: An Overview

- Select $X|n$ from $WL$. Compute $In_n$.
  - If $n = C_i$ calling procedure $p$
  - If $n = E_p$
    Propagate $In_n$ to appropriate value contexts of the callers of $p$
  - If $n$ is some other node
Interprocedural Data Flow Analysis Using Value Contexts: An Overview

- Select $X|n$ from $WL$. Compute $In_n$.
  - If $n = C_i$ calling procedure $p$
  - If $n = E_p$
  - If $n$ is some other node
    Compute $Out_n$
Interprocedural Data Flow Analysis Using Value Contexts: An Overview

• Select $X|n$ from $WL$. Compute $ln_n$.
  
  ▶ If $n = C_i$ calling procedure $p$

  ▶ If $n = E_p$

  ▶ If $n$ is some other node

Update $WL$
Interprocedural Data Flow Analysis Using Value Contexts: An Overview

- Select $X|n$ from $WL$. Compute $In_n$.
  - If $n = C_i$ calling procedure $p$
    Propagate $In_n$ to appropriate value context of the callee procedure $p$
  - If $n = E_p$
    Propagate $In_n$ to appropriate value contexts of the callers of $p$
  - If $n$ is some other node
    Compute $Out_n$

Update $WL$
Interprocedural Data Flow Analysis Using Value Contexts: An Overview

- Select $X|n$ from $WL$. Compute $In_n$.
  - If $n = C_i$ calling procedure $p$
    Propagate $In_n$ to appropriate value context of the callee procedure $p$
  - If $n = E_p$
    Propagate $In_n$ to appropriate value contexts of the callers of $p$
  - If $n$ is some other node
    Compute $Out_n$

Update $WL$

- Repeat until $WL$ is empty
Interprocedural Data Flow Analysis Using Value Contexts (2)

Select $X|n$ from $WL$. Compute $In_n$. Let $X.v$ be in $In_n$
Interprocedural Data Flow Analysis Using Value Contexts (2)

Select $X|n$ from $WL$. Compute $In_n$. Let $X.v$ be in $In_n$

- If $n = C_i$ calling procedure $p$
Select $X|n$ from $WL$. Compute $In_n$. Let $X.v$ be in $In_n$

- If $n = C_i$ calling procedure $p$
  - If some context $\langle p, v \rangle$ exists (say $Y$) /* $p$ is the callee */
  - If it does not exist
Select $X|n$ from $WL$. Compute $In_n$. Let $X.v$ be in $In_n$

- If $n = C_i$ calling procedure $p$
  - If some context $\langle p,v \rangle$ exists (say $Y$) /* $p$ is the callee */
    - record the transition $X \xrightarrow{C_i} Y$
    - $Out_{C_i}[X] = Out_{C_i}[X] \sqcap \text{exitValue}(Y)$
    - if there is a change, add $X|m$, $\forall m \in \text{succ}(C_i)$ to $WL$
  - If it does not exist
Select $X|n$ from $WL$. Compute $ln_n$. Let $X.v$ be in $ln_n$

- If $n = C_i$ calling procedure $p$
  - If some context $\langle p, v \rangle$ exists (say $Y$) /* $p$ is the callee */

  - If it does not exist
    - create a new context $Y = \langle p, v \rangle$ /* $p$ is the callee */
    - initialize $exitValue(Y) = \top$
    - record the transition $X \xrightarrow{C_i} Y$
    - initialize $Out_m[Y] = \top$ for all nodes $m$ of procedure $p$
    - add entries $Y|m$ for all nodes $m$ of procedure $p$ to $WL$
Interprocedural Data Flow Analysis Using Value Contexts (2)

Select $X|n$ from $WL$. Compute $In_n$. Let $X.v$ be in $In_n$

- If $n = C_i$ calling procedure $p$
  - If some context $\langle p,v \rangle$ exists (say $Y$) /* $p$ is the callee */
    - record the transition $X \xleftarrow{C_i} Y$
    - $Out_{C_i}[X] = Out_{C_i}[X] \cap exitValue(Y)$
    - if there is a change, add $X|m$, $\forall m \in succ(C_i)$ to $WL$
  - If it does not exist
    - create a new context $Y = \langle p,v \rangle$ /* $p$ is the callee */
    - initialize $exitValue(Y) = \top$
    - record the transition $X \xrightarrow{C_i} Y$
    - initialize $Out_m[Y] = \top$ for all nodes $m$ of procedure $p$
    - add entries $Y|m$ for all nodes $m$ of procedure $p$ to $WL$
Select $X|n$ from $WL$. Compute $In_n$. Let $X.v$ be in $In_n$.
Interprocedural Data Flow Analysis Using Value Contexts (3)

Select $X|n$ from $WL$. Compute $In_n$. Let $X.v$ be in $ln_n$

- If $n = E_p$

- For all other nodes
Select $X|n$ from $WL$. Compute $ln_n$. Let $X.v$ be in $ln_n$

- If $n = E_p$
  - Set $exitValue(X) = v$  
    /* $E_p$ is an empty block */

- For all other nodes
Select $X|n$ from $WL$. Compute $In_n$. Let $X.v$ be in $In_n$

- If $n = E_p$
  
  - Set $exitValue(X) = v$  
    
  - Find out all transitions $Z \xrightarrow{C_i} X$
    - Set $Out_{C_j}[Z] = v$
    - If there is a change, add $Z|m, \forall m \in succ(C_j)$ to $WL$

- For all other nodes
Interprocedural Data Flow Analysis Using Value Contexts (3)

Select \( X|n \) from \( WL \). Compute \( ln_n \). Let \( X.v \) be in \( ln_n \)

- If \( n = E_p \)

- For all other nodes
  - Set \( Out_n[X] = f_n(v) \)
Select $X|n$ from $WL$. Compute $In_n$. Let $X.v$ be in $In_n$

- If $n = E_p$
  - Set $exitValue(X) = v$  
    /* $E_p$ is an empty block */
  - Find out all transitions $Z \xrightarrow{C_i} X$
    - Set $Out_{C_j}[Z] = v$
    - If there is a change, add $Z|m$, $\forall m \in succ(C_j)$ to $WL$

- For all other nodes
  - Set $Out_n[X] = f_n(v)$
  - If there is a change, add $X|m$, $\forall m \in succ(n)$ to $WL$
Available Expressions Analysis Using Value Contexts

\[ S_{\text{main}} \]

- \text{read} a, b
- \text{t := a} \ast b

\[ C_1 \]

- \text{call } p

\[ n_1 \]

- \text{print } a \ast b

\[ E_{\text{main}} \]

\[ S_p \]

- \text{if } a \text{ == 0}

\[ n_2 \]

- \text{a = a} - 1

\[ C_2 \]

- \text{call } p

\[ n_3 \]

- \text{t = a} \ast b

\[ E_p \]
Available Expressions Analysis Using Value Contexts

$S_{main}$

read $a, b$

$t := a \ast b$

$C_1$

call $p$

$n_1$

print $a \ast b$

$E_{main}$

$n_2$

$a = a - 1$

$C_2$

call $p$

$n_3$

$t = a \ast b$

$E_p$

$S_p$

if $a == 0$

Is $a \ast b$ available?
Available Expressions Analysis Using Value Contexts

\[ S_{\text{main}} \]
\[
S_{\text{main}}: \text{read } a, b \rightarrow t := a \times b
\]
\[
C_1: \text{call } p \rightarrow \text{Is } a \times b \text{ available?}
\]
\[
n_1: \text{print } a \times b \rightarrow E_{\text{main}}
\]

\[ S_p \]
\[
S_p: \text{if } a \equiv 0
\]
\[
n_2: \text{a = } a - 1
\]
\[
C_2: \text{call } p \rightarrow E_p
\]
\[
n_3: t = a \times b
\]

\[
\text{int } a, b, t;
\]
\[
\text{void } p()
\]
\[
\{ \text{if } (a \equiv 0) \}
\]
\[
\{ \ a = a - 1; \}
\]
\[
\{ \ p(); \}
\]
\[
\{ \ t = a \times b; \}
\]
\}
Available Expressions Analysis Using Value Contexts

```
int a, b, t;
void p()
{
    if (a == 0)
    {
        a = a-1;
        p();
        t = a*b;
    }
}

int a, b, t;
void p()
{
    if (a == 0)
    {
        a = a-1;
        p();
        t = a*b;
    }
}
```
Available Expressions Analysis Using Value Contexts

\[
\begin{align*}
S_{main} & : \text{read } a, b \\
& \quad t := a \times b \\
& \quad \text{call } p \\
& \quad \text{print } a \times b \\
& \quad E_{main}
\end{align*}
\]

\[
\begin{align*}
S_p & : \text{if } a == 0 \\
& \quad a = a - 1 \\
& \quad \text{call } p \\
& \quad t = a \times b \\
& \quad E_p
\end{align*}
\]
Available Expressions Analysis Using Value Contexts

Create a new context $X_0$ with $BI$ which is 0 for available expressions analysis.
Available Expressions Analysis Using Value Contexts

\[ WL = [X_0|S_m, X_0|C_1, X_0|n_1, X_0|E_m] \]

Create a new context \( X_0 \) with BI which is 0 for available expressions analysis
Initialize \( exitValue(X_0) \) to \( \top = 1 \)
Initialize the work list with all nodes in procedure main for \( X_0 \)
Available Expressions Analysis Using Value Contexts

\[ WL = [X_0|S_m, X_0|C_1, X_0|n_1, X_0|E_m] \]

Context | exitValue
--- | ---
\( X_0 = \langle\text{main},0\rangle \) | 1

Compute the data flow values for \( S_m \) for context \( X_0 \)
Available Expressions Analysis Using Value Contexts

\[ WL = [X_0|C_1, X_0|n_1, X_0|E_m] \]

<table>
<thead>
<tr>
<th>Context</th>
<th>exitValue</th>
</tr>
</thead>
<tbody>
<tr>
<td>X_0 = \langle \text{main}, 0 \rangle</td>
<td>1</td>
</tr>
</tbody>
</table>

Compute the data flow values for \( S_m \) for context \( X_0 \)
Available Expressions Analysis Using Value Contexts

\[ WL = [X_0 | C_1, X_0 | n_1, X_0 | E_m] \]

<table>
<thead>
<tr>
<th>Context</th>
<th>exitValue</th>
</tr>
</thead>
<tbody>
<tr>
<td>X_0 = \langle \text{main}, 0 \rangle</td>
<td>1</td>
</tr>
</tbody>
</table>
Available Expressions Analysis Using Value Contexts

\[ WL = [X_1|S_p, X_1|n_2, X_1|C_2, X_1|n_3, X_1|E_p, X_0|n_1, X_0|E_m] \]

Create a new context \( X_1 \) with entry value 1
Record the transition to \( X_1 \)
Initialize \( exitValue(X_1) \) to \( \top = 1 \)
Add all nodes of procedure \( p \) to the work list for \( X_1 \)

<table>
<thead>
<tr>
<th>Context</th>
<th>exitValue</th>
</tr>
</thead>
<tbody>
<tr>
<td>( X_0 = \langle \text{main,0} \rangle )</td>
<td>1</td>
</tr>
<tr>
<td>( X_1 = \langle p,1 \rangle )</td>
<td>1</td>
</tr>
</tbody>
</table>
Available Expressions Analysis Using Value Contexts

\[ WL = [X_1|S_p, X_1|n_2, X_1|C_2, X_1|n_3, X_1|E_p, X_0|n_1, X_0|E_m] \]

<table>
<thead>
<tr>
<th>Context</th>
<th>exitValue</th>
</tr>
</thead>
<tbody>
<tr>
<td>X_0 = 〈main,0〉</td>
<td>1</td>
</tr>
<tr>
<td>X_1 = 〈p,1〉</td>
<td>1</td>
</tr>
</tbody>
</table>
Available Expressions Analysis Using Value Contexts

\[ WL = [X_0|n_2, X_1|C_2, X_1|n_3, X_1|E_p, X_0|n_1, X_0|E_m] \]

Context | exitValue
--- | ---
\( X_0 = \langle \text{main}, 0 \rangle \) | 1
\( X_1 = \langle \text{p}, 1 \rangle \) | 1
Available Expressions Analysis Using Value Contexts

\[ WL = [X_1|n_2, X_1|C_2, X_1|n_3, X_1|E_p, X_0|n_1, X_0|E_m] \]

<table>
<thead>
<tr>
<th>Context</th>
<th>exitValue</th>
</tr>
</thead>
<tbody>
<tr>
<td>( X_0 = \langle \text{main}, 0 \rangle )</td>
<td>1</td>
</tr>
<tr>
<td>( X_1 = \langle \text{p}, 1 \rangle )</td>
<td>1</td>
</tr>
</tbody>
</table>
Available Expressions Analysis Using Value Contexts

\[ WL = [X_1|n_2, X_1|C_2, X_1|n_3, X_1|E_p, X_0|n_1, X_0|E_m] \]

<table>
<thead>
<tr>
<th>Context</th>
<th>exitValue</th>
</tr>
</thead>
<tbody>
<tr>
<td>( X_0 = \langle \text{main,0} \rangle )</td>
<td>1</td>
</tr>
<tr>
<td>( X_1 = \langle p,1 \rangle )</td>
<td>1</td>
</tr>
</tbody>
</table>

Oct 2015
Available Expressions Analysis Using Value Contexts

\[ WL = [X_1|C_2, X_1|n_3, X_1|E_p, X_0|n_1, X_0|E_m] \]
Available Expressions Analysis Using Value Contexts

\[ WL = [X_1|C_2, X_1|n_3, X_1|E_p, X_0|n_1, X_0|E_m] \]

\[ X_0 \xrightarrow{C_1} X_1 \]

<table>
<thead>
<tr>
<th>Context</th>
<th>exitValue</th>
</tr>
</thead>
<tbody>
<tr>
<td>( X_0 = \langle \text{main}, 0 \rangle )</td>
<td>1</td>
</tr>
<tr>
<td>( X_1 = \langle \text{p}, 1 \rangle )</td>
<td>1</td>
</tr>
</tbody>
</table>

\[ S_{\text{main}} \]

- \( X_{0.0} \)
  - read \( a, b \)
  - \( t := a \ast b \)
- \( X_{0.1} \)
  - call \( p \)
  - \( n_1 \)
  - print \( a \ast b \)
- \( E_{\text{main}} \)

\[ S_p \]

- if \( a == 0 \)
  - \( X_{1.1} \)
  - \( n_2 \)
  - \( a = a - 1 \)
- \( X_{1.0} \)
  - call \( p \)
  - \( n_3 \)
  - \( t = a \ast b \)
- \( E_p \)
Available Expressions Analysis Using Value Contexts

\[ WL = [X_1|C_2, X_1|n_3, X_1|E_p, X_0|n_1, X_0|E_m] \]

Since there is no context for \( p \) with value 0, create context \( X_2 \)
Record the transition to \( X_2 \)
Initialize \( exitValue(X_2) \) to \( \top = 1 \)
Add all nodes of procedure \( p \) to the work list for \( X_2 \)
Available Expressions Analysis Using Value Contexts

\[ WL = [X_2|S_p, X_2|n_2, X_2|C_2, X_2|n_3, X_2|E_p, X_1|n_3, X_1|E_p, X_0|n_1, X_0|E_m] \]

<table>
<thead>
<tr>
<th>Context</th>
<th>exitValue</th>
</tr>
</thead>
<tbody>
<tr>
<td>X_0 = \langle \text{main,0} \rangle</td>
<td>1</td>
</tr>
<tr>
<td>X_1 = \langle p,1 \rangle</td>
<td>1</td>
</tr>
<tr>
<td>X_2 = \langle p,0 \rangle</td>
<td>1</td>
</tr>
</tbody>
</table>
Available Expressions Analysis Using Value Contexts

\[ WL = [X_2|S_p, X_2|n_2, X_2|C_2, X_2|n_3, X_2|E_p, X_1|n_3, X_1|E_p, X_0|n_1, X_0|E_m] \]

<table>
<thead>
<tr>
<th>Context</th>
<th>exitValue</th>
</tr>
</thead>
<tbody>
<tr>
<td>( X_0 = \langle \text{main,0} \rangle )</td>
<td>1</td>
</tr>
<tr>
<td>( X_1 = \langle \text{p,1} \rangle )</td>
<td>1</td>
</tr>
<tr>
<td>( X_2 = \langle \text{p,0} \rangle )</td>
<td>1</td>
</tr>
</tbody>
</table>
Available Expressions Analysis Using Value Contexts

\[ WL = [X_2|n_2, X_2|C_2, X_2|n_3, X_2|E_p, X_1|n_3, X_1|E_p, X_0|n_1, X_0|E_m] \]

\[
\begin{array}{c}
\text{Context} & \text{exitValue} \\
X_0 = \langle \text{main}, 0 \rangle & 1 \\
X_1 = \langle p, 1 \rangle & 1 \\
X_2 = \langle p, 0 \rangle & 1 \\
\end{array}
\]
Available Expressions Analysis Using Value Contexts

\[ WL = [X_2|n_2, X_2|C_2, X_2|n_3, X_2|E_p, X_1|n_3, X_1|E_p, X_0|n_1, X_0|E_m] \]

\[
\begin{array}{c}
X_0 \xrightarrow{C_1} X_1 \xrightarrow{C_2} X_2 \\
\end{array}
\]

<table>
<thead>
<tr>
<th>Context</th>
<th>exitValue</th>
</tr>
</thead>
<tbody>
<tr>
<td>(X_0 = \langle \text{main},0 \rangle)</td>
<td>1</td>
</tr>
<tr>
<td>(X_1 = \langle p,1 \rangle)</td>
<td>1</td>
</tr>
<tr>
<td>(X_2 = \langle p,0 \rangle)</td>
<td>1</td>
</tr>
</tbody>
</table>
Available Expressions Analysis Using Value Contexts

\[ WL = [X_2|C_2, X_2|n_3, X_2|E_p, X_1|n_3, X_1|E_p, X_0|n_1, X_0|E_m] \]

\[
\begin{align*}
X_0 & \xrightarrow{C_1} X_1 \xrightarrow{C_2} X_2 \\
X_0.0 & \quad \quad \quad X_1.1 \quad \quad \quad X_2.0 \\
S_{main} & \quad read \ a, b \\
& \quad t := a \ast b \\
& \quad X_0.1 \\
& \quad \quad \quad \quad \quad \rightarrow \ call \ p \\
& \quad \quad \quad \quad \quad \quad \quad \downarrow n_1 \\
& \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \rightarrow \ print \ a \ast b \\
& \quad \quad \quad \quad \quad \quad \quad \downarrow E_{main} \\
X_1.1 & \quad X_2.0 \\
& \quad \quad \quad \quad \quad \rightarrow \ if \ a == 0 \\
& \quad \quad \quad \quad \quad \quad \quad \downarrow n_2 \\
& \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \rightarrow \ a = a - 1 \\
& \quad \quad \quad \quad \quad \quad \quad \downarrow X_1.0 \\
& \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \rightarrow \ call \ p \\
& \quad \quad \quad \quad \quad \quad \quad \downarrow C_2 \\
& \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \rightarrow \ print \ a \ast b \\
& \quad \quad \quad \quad \quad \quad \quad \downarrow E_p \\
X_1.1 & \quad X_2.0 \\
& \quad \quad \quad \quad \quad \rightarrow \ call \ p \\
& \quad \quad \quad \quad \quad \quad \quad \downarrow C_2 \\
& \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \rightarrow \ print \ a \ast b \\
& \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \rightarrow \ print \ a \ast b \\
& \quad \quad \quad \quad \quad \quad \quad \downarrow E_p \\
X_2.0 & \quad \quad \quad \quad \quad \rightarrow \ print \ a \ast b \\
& \quad \quad \quad \quad \quad \quad \quad \downarrow E_p \\
X_0.0 & \quad X_1.1 \\
& \quad \quad \quad \quad \quad \rightarrow \ if \ a == 0 \\
& \quad \quad \quad \quad \quad \quad \quad \downarrow n_2 \\
& \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \rightarrow \ a = a - 1 \\
& \quad \quad \quad \quad \quad \quad \quad \downarrow X_1.0 \\
& \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \rightarrow \ call \ p \\
& \quad \quad \quad \quad \quad \quad \quad \downarrow C_2 \\
& \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \rightarrow \ print \ a \ast b \\
& \quad \quad \quad \quad \quad \quad \quad \downarrow E_p \\
X_1.1 & \quad X_2.0 \\
& \quad \quad \quad \quad \quad \rightarrow \ call \ p \\
& \quad \quad \quad \quad \quad \quad \quad \downarrow C_2 \\
& \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \rightarrow \ print \ a \ast b \\
& \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \rightarrow \ print \ a \ast b \\
& \quad \quad \quad \quad \quad \quad \quad \downarrow E_p \\
X_2.0 & \quad \quad \quad \quad \quad \rightarrow \ print \ a \ast b \\
& \quad \quad \quad \quad \quad \quad \quad \downarrow E_p
\end{align*}
\]

<table>
<thead>
<tr>
<th>Context</th>
<th>exitValue</th>
</tr>
</thead>
<tbody>
<tr>
<td>(X_0 = \langle main,0 \rangle)</td>
<td>1</td>
</tr>
<tr>
<td>(X_1 = \langle p,1 \rangle)</td>
<td>1</td>
</tr>
<tr>
<td>(X_2 = \langle p,0 \rangle)</td>
<td>1</td>
</tr>
</tbody>
</table>
Available Expressions Analysis Using Value Contexts

\[ WL = [X_2|C_2, X_2|n_3, X_2|E_p, X_1|n_3, X_1|E_p, X_0|n_1, X_0|E_m] \]

**Context** | **exitValue**  
--- | ---  
\( X_0 = \langle \text{main}, 0 \rangle \) | 1  
\( X_1 = \langle p, 1 \rangle \) | 1  
\( X_2 = \langle p, 0 \rangle \) | 1  

\( p \) has context \( X_2 \) with value 0 so no need to create a new context
Available Expressions Analysis Using Value Contexts

\[ WL = [X_2|C_2, X_2|n_3, X_2|E_p, X_1|n_3, X_1|E_p, X_0|n_1, X_0|E_m] \]

\[ \begin{array}{c}
X_0 \xrightarrow{C_1} X_1 \xrightarrow{C_2} X_2 \xrightarrow{C_2}
\end{array} \]

<table>
<thead>
<tr>
<th>Context</th>
<th>exitValue</th>
</tr>
</thead>
<tbody>
<tr>
<td>(X_0 = \langle \text{main}, 0 \rangle)</td>
<td>1</td>
</tr>
<tr>
<td>(X_1 = \langle p, 1 \rangle)</td>
<td>1</td>
</tr>
<tr>
<td>(X_2 = \langle p, 0 \rangle)</td>
<td>1</td>
</tr>
</tbody>
</table>

\(p\) has context \(X_2\) with value 0 so no need to create a new context

Record the transition from context \(X_2\) to itself.
Available Expressions Analysis Using Value Contexts

\[ WL = [X_2|C_2, X_2|n_3, X_2|E_p, X_1|n_3, X_1|E_p, X_0|n_1, X_0|E_m] \]

<table>
<thead>
<tr>
<th>Context</th>
<th>exitValue</th>
</tr>
</thead>
<tbody>
<tr>
<td>X_0 = 〈main,0〉</td>
<td>1</td>
</tr>
<tr>
<td>X_1 = 〈p,1〉</td>
<td>1</td>
</tr>
<tr>
<td>X_2 = 〈p,0〉</td>
<td>1</td>
</tr>
</tbody>
</table>

\( p \) has context \( X_2 \) with value 0 so no need to create a new context

Record the transition from context \( X_2 \) to itself

Use the \( exitValue(X_2) \) to compute \( Out_{C_2}[X_2] \)
Available Expressions Analysis Using Value Contexts

\[ WL = [X_2|n_3, X_2|E_p, X_1|n_3, X_1|E_p, X_0|n_1, X_0|E_m] \]

<table>
<thead>
<tr>
<th>Context</th>
<th>exitValue</th>
</tr>
</thead>
<tbody>
<tr>
<td>( X_0 = \langle \text{main}, 0 \rangle )</td>
<td>1</td>
</tr>
<tr>
<td>( X_1 = \langle p, 1 \rangle )</td>
<td>1</td>
</tr>
<tr>
<td>( X_2 = \langle p, 0 \rangle )</td>
<td>1</td>
</tr>
</tbody>
</table>
Available Expressions Analysis Using Value Contexts

WL = \([X_2|n_3, X_2|E_p, X_1|n_3, X_1|E_p, X_0|n_1, X_0|E_m]\)

Context | exitValue
---|---
\(X_0 = \langle \text{main}, 0 \rangle\) | 1
\(X_1 = \langle \text{p}, 1 \rangle\) | 1
\(X_2 = \langle \text{p}, 0 \rangle\) | 1
Available Expressions Analysis Using Value Contexts

\[ WL = [X_2|E_p, X_1|n_3, X_1|E_p, X_0|n_1, X_0|E_m] \]

<table>
<thead>
<tr>
<th>Context</th>
<th>exitValue</th>
</tr>
</thead>
<tbody>
<tr>
<td>( X_0 = \langle \text{main}, 0 \rangle )</td>
<td>1</td>
</tr>
<tr>
<td>( X_1 = \langle \text{p}, 1 \rangle )</td>
<td>1</td>
</tr>
<tr>
<td>( X_2 = \langle \text{p}, 0 \rangle )</td>
<td>1</td>
</tr>
</tbody>
</table>
Available Expressions Analysis Using Value Contexts

\[ WL = [X_2|E_p, X_1|n_3, X_1|E_p, X_0|n_1, X_0|E_m] \]

\[ C_1 \rightarrow X_1 \rightarrow C_2 \rightarrow X_2 \]

At \( E_p \) the values from \( S_p \) and \( n_3 \) are merged for context \( X_2 \)

<table>
<thead>
<tr>
<th>Context</th>
<th>exitValue</th>
</tr>
</thead>
<tbody>
<tr>
<td>( X_0 ) = \langle \text{main,0} \rangle</td>
<td>1</td>
</tr>
<tr>
<td>( X_1 ) = \langle \text{p,1} \rangle</td>
<td>1</td>
</tr>
<tr>
<td>( X_2 ) = \langle \text{p,0} \rangle</td>
<td>1</td>
</tr>
</tbody>
</table>
Available Expressions Analysis Using Value Contexts

\[ WL = [X_2|E_p, X_1|n_3, X_1|E_p, X_0|n_1, X_0|E_m] \]

\[
\begin{array}{c}
X_0 \xrightarrow{C_1} X_1 \xrightarrow{C_2} X_2 \xrightarrow{C_2} X_2
\end{array}
\]

<table>
<thead>
<tr>
<th>Context</th>
<th>exitValue</th>
</tr>
</thead>
<tbody>
<tr>
<td>(X_0 = \langle \text{main},0 \rangle)</td>
<td>1</td>
</tr>
<tr>
<td>(X_1 = \langle p,1 \rangle)</td>
<td>1</td>
</tr>
<tr>
<td>(X_2 = \langle p,0 \rangle)</td>
<td>0</td>
</tr>
</tbody>
</table>

At \(E_p\) the values from \(S_p\) and \(n_3\) are merged for context \(X_2\) 
\(\text{exitValue}(X_2)\) is set to 0
Available Expressions Analysis Using Value Contexts

\[ WL = [X_2|E_p, X_1|n_3, X_1|E_p, X_0|n_1, X_0|E_m] \]

- \( X_0 = \langle \text{main}, 0 \rangle \)
- \( X_1 = \langle \text{p}, 1 \rangle \)
- \( X_2 = \langle \text{p}, 0 \rangle \)

At \( E_p \) the values from \( S_p \) and \( n_3 \) are merged for context \( X_2 \).

\( \text{exitValue}(X_2) \) is set to 0.

Since \( X_2 \) has transitions \( X_1 \xrightarrow{C_2} X_2 \) and \( X_2 \xrightarrow{C_2} X_2 \), \( \text{Out}_{C_2}[X_1] \) and \( \text{Out}_{C_2}[X_2] \) become 0.
Available Expressions Analysis Using Value Contexts

\[ WL = [X_2|n_3, X_1|n_3, X_1|E_p, X_0|n_1, X_0|E_m] \]

\[ X_0 \xrightarrow{C_1} X_1 \xrightarrow{C_2} X_2 \]

\[ X_0.0 \]

\[ S_{main} \]

\[ \text{read } a, b \]

\[ t := a \times b \]

\[ X_0.1 \]

\[ C_1 \]

\[ \text{call } p \]

\[ n_1 \]

\[ \text{print } a \times b \]

\[ E_{main} \]

\[ X_1.1 \]

\[ n_2 \]

\[ a = a - 1 \]

\[ X_1.0 \]

\[ X_2.0 \]

\[ X_1.1 \]

\[ C_2 \]

\[ \text{call } p \]

\[ X_1.0 \]

\[ X_2.0 \]

\[ X_2.0 \]

\[ n_3 \]

\[ t = a \times b \]

\[ X_2.0 \]

\[ X_2.0 \]

\[ X_2.1 \]

\[ Context \]

\[ exitValue \]

\[ X_0 = \langle \text{main}, 0 \rangle \]

\[ 1 \]

\[ X_1 = \langle p, 1 \rangle \]

\[ 1 \]

\[ X_2 = \langle p, 0 \rangle \]

\[ 0 \]

At \( E_p \) the values from \( S_p \) and \( n_3 \) are merged for context \( X_2 \)

\[ \text{exitValue}(X_2) \text{ is set to 0} \]

Since \( X_2 \) has transitions \( X_1 \xrightarrow{C_2} X_2 \)

and \( X_2 \xrightarrow{C_2} X_2 \), \( \text{Out}_{C_2}[X_1] \) and \( \text{Out}_{C_2}[X_2] \) become 0

Since \( \text{Out}_{C_2}[X_2] \) changes, \( X_2|n_3 \) is added to the work list.
Available Expressions Analysis Using Value Contexts

\[ WL = [X_2|n_3, X_1|n_3, X_1|E_p, X_0|n_1, X_0|E_m] \]

<table>
<thead>
<tr>
<th>Context</th>
<th>exitValue</th>
</tr>
</thead>
<tbody>
<tr>
<td>( X_0 = \langle \text{main},0 \rangle )</td>
<td>1</td>
</tr>
<tr>
<td>( X_1 = \langle \text{p},1 \rangle )</td>
<td>1</td>
</tr>
<tr>
<td>( X_2 = \langle \text{p},0 \rangle )</td>
<td>0</td>
</tr>
</tbody>
</table>

There is no change in \( \text{Out}_{n_3}[X_2] \).
Available Expressions Analysis Using Value Contexts

\[ WL = [X_1|n_3, X_1|E_p, X_0|n_1, X_0|E_m] \]

<table>
<thead>
<tr>
<th>Context</th>
<th>exitValue</th>
</tr>
</thead>
<tbody>
<tr>
<td>X_0 = ⟨main,0⟩</td>
<td>1</td>
</tr>
<tr>
<td>X_1 = ⟨p,1⟩</td>
<td>1</td>
</tr>
<tr>
<td>X_2 = ⟨p,0⟩</td>
<td>0</td>
</tr>
</tbody>
</table>

There is no change in \( Out_{n_3}[X_2] \)
Available Expressions Analysis Using Value Contexts

\[ WL = [X_1|n_3, X_1|E_p, X_0|n_1, X_0|E_m] \]

\[
\begin{align*}
X_0.0 & \quad \text{read } a, b \\
& \quad t := a \times b \\
S_{\text{main}} & \quad C_1 \quad \text{call } p \\
& \quad n_1 \quad \text{print } a \times b \\
E_{\text{main}} & \\
X_1.1 \quad \text{if } a == 0 \\
S_p & \quad C_1 \quad C_2 \quad C_2 \\
X_1.0 \quad \text{a = a - 1} \\
n_2 & \quad X_1.0 \quad X_2.0 \\
& \quad X_2.0 \quad X_2.0 \\
& \quad \text{call } p \\
C_2 & \quad \text{call } p \\
X_1.0 & \quad X_2.0 \\
n_3 & \quad X_2.0 \\
t = a \times b \\
X_2.0 & \quad X_2.0 \\
& \quad X_2.0 \\
& \quad X_2.1 \\
E_p & \\
X_2.0 & \\
\end{align*}
\]

<table>
<thead>
<tr>
<th>Context</th>
<th>exitValue</th>
</tr>
</thead>
<tbody>
<tr>
<td>X_0 = \langle \text{main}, 0 \rangle</td>
<td>1</td>
</tr>
<tr>
<td>X_1 = \langle p, 1 \rangle</td>
<td>1</td>
</tr>
<tr>
<td>X_2 = \langle p, 0 \rangle</td>
<td>0</td>
</tr>
</tbody>
</table>

There is no change in \( Out_{n_3} [X_1] \) also.
Available Expressions Analysis Using Value Contexts

\[ WL = [X_1 \mid E_p, X_0 \mid n_1, X_0 \mid E_m] \]

<table>
<thead>
<tr>
<th>Context</th>
<th>exitValue</th>
</tr>
</thead>
<tbody>
<tr>
<td>( X_0 = \langle \text{main}, 0 \rangle )</td>
<td>1</td>
</tr>
<tr>
<td>( X_1 = \langle p, 1 \rangle )</td>
<td>1</td>
</tr>
<tr>
<td>( X_2 = \langle p, 0 \rangle )</td>
<td>0</td>
</tr>
</tbody>
</table>

There is no change in \( Out_{n_3}[X_1] \) also
Available Expressions Analysis Using Value Contexts

\[ WL = [X_1 | E_p, X_0 | n_1, X_0 | E_m] \]

\[ \text{Context} \quad \text{exitValue} \]
\begin{array}{|c|c|}
\hline
X_0 = \langle \text{main}, 0 \rangle & 1 \\
X_1 = \langle p, 1 \rangle & 1 \\
X_2 = \langle p, 0 \rangle & 0 \\
\hline
\end{array}

At \( E_p \) the values from \( S_p \) and \( n_3 \) are merged for context \( X_1 \)
Available Expressions Analysis Using Value Contexts

\[ WL = [X_1|E_p, X_0|n_1, X_0|E_m] \]

### Contexts and Exit Values

<table>
<thead>
<tr>
<th>Context</th>
<th>exitValue</th>
</tr>
</thead>
<tbody>
<tr>
<td>( X_0 = \langle \text{main}, 0 \rangle )</td>
<td>1</td>
</tr>
<tr>
<td>( X_1 = \langle p, 1 \rangle )</td>
<td>1</td>
</tr>
<tr>
<td>( X_2 = \langle p, 0 \rangle )</td>
<td>0</td>
</tr>
</tbody>
</table>

### Diagram Explanation

- **Main Function**
  - Read `a, b`
  - Assign `t := a \times b`
  - Print `a \times b`

- **Procedure Call**
  - If `a == 0`
    - Assign `a = a - 1`
  - Call `p`

- **Exit Value**
  - At \( E_p \) the values from \( S_p \) and \( n_3 \) are merged for context \( X_1 \)
  - `exitValue(X_1)` remains 1
Available Expressions Analysis Using Value Contexts

\[WL = [X_1|E_p, X_0|n_1, X_0|E_m]\]

\[
\begin{align*}
X_0 &= \langle \text{main}, 0 \rangle \\
X_1 &= \langle p, 1 \rangle \\
X_2 &= \langle p, 0 \rangle
\end{align*}
\]

At \(E_p\) the values from \(S_p\) and \(n_3\) are merged for context \(X_1\)

\(\text{exitValue}(X_1)\) remains 1

Since \(X_1\) has transition \(X_0 \xrightarrow{C_1} X_1\), \(\text{Out}_{C_1}[X_0]\) becomes 1
Available Expressions Analysis Using Value Contexts

\[ WL = [X_0|n_1, X_0|E_m] \]
Available Expressions Analysis Using Value Contexts

\[ WL = [X_0|n_1, X_0|E_m] \]
Available Expressions Analysis Using Value Contexts

\[ WL = [X_0 | E_m] \]

\[
\begin{array}{c}
X_0 \xrightarrow{C_1} X_1 \xrightarrow{C_2} X_2 \xrightarrow{C_2} X_0
\end{array}
\]

<table>
<thead>
<tr>
<th>Context</th>
<th>exitValue</th>
</tr>
</thead>
<tbody>
<tr>
<td>(X_0) = (\langle\text{main}, 0\rangle)</td>
<td>1</td>
</tr>
<tr>
<td>(X_1) = (\langle p, 1\rangle)</td>
<td>1</td>
</tr>
<tr>
<td>(X_2) = (\langle p, 0\rangle)</td>
<td>0</td>
</tr>
</tbody>
</table>

\(S_{main}\)

- read \(a, b\)
- \(t := a \times b\)

\(C_1\)

- call \(p\)

\(n_1\)

- print \(a \times b\)

\(E_{main}\)

\(S_p\)

- if \(a == 0\)

\(n_2\)

- \(a = a - 1\)

\(C_2\)

- call \(p\)

\(n_3\)

- \(t = a \times b\)
Available Expressions Analysis Using Value Contexts

\[ WL = [X_0|E_m] \]

Context exitValue
\[
\begin{array}{|c|c|}
\hline
X_0 &= \langle \text{main}, 0 \rangle \\
X_1 &= \langle p, 1 \rangle \\
X_2 &= \langle p, 0 \rangle \\
\hline
\end{array}
\]

Contexts:

- \( X_0 \): \langle \text{main}, 0 \rangle
- \( X_1 \): \langle p, 1 \rangle
- \( X_2 \): \langle p, 0 \rangle

Oct 2015 IIT Bombay
Available Expressions Analysis Using Value Contexts

\[ WL = [X_0 | E_m] \]
Available Expressions Analysis Using Value Contexts

\[ WL = [ ] \]

\[
\text{Context} \quad \text{exitValue}
\begin{array}{|c|c|}
\hline
X_0 = \langle \text{main,0} \rangle & 1 \\
X_1 = \langle \text{p,1} \rangle & 1 \\
X_2 = \langle \text{p,0} \rangle & 0 \\
\hline
\end{array}
\]

Work list is empty and the analysis is over.
## A Trace of Value Context Based Analysis (1)

<table>
<thead>
<tr>
<th>S. No.</th>
<th>Work List</th>
<th>Sel. node</th>
<th>Data flow value</th>
<th>New context</th>
<th>New trans.</th>
<th>exit value</th>
<th>Addition to the work list</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td>$X_0 = \langle m,0 \rangle$</td>
<td>$X_0 .1$</td>
<td>$X_0 \mid S_m, X_0 \mid C_1, X_0 \mid n_1, X_0 \mid E_m$</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>$X_0 \mid S_m, X_0 \mid C_1, X_0 \mid n_1, X_0 \mid E_m$</td>
<td>$S_m$</td>
<td>$Out_{S_m}[X_0] = 1$</td>
<td>$X_1 = \langle p,1 \rangle$</td>
<td>$X_0 \xrightarrow{C_1} X_1$</td>
<td>$X_1 .1$</td>
<td>$X_1 \mid S_p, X_1 \mid n_2, X_1 \mid C_2, X_1 \mid n_3, X_1 \mid E_p$</td>
</tr>
<tr>
<td>3</td>
<td>$X_0 \mid C_1, X_0 \mid n_1, X_0 \mid E_m$</td>
<td>$C_1$</td>
<td>$X_1 = \langle p,1 \rangle$</td>
<td>$X_0 \xrightarrow{C_1} X_1$</td>
<td>$X_1 .1$</td>
<td>$X_1 \mid S_p, X_1 \mid n_2, X_1 \mid C_2, X_1 \mid n_3, X_1 \mid E_p$</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>$X_1 \mid S_p, X_1 \mid n_2, X_1 \mid C_2, X_1 \mid n_3, X_1 \mid E_p, X_0 \mid n_1, X_0 \mid E_m$</td>
<td>$S_p$</td>
<td>$Out_{S_p}[X_1] = 1$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>$X_1 \mid n_2, X_1 \mid C_2, X_1 \mid n_3, X_1 \mid E_p, X_0 \mid n_1, X_0 \mid E_m$</td>
<td>$n_2$</td>
<td>$Out_{n_2}[X_1] = 0$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>$X_1 \mid C_2, X_1 \mid n_3, X_1 \mid E_p, X_0 \mid n_1, X_0 \mid E_m$</td>
<td>$C_2$</td>
<td>$X_2 = \langle p,0 \rangle$</td>
<td>$X_1 \xrightarrow{C_2} X_2$</td>
<td>$X_2 .1$</td>
<td>$X_2 \mid S_p, X_2 \mid n_2, X_2 \mid C_2, X_2 \mid n_3, X_2 \mid E_p$</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>$X_2 \mid S_p, X_2 \mid n_2, X_2 \mid C_2, X_2 \mid n_3, X_1 \mid E_p, X_0 \mid n_1, X_0 \mid E_m$</td>
<td>$S_p$</td>
<td>$Out_{S_p}[X_2] = 0$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
# A Trace of Value Context Based Analysis (2)

<table>
<thead>
<tr>
<th>S. No.</th>
<th>Work List</th>
<th>Sel. node</th>
<th>Data flow value</th>
<th>New context</th>
<th>New trans.</th>
<th>exit value</th>
<th>Addition to the work list</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>$X_2</td>
<td>n_2, X_2</td>
<td>C_2, X_2</td>
<td>n_3, X_2</td>
<td>E_p, X_1</td>
<td>n_3, X_1</td>
<td>E_p, X_0</td>
</tr>
<tr>
<td>9</td>
<td>$X_2</td>
<td>C_2, X_2</td>
<td>n_3, X_2</td>
<td>E_p, X_1</td>
<td>n_3, X_1</td>
<td>E_p, X_0</td>
<td>n_1, X_0</td>
</tr>
<tr>
<td>10</td>
<td>$X_2</td>
<td>n_3, X_2</td>
<td>E_p, X_1</td>
<td>n_3, X_1</td>
<td>E_p, X_0</td>
<td>n_1, X_0</td>
<td>E_m$</td>
</tr>
<tr>
<td>11</td>
<td>$X_2</td>
<td>E_p, X_1</td>
<td>n_3, X_1</td>
<td>E_p, X_0</td>
<td>n_1, X_0</td>
<td>E_m$</td>
<td>$E_p$</td>
</tr>
<tr>
<td>12</td>
<td>$X_2</td>
<td>n_3, X_1</td>
<td>n_3, X_1</td>
<td>E_p, X_0</td>
<td>n_1, X_0</td>
<td>E_m$</td>
<td>$n_3$</td>
</tr>
<tr>
<td>13</td>
<td>$X_1</td>
<td>n_3, X_1</td>
<td>E_p, X_0</td>
<td>n_1, X_0</td>
<td>E_m$</td>
<td>$n_3$</td>
<td>$Out_{n_3}[X_1]=1$</td>
</tr>
<tr>
<td>14</td>
<td>$X_1</td>
<td>E_p, X_0</td>
<td>n_1, X_0</td>
<td>E_m$</td>
<td>$E_p$</td>
<td>$Out_{E_p}[X_1]=1$</td>
<td>$Out_{C_1}[X_0]=1$</td>
</tr>
<tr>
<td>15</td>
<td>$X_0</td>
<td>n_1, X_0</td>
<td>E_m$</td>
<td>$n_1$</td>
<td>$Out_{n_1}[X_0]=1$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>$X_0</td>
<td>E_m$</td>
<td>$E_m$</td>
<td>$Out_{E_m}[X_0]=1$</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
1. int a, b, c;
2. void main()
3. {
4.     c = a*b;
5. }
6. void p()
7. {
8.     p();
9.     Is a*b available?
10.    a = a*b;
11. }
12. }

```
1. int a, b, c;
2. void main()
3. {
4.     c = a*b;
5.     p();
6. }
7. void p()
8. {
9.     p();
10.    Is a*b available?
11.    a = a*b;
12. }
```
Tutorial Problem #2 for Value Contexts

Perform interprocedural live variables analysis using value contexts

```
main()
{
    p();
}

p()
{
    while (...)
    {
        printf ("%d\n",a);
        p();
    }
}
```

Observe the change in edges in the transition diagram
Perform interprocedural available expressions analysis using value contexts

```
main()
{
    c = a*b;
p();
}

p()
{
    while (a > b)
    {
        p();
a = a*b;
    }
}
```

Observe the change in edges in the transition diagram
Tutorial Problem #4 for Value Contexts

Perform interprocedural available expressions analysis using value contexts

```
1. main()
2. {
3.     c = a*b;
4.     p();
5.     a = a*b;
6. }
7. p()
8. {   if (...)  
9.     {  a = a*b;
10.     p();
11. }
12. else if (...)  
13. {  c = a * b;
14.     p();
15.     c = a;
16. }
17. else
18.     ; /* ignore */
19. }
```
Tutorial Problem #5 for Value Contexts

Perform interprocedural live variables analysis using value contexts

```c
main()
{
    a = 5; b = 3;
    c = 7; d = 2;
    p();
    a = a + 2;
    e = c+d;
    d = a*b;
    q();
    print a+c+e;
}

p()
{
    b = 2;
    if (b<d)
        c = a+b;
    else
        q();
    print c+d;
}

q()
{
    a = 1;
    p();
    a = a*b;
}
```

Context sensitivity: e is live on entry to p but not before its call in main
Result of Tutorial #5

main()
{
    a = 5; b = 3;
    c = 7; d = 2;
    /*{a,d}*/
    p();
    /*{a,b,c,d}*/
    a = a + 2;
    e = c+d;
    /*{a,b,e}*/
    d = a*b;
    /*{d,e}*/
    q();
    /*{a,c,e}*/
    print a+c+e;
}

p()
{
    /*{a,d,e}*/
    b = 2;
    if (b<d)
    /*{a,b,d,e}*/
        c = a+b;
    else
    /*{d,e}*/
        p();
    /*{a,b,c,d,e}*/
    print c+d;
}

q()
{
    /*{d,e}*/
    a = 1;
    /*{a,d,e}*/
    p();
    /*{a,b,c,d,e}*/
    a = a*b;
}
**Tutorial Problem #6: Interprocedural Points-to Analysis**

```c
main()
{   x = &y;
    z = &x;
    y = &z;
    p(); /* C1 */
}

p()
{   if (...)
    {   p(); /* C2 */
        x = *x;
    }
}
```

Value contexts method requires three contexts as shown below in the transition diagram.

![Transition Diagram](chart.png)
Reaching Definitions Analysis in GCC 4.0

<table>
<thead>
<tr>
<th>Program</th>
<th>LoC</th>
<th>#F</th>
<th>#C</th>
<th>3K length bound</th>
<th>Proposed Approach</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>K</td>
<td>#CS</td>
<td>Max</td>
<td>Time</td>
<td>#CS</td>
</tr>
<tr>
<td>hanoi</td>
<td>33</td>
<td>2</td>
<td>4</td>
<td>100000+</td>
<td>99922</td>
</tr>
<tr>
<td>bit_gray</td>
<td>53</td>
<td>5</td>
<td>11</td>
<td>100000+</td>
<td>31374</td>
</tr>
<tr>
<td>analyzer</td>
<td>288</td>
<td>14</td>
<td>20</td>
<td>2</td>
<td>21</td>
</tr>
<tr>
<td>distray</td>
<td>331</td>
<td>9</td>
<td>21</td>
<td>6</td>
<td>96</td>
</tr>
<tr>
<td>mason</td>
<td>350</td>
<td>9</td>
<td>13</td>
<td>8</td>
<td>100000+</td>
</tr>
<tr>
<td>fourinarow</td>
<td>676</td>
<td>17</td>
<td>45</td>
<td>5</td>
<td>510</td>
</tr>
<tr>
<td>sim</td>
<td>1146</td>
<td>13</td>
<td>45</td>
<td>8</td>
<td>100000+</td>
</tr>
<tr>
<td>181_mcf</td>
<td>1299</td>
<td>17</td>
<td>24</td>
<td>6</td>
<td>32789</td>
</tr>
<tr>
<td>256_bzip2</td>
<td>3320</td>
<td>63</td>
<td>198</td>
<td>7</td>
<td>492</td>
</tr>
</tbody>
</table>

- LoC is the number of lines of code,
- #F is the number of procedures,
- #C is the number of call sites,
- #CS is the number of call strings
- Max denotes the maximum number of call strings reaching any node.
- Analysis time is in milliseconds.

(Implementation was carried out by Seema Ravandale.)
Some Observations

- Compromising on precision may not be necessary for efficiency.
- Separating the necessary information from redundant information is much more significant.
- Data flow propagation in real programs seems to involve only a small subset of all possible values.
  Much fewer changes than the theoretically possible worst case number of changes.
- A precise modelling of the process of analysis is often an eye opener.
Some Observations

• Compromising on precision may not be necessary for efficiency.

• Separating the necessary information from redundant information is much more significant.

• Data flow propagation in real programs seems to involve only a small subset of all possible values.
  Much fewer changes than the theoretically possible worst case number of changes.

• A precise modelling of the process of analysis is often an eye opener.

\[
\# \text{ distinct tagged values} = \min (\# \text{ actual contexts}, \# \text{ actual data flow values})
\]