Program Slicing

Author: Mark Weiser
Published in TSE, 1984

Presented by Peeratham (Karn) Techapalokul
10/13/2015
About Mark Weiser

- a chief scientist at Xerox PARC
- Widely considered to be the father of ubiquitous computing

“My research interests are garbage collection, operating systems, user interfaces, and ubiquitous computing. I used to work on software engineering and program slicing, but not much any more.” – Weiser
Outline

• Definitions
• Finding slice (manually)
• Applications of Program slicing
• Finding slice using data flow analysis (Intraprocedural slicing)
• Interprocedural slicing
• Testing the slicer on student compiler programs
What is Program Slicing?

• A program slice $S$ is a *reduced, executable program* obtained from a program $P$ by removing statements, such that the program slice $S$ replicates part of the behavior of program $P$.

• Program slicing is the computation of a set of program statements (the program slice) that can possibly affects the values at some points of interest (slicing criterion).
Applications of program slicing:

- Debugging
- Parallelization
- Software maintenance
- Testing
- Reverse engineering
- Compiler tuning
- Security
Finding Slices

Slicing criterion $C = <\text{statement, variables}>$

< 10, \{\text{product}\}>

(1) read (n);
(2) i := 1;
(3) sum := 0;
(4) product := 1;
(5) while $i \leq n$ do 
   begin 
   (6) sum := sum + i;
   (7) product := product * i;
   (8) $i := i + 1$
   end;
(9) write(sum);
(10) write(product)
Finding Slices

(1) read (n);
(2) i := 1;
(3) sum := 0;
(4) product := 1;
(5) while i <= n do
   begin
      (6) sum := sum + i;
      (7) product := product * i;
      (8) i := i + 1
   end;
(9) write(sum);
(10) write(product)

Slice on criterion < 10, {product}>

read (n);
i := 1;
product := 1;
while i <= n do
   begin
      product := product * i;
i := i + 1
   end;
write(product)
CFG of the example program

Figure 3: CFG of the example program of Figure 1 (a).
Overview: finding program slices

Two types of iteration:

1. Tracing transitive data dependences
   - determine directly relevant variables $R_C^0$
   - derive $S_C^0$ from $R_C^0$

2. Tracing control dependences (dealing with branch statement)
   - \text{INFL} (b) : set of statements control dependent on b

\begin{verbatim}
1 READ (X)
2 IF X < 1
3 THEN Z := 1
4 ELSE Z := 2
5 WRITE (Z).
\end{verbatim}
Tracing transitive data dependences (1)

1. Determine \textit{directly relevant variables} $R_C^0(i)$ at each node $i$ in the CFG
   
   \begin{itemize}
   \item starts with initial values $R_C^0(n) = V$, $R_C^0(m) = \emptyset$ for any node $m \neq n$
   \item requires iteration in the presence of loop
   \end{itemize}

   \begin{align*}
   R_C^0(i) &= R_C^0(i) \cup \{ v \mid v \in R_C^0(j), v \notin \text{DEF}(i) \} \\
   &\quad \cup \{ v \mid v \in \text{REF}(i), \text{DEF}(i) \cap R_C^0(j) \neq \emptyset \}
   \end{align*}
< 10 , {product}>
Tracing transitive data dependences (2)

2. A set of *directly relevant statements*, $S_C^0$, is derived from $R_C^0$ at each node $i$ in the CFG

$$S_C^0 = \{ i \mid (\text{DEF}(i) \cap R_C^0(j)) \neq \emptyset, i \rightarrow_{\text{CFG}} j \}$$
< 10 , \{\text{product}\}>

<table>
<thead>
<tr>
<th>Node #</th>
<th>Def</th>
<th>Ref</th>
<th>INFL</th>
<th>$R^0_C$</th>
<th>$\ln S^0_C$</th>
<th>$\ln B^0_C$</th>
<th>$R^1_C$</th>
<th>$\ln S^1_C$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>{n}</td>
<td>{}</td>
<td>{}</td>
<td>{}</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>{i}</td>
<td>{}</td>
<td>{}</td>
<td>{}</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>{\text{sum}}</td>
<td>{}</td>
<td>{}</td>
<td>{i}</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>{\text{product}}</td>
<td>{}</td>
<td>{}</td>
<td>{i}</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>{}</td>
<td>{i, n}</td>
<td>{}</td>
<td>{\text{product, i}}</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>{\text{sum}}</td>
<td>{\text{sum, i}}</td>
<td>{}</td>
<td>{\text{product, i}}</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>{\text{product}}</td>
<td>{\text{product, i}}</td>
<td>{}</td>
<td>{\text{product, i}}</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>{i}</td>
<td>{i}</td>
<td>{}</td>
<td>{\text{product, i}}</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>{}</td>
<td>{\text{sum}}</td>
<td>{}</td>
<td>{\text{product}}</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>{}</td>
<td>{\text{product}}</td>
<td>{}</td>
<td>{\text{product}}</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Tracing control dependences (dealing with branch statement)

• INFL(b) is set of statements that is control dependent on branch statement b
• branching statement b is indirectly relevant to the slice if there is at least one directly relevant statement under its range of influence

\[ B_C^k = \{ b \mid \exists i \in S_C^k, i \in \text{INFL}(b) \} \]
< 10 , {product}>

<table>
<thead>
<tr>
<th>Node #</th>
<th>Def</th>
<th>Ref</th>
<th>INFL</th>
<th>$R_C^0$</th>
<th>$In S_C^0$</th>
<th>$In B_C^0$</th>
<th>$R_C^1$</th>
<th>$In S_C^1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>{n}</td>
<td>{}</td>
<td>{}</td>
<td>{}</td>
<td>{}</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>{i}</td>
<td>{}</td>
<td>{}</td>
<td>{}</td>
<td>/</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>{sum}</td>
<td>{}</td>
<td>{}</td>
<td>{i}</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>{product}</td>
<td>{}</td>
<td>{}</td>
<td>{i}</td>
<td>/</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>{}</td>
<td>{i, n}</td>
<td>{6,7,8}</td>
<td>{product, i}</td>
<td>/</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>{sum}</td>
<td>{sum, i}</td>
<td>{}</td>
<td>{product, i}</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>{product}</td>
<td>{product, i}</td>
<td>{}</td>
<td>{product, i}</td>
<td>/</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>{i}</td>
<td>{i}</td>
<td>{}</td>
<td>{product, i}</td>
<td>/</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>{}</td>
<td>{sum}</td>
<td>{}</td>
<td>{product}</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>{}</td>
<td>{product}</td>
<td>{}</td>
<td>{product}</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Tracing control dependences (cont’d) (dealing with branch statement)

- Trace relevant variables and statements with direct influence on $B_C^0$

$$R_C^{k+1}(i) = R_C^k(i) \cup \bigcup_{b \in B_C^k} R_{(b, \text{REF}(b))}^0(i)$$

$$S_C^{k+1} = B_C^k \cup \{ i \mid \text{DEF}(i) \cap R_C^{k+1}(j) \neq \emptyset, i \rightarrow_{\text{CFG}} j \}$$

- The sets $R_C^k$ and $S_C^k$ are nondecreasing subsets of the program’s variables and statements respectively
- The fixpoint of the computation of the $S_C^k$ sets => the desired program slice.
### Node Table

<table>
<thead>
<tr>
<th>Node #</th>
<th>Def</th>
<th>Ref</th>
<th>INFL</th>
<th>$R^0_C$</th>
<th>$\ln S^0_C$</th>
<th>$\ln B^0_C$</th>
<th>$R^1_C$</th>
<th>$\ln S^1_C$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>{n}</td>
<td>{}</td>
<td>{}</td>
<td>{}</td>
<td>{}</td>
<td></td>
<td>{}</td>
<td>/</td>
</tr>
<tr>
<td>2</td>
<td>{i}</td>
<td>{}</td>
<td>{}</td>
<td>{}</td>
<td>/</td>
<td>{n}</td>
<td>/</td>
<td>/</td>
</tr>
<tr>
<td>3</td>
<td>{sum}</td>
<td>{}</td>
<td>{}</td>
<td>{i}</td>
<td></td>
<td>{i, n}</td>
<td>/</td>
<td>/</td>
</tr>
<tr>
<td>4</td>
<td>{product}</td>
<td>{}</td>
<td>{}</td>
<td>{i}</td>
<td>/</td>
<td>{i, n}</td>
<td>/</td>
<td>/</td>
</tr>
<tr>
<td>5</td>
<td>{}</td>
<td>{i, n}</td>
<td>{6,7,8}</td>
<td>{product, i}</td>
<td>/</td>
<td>{product, i, n}</td>
<td>/</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>{sum}</td>
<td>{sum, i}</td>
<td>{}</td>
<td>{product, i}</td>
<td>{product, i,n}</td>
<td>/</td>
<td>/</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>{product}</td>
<td>{product, i}</td>
<td>{}</td>
<td>{product, i}</td>
<td>/</td>
<td>{product, i,n}</td>
<td>/</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>{i}</td>
<td>{i}</td>
<td>{}</td>
<td>{product, i}</td>
<td>/</td>
<td>{product, i,n}</td>
<td>/</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>{}</td>
<td>{sum}</td>
<td>{}</td>
<td>{product}</td>
<td>{product}</td>
<td>/</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>{}</td>
<td>{product}</td>
<td>{}</td>
<td>{product}</td>
<td>{product}</td>
<td>/</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
(1) read (n);
(2) i := 1;
(3) sum := 0;
(4) product := 1;
(5) while i <= n do
    begin
(6)      sum := sum + i;
(7)      product := product * i;
(8)      i := i + 1
    end;
(9) write(sum);
(10) write(product)

<table>
<thead>
<tr>
<th>Node #</th>
<th>Def</th>
<th>Ref</th>
<th>INFL</th>
<th>$R^0_C$</th>
<th>In $S^0_C$</th>
<th>In $B^0_C$</th>
<th>$R^1_C$</th>
<th>In $S^1_C$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>{n}</td>
<td>∅</td>
<td>∅</td>
<td>∅</td>
<td>∅</td>
<td></td>
<td>∅</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>{i}</td>
<td>∅</td>
<td>∅</td>
<td>∅</td>
<td></td>
<td></td>
<td>{n}</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>{sum}</td>
<td>∅</td>
<td>∅</td>
<td>{i}</td>
<td></td>
<td></td>
<td>{i, n}</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>{product}</td>
<td>∅</td>
<td>∅</td>
<td>{i}</td>
<td></td>
<td></td>
<td>{i, n}</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>∅</td>
<td>{i, n}</td>
<td>{6,7,8}</td>
<td>{product, i}</td>
<td></td>
<td></td>
<td>{product, i, n}</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>{sum}</td>
<td>{sum, i}</td>
<td>∅</td>
<td>{product, i}</td>
<td></td>
<td></td>
<td>{product, i, n}</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>{product}</td>
<td>{product, i}</td>
<td>∅</td>
<td>{product, i}</td>
<td></td>
<td></td>
<td>{product, i, n}</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>{i}</td>
<td>{i}</td>
<td>∅</td>
<td>{product, i}</td>
<td></td>
<td></td>
<td>{product, i, n}</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>∅</td>
<td>{sum}</td>
<td>∅</td>
<td>{product}</td>
<td></td>
<td></td>
<td>{product}</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>∅</td>
<td>{product}</td>
<td>∅</td>
<td>{product}</td>
<td></td>
<td></td>
<td>{product}</td>
<td></td>
</tr>
</tbody>
</table>
Interprocedural Slicing

- Compute interprocedural summary information for each procedure $P$
  - $\text{MOD}(P) =$ variables that may be modified by $P$
  - $\text{USE}(P) =$ variables that may be used by $P$
- Generation of new slicing criteria
  Translate relevant variables $R_c$ into the scope of new procedure
P is sliced, P calls Q
generates: `<last statement of Q, relevant vars in P in the scope of Q>`

Q is sliced, Q is called by P
generates: `<first statement in P, relevant vars in Q in the scope of P>`
Interprocedural Slicing (cont’d)

• **UP** maps set $C$ of slicing criteria in a procedure $P$ to a set of criteria in **procedures that call $P$**

• **DOWN** maps set $C$ of slicing criteria in a procedure $P$ to a set of criteria in **procedures called by $P$**

• The complete interprocedural slice for a criterion $C$ = union of the intraprocedural slices for each criterion in $(UP \cup DOWN)^*\{ C \}$

• Interprocedurally imprecise because it does not model calling contexts
A Sampling of Slices

• Test the program slicer on 19 student compilers (500 – 900 executable statements long; 20 – 80 subroutines)

• The compilers were sliced at each write statement $i$ and a set of output variables $V$

• Slices differed by less than 30 statements were merged into a new slightly large slice
Statistics on slices

**TABLE I**

<table>
<thead>
<tr>
<th>Measure</th>
<th>Mean</th>
<th>Median</th>
<th>Min</th>
<th>Max</th>
</tr>
</thead>
<tbody>
<tr>
<td>Useless</td>
<td>9.16</td>
<td>6</td>
<td>1</td>
<td>23</td>
</tr>
<tr>
<td>Common</td>
<td>14.32</td>
<td>0</td>
<td>0</td>
<td>86</td>
</tr>
<tr>
<td>Slices</td>
<td>37.26</td>
<td>32</td>
<td>7</td>
<td>74</td>
</tr>
<tr>
<td>Clusters</td>
<td>9.74</td>
<td>7</td>
<td>3</td>
<td>25</td>
</tr>
</tbody>
</table>

Per program measures $N = 19$

Length of contiguous statements in a cluster which were contiguous in original program $= 11.78$

Low uniqueness of slices reflects high degree of interrelatedness of compiler programs
References:


