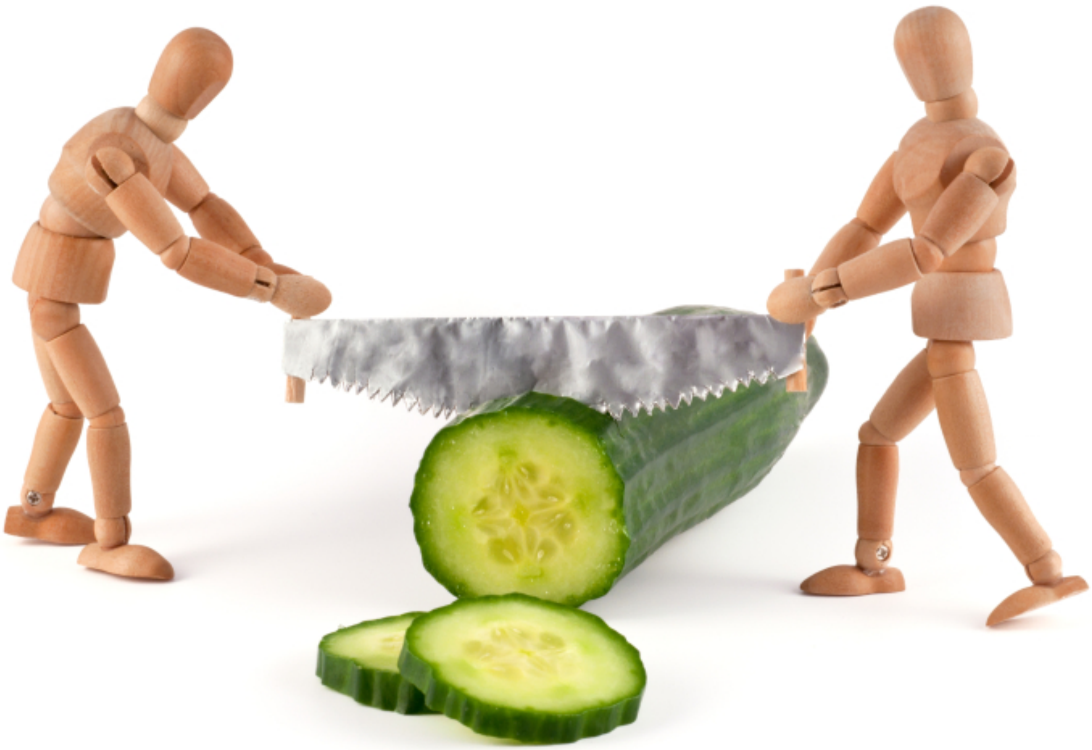
Program Slicing

Part 2

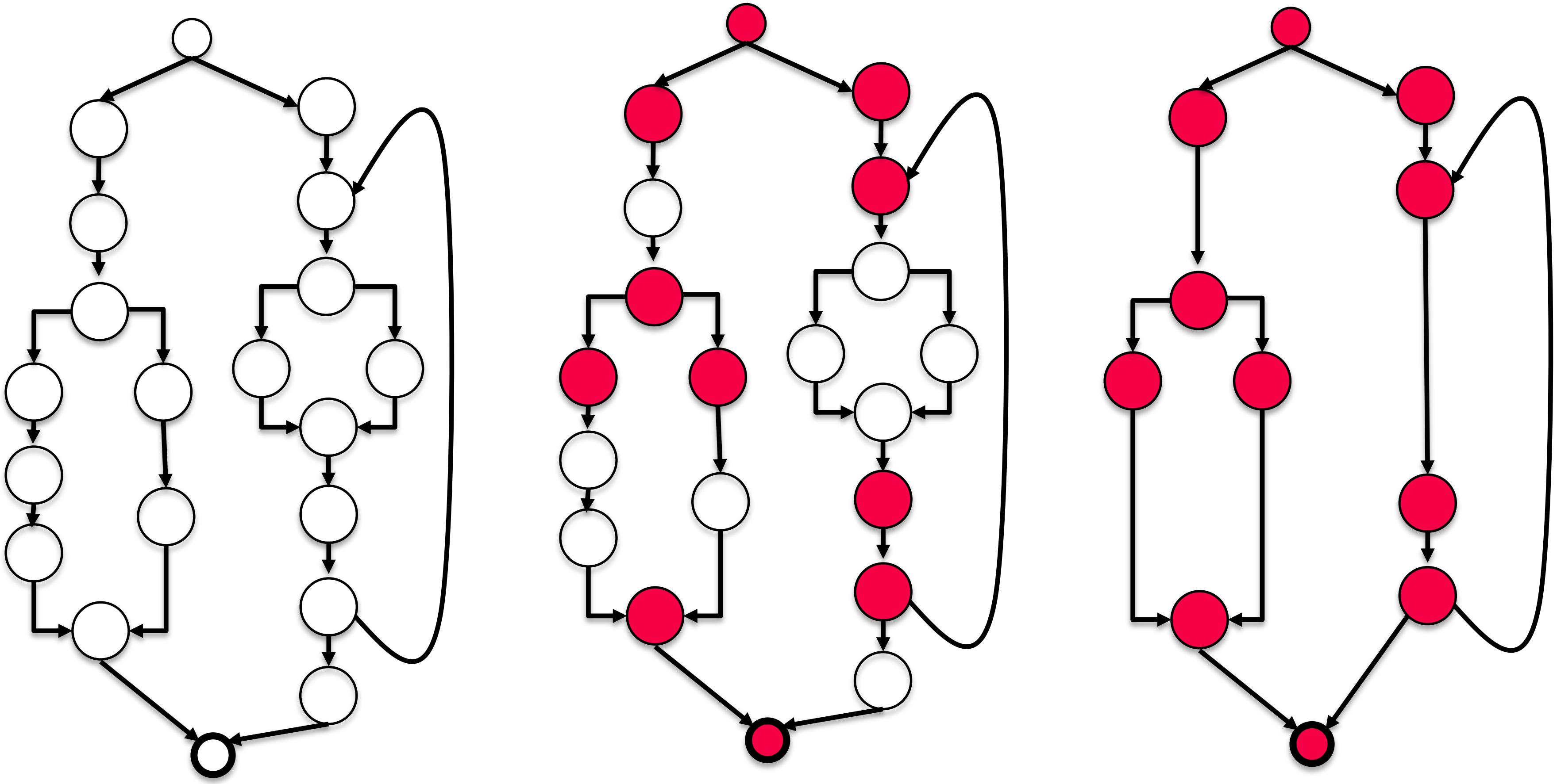
soma@ist.tugraz.at



Folien freundlicherweise zur Verfügung gestellt von Dr. Birgit Hofer

Quelle: TUGraz

Motivation



Outline

- Recap:
  - Slicing with the PDG
  - Slicing with the relevant variables table (1)
- Static Slicing continuation
  - Static slicing with relevant variables table (2)
- Dynamic Slicing
  - Motivation
  - Algorithm & Examples
- Types of Slices
- Open Problems, Applications and Tools

Why Program Slicing?

Example 1

```
1. begin
2.     z = 4 + a;
3.     ~~y = z + 1~~;
4.     x = 5 + z;
5. end
```

Slicing Criterion: (5, {x})

What is a Slice?

- *A Slice is a (****reduced****) program, that* ***preserves*** *the original program's* ***behavior*** *for a* ***given*** *set of* ***variables*** *at a chosen point in a program*

- Slicing criterion C = (i,V)
  - with line number i, variables of interest V

- Characteristics:
  - Defined for variables and a given point
  - A slice is itself a program
  - Ignoring other (irrelevant) statements
  - Focus on relevant parts

Program dependency graph

- **Directed** graph of a single procedure in a program
- **Statements** are the **nodes**
- Data and control **dependencies** are the **edges**
  - Control dependency
    - Statements in the branches of *if* or *while* statements are control dependent on the control predicate
  - Data dependency
    - Node j is data dependent on node i if:
      - Variable x is defined in statement i
      - Variable x is referenced/used in statement j
      - there exists a path in the CFG from i to j without intervening definitions of x

PDG != CFG

Example *

```
1.  begin
2.      a = 0 ;
3.      n = 0 ;
4.      while ( i < n) do
5.          begin
6.              a = b ;
7.              b = i ;
8.              i = i + 1 ;
9.          end;
10.     od;
11. end;
```

Slicing Criterion:
(11, {a})

Static slicing with relevant variables table

Static Slicing – Definitions

- Slicing criterion $C = (i, V)$
  - with line number i, variables of interest V
- $DEF(n)$
  - Set of variables that are defined in line n
- $REF(n)$
  - Set of variables that are referenced/used in line n
- $PRE(n)$
  - Set of predecessor lines of n

Algorithm 1

- Input: Program P, $C = (i, V)$
- Output: Static Slice S_C

1. Compute all relevant Variables $R_C(n)$ backwards from line i to line 1
   - n… current line
   - m…successor line of n ($PRE(m)$ $contains$ n)

- $R_C(n) =$

1	all $v \in V$	if $n = i$ (base case)
2a	all $v \in REF(n) \cup (R_c(m) \backslash \mathrm{DEF}(n))$	if $\exists \{w \mid w \in DEF(n) \wedge w \in R_c(m)\}$
2b	all $w \in R_C(m)$	where $w \notin DEF(n) \wedge w \in R_c(m)$

2. Compute the Slice S_C
   - comprises all statements n where $R_C(m) \cap DEF(n) \neq \emptyset$

Example *

Slicing Criterion: (11, {a})

```
1.  begin
2.      a = 0 ;
3.      n = 0 ;
4.      while ( i < n) do
5.          begin
6.              a = b ;
7.              b = i ;
8.              i = i + 1 ;
9.          end;
10.     od;
11. end;
```

Outline

- Recap:
  - Slicing with the PDG
  - Slicing with the relevant variables table (1)
- Static Slicing continuation
  - Static slicing with relevant variables table (2)
- Dynamic Slicing
  - Motivation
  - Algorithm & Examples
  - Types of Slices
  - Open Problems, Applications and Tools

Algorithm 2

- Influence set $INFL(b)$
  - Set of statements influenced by the control statement b
- Branch statement B_C
  - Control statement influencing a statement in the slice
- **Improved algorithm**:

1. Compute R_C^0 and S_C^0
2. Compute $B_C = \bigcup_{b \in P} \{b | j \in S_C^0, j \in INFL(b)\}$
3. Compute the slice S_C^1

   $S_C^1 = S_C^0 \cup B_C$

Example *

```
1.  begin
2.      a = 0 ;
3.      n = 0 ;
4.      while ( i < n) do
5.          begin
6.              a = b ;
7.              b = i ;
8.              i = i + 1 ;
9.          end;
10.     od;
11. end;
```

Slicing Criterion:
(11, {a})

Algorithm 3

1. Compute R_C^0 and S_C^0
2. For $i \geq 0$
   a. Compute control flow statements influencing statements in the slice:

   $$B_C^i = \bigcup_{b \in P} \{b \mid j \in S_C^i, j \in INFL(b)\}$$

   b. Compute statements which influence variables referenced in B_C^i :

   $$R_C^{i+1}(n) = R_C^i(n) \cup \bigcup_{b \in B_C^i} R^0_{(b,REF(b))}(n)$$

   c. Compute the slice S_C^{i+1}: $S_C^{i+1} = S_C^i \cup B_C^i$

Example **

```
1.  begin
2.      x = 10 ;
3.      if ( a < 0) then
4.          begin
5.              b = a ;
6.          end;
7.      else
8.          begin
9.              x = x + b ;
10.         end;
11.     fi;
12.     res = x * -1;
13. end;
```

Slicing Criterion: (13, {res})

Control of Study Objectives

```
1.  begin
2.      x = 8;
3.      y = x + 3;
4.      if (y > 10) then
5.          begin
6.            a = b - 2;
7.          end
8.      else
9.          begin
10.           x = b * a;
11.         end
12.     fi;
13.     b = a + x - 4;
14. end
```

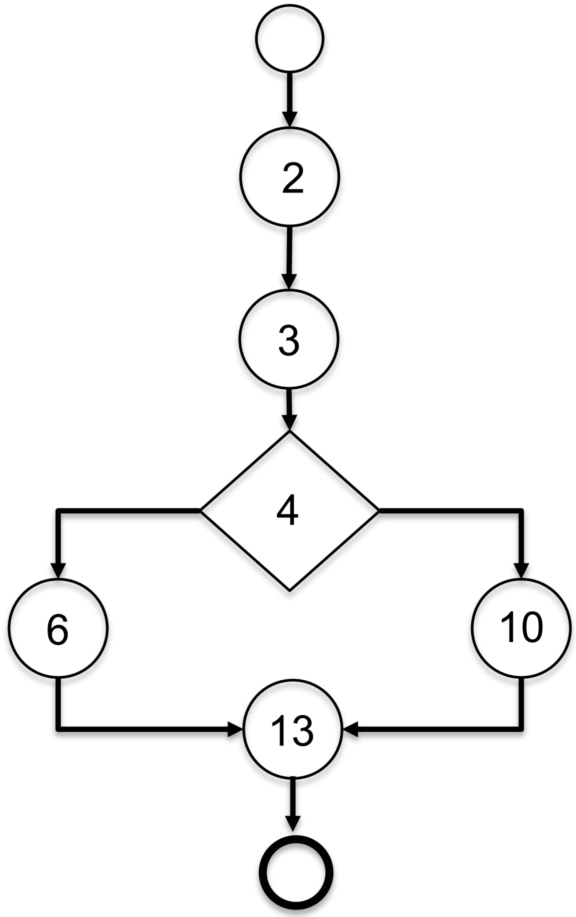
1.) Draw the Control Flow Graph (CFG) for the program.

2.) Draw the Program Dependency Graph (PDG) for the program. Show the different dependencies in a graphical way. Afterwards, compute the static slice for the slicing criterion < 14,{b}> using the PDG.

3.) Compute the static slice for the slicing criterion < 14, {b}> using a table!

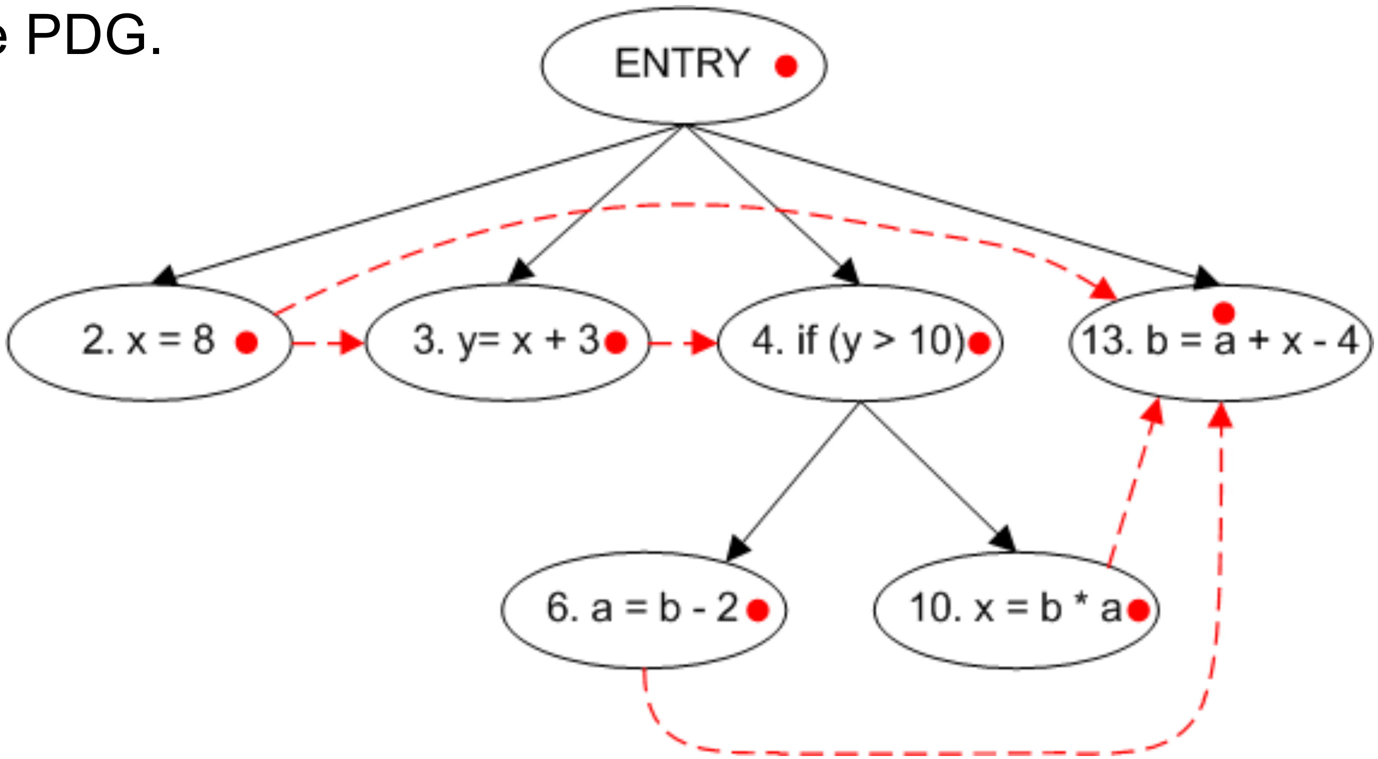
Control of Study Objectives

1.) Draw the Control Flow Graph (CFG) for the program.



Control of Study Objectives

2.) Draw the Program Dependency Graph (PDG) for the program. Show the different dependencies in a graphical way.
Afterwards, compute the static slice for the slicing criterion < 14,{b}> using the PDG.



$$S^1_{(14,\{b\})} = \{2,3,4,6,10,13\}$$

Control of Study Objectives

3.) Compute the static slice for the slicing criterion < 14, {b}> using a table!

n	Pre	Ref	Def	$R^0_{(14,\{b\})}$	$S^0_{(14,\{b\})}$	INFL	B	$R^0_{(4,\{y\})}$	$S^0_{(4,\{y\})}$	$S^1_{(14,\{b\})}$
2	-	-	{x}	{b,a}	●			-	●	●
3	2	{x}	{y}	{b,x,a}				{x}	●	●
4	3	{y}	-	{b,x,a}		6,10	●	{y}		●
6	4	{b}	{a}	{b,x}	●					●
10	4	{b, a}	{x}	{b,a}	●					●
13	6,10	{a,x}	{b}	{a,x}	●					●
14	13	-	-	{b}						

$$S^1_{(14,\{b\})} = \{2,3,4,6,10,13\}$$

Outline

- Recap:
  - Slicing with the PDG
  - Slicing with the relevant variables table (1)
- Static Slicing continuation
  - Static slicing with relevant variables table (2)
  - Types of Slices
  - Open Problems, Applications and Tools
- Dynamic Slicing
  - Motivation
  - Algorithm & Examples

Why dynamic slicing?

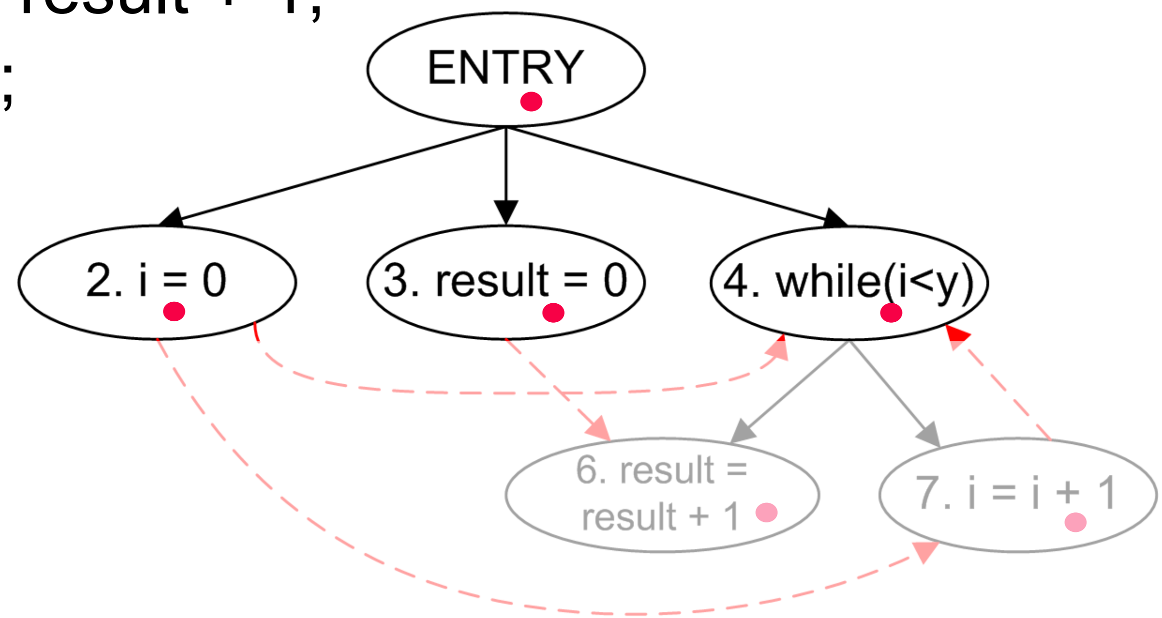
- Static slices too big
- Focus on executed parts

Example 1

Slicing Criterion: (11, {result})

Test Case
x = 3
y = 0

```
1.  begin
2.      i = 0;
3.      result = 0; // Bug  should be result = x;
4.      while (i < y) do
5.      begin
6.          result = result + 1;
7.          i = i + 1;
8.      end;
9.      od;
10.     . . .
11. end;
```



Dynamic Slicing

- Slicing Criterion
  - $C = (x, I^n, V)$ with
    - program input x = test case (TC)
    - n^{th} element of the trajectory I^n
    - and a set of variables of interest V
  - Example: $C = (\{a = 4\}, 5^3, \{x\})$

Dynamic slicing

Dynamic Slicing - Algorithm

1. Compute execution trace (ET) for TC x
2. Draw a directed graph (Execution trace graph (ETG)) with
   - Execution trace elements as nodes
   - Dependencies as edges
     - Data dependency (Definition-Use)
     - Control dependency (Test-Control)
3. Mark all nodes m ($m \leq n$) in the graph
   - that redefine a variable $v_1 \in v$ and
   - there is no other node i ($m < i \leq n$) that redefines v_1
4. Trace back dependencies and mark visited nodes
   - the marked nodes are the slice

Example 1a

```
1.  begin
2.      i = 0;
3.      result = 0; // Bug  should be result = x;
4.      while (i < y) do
5.      begin
6.          result = result + 1;
7.          i = i + 1;
8.      end;
9.      od;
10.     . . .
11. end;
```

Slicing Criterion

({x=3, y=0}, 10^4, {result})

Example 1b

```
1.  begin
2.      i = 0;
3.      result = 0; // Bug  should be result = x;
4.      while (i < y) do
5.      begin
6.          result = result + 1;
7.          i = i + 1;
8.      end;
9.      od;
10.     . . .
11. end;
```

Slicing Criterion

({x=3, y=2}, 10^{10}, {result})

Example 2

```
1.  begin
2.      i = 0;
3.      while (i < n) do
4.      begin
5.          x = x + 1;
6.          i = i + 1;
7.      end;
8.      od;
9.      . . .
10. end;
```

Slicing Criterion

$(\{n=1, x=1\}, 9^6, \{x\})$

Problems with dynamic slicing

Non-terminating dynamic slices:

Statement responsible for incrementing a variable used in the while-condition is not part of the slice

Solution

Symmetric dependencies (aka identical corresponding statements)

Dynamic Slicing - Algorithm

1. Compute execution trace
2. Draw a directed graph with
   - Statements as nodes
   - Dependencies as edges
     - Data dependency (Definition-Use)
     - Control dependency (Test-Control)
     - Symmetric dependency
3. Mark all nodes m ($m \leq n$) in the graph
   – that redefine a variable $v_1 \in v$ and
   – there is no other node i ($m < i \leq n$) that redefines v_1

4. Trace back dependencies and mark visited nodes

Example ***

```
1.  begin
2.      read (n) ;
3.      i = 1 ;
4.      while (i <= n) do
5.          begin
6.              i f (i mod 2 == 0) then
7.                  x = 1 7 ;
8.              el se
9.                  x = 1 8 ;
10.             f i;
11.             i = i + 1 ;
12.         end
13.     od
14.     write ( x ) ;
15. end
```

Slicing Criterion:
({n=1}, 14^8, {x})

Types of slices – Input constraint

- A slice S can be obtained from a program P by deleting zero or more statements from P.
- S has the same values for the variables in V as P
  - any input I (**static slicing**)
  - for a given input I (**dynamic slicing**)

Types of slices – Input constraint

- Static slice
  - C = (i, V) with
    - a line number *i* and
    - a set of variables of interest *V*
  - Example: C = (5, {x})
- Dynamic slice
  - C = (x, I^q, V) with
    - program input x,
    - q^{th} element of the trajectory I^q
    - and a set of variables of interest V
  - Example: C = ({a = 4}, 5^3, {x})

Types of slices – Direction

Backward

- Computation of statements influencing the value of a variable

Forward

- Computation of statements influenced by a statement

```
begin
  r = 0;
  i = 0;
  while (i < x) do
    begin
      r = r + y;
      i = i + 1;
    end
  od
  ...
end
```

Open Problems

- Function calls
- Other control flow statements
  - goto
  - break
  - continue
- Composite data types
  - arrays
- Interprocess communication

Other Applications

- Change Impact Analysis
- Code optimization
  - Dead code removal
- Identify Duplicates in Source
  - To capture semantic differences between two programs
- Software Quality Assurance
  - Validate interactions between safety critical components
- Test Suite Reduction
  - Reduce cost of regression testing after modifications (only run those tests that are needed)

Tools

- Codesurfer
  - forward/backward slicing
- Unravel
  - backward slicing of C programs
  - Limited: e.g. no pointers to functions
- Indus/Kaveri
  - forward/ backward slicing (Java) programs
- ValSoft/Joana
  - program dependence graphs for C and Java

Questions?

Control of Study Objectives

```
begin
    z = 0;
    t = b;
    if (x > b) then
        begin
          x = x - b;
        end
    fi
    while (x <= t ) do
        begin
          t = t - x;
          z = t + 1;
        end
    od
    write(z);
end
```

1.) Compute the Dynamic Slice for the program for the following test case:

Test Input: x = 2, b = 1

Expected Output: z = 2

Control of Study Objectives

ET		Data Dep.	Control Dep.	Sym. Dep.	Slice
2^1	z = 0;				
3^2	t = b;				x
4^3	if (x > b)				x
6^4	x = x – b;				x
9^5	while (x <= t)				x
11^6	t = t – x;				x
12^7	z = t + 1;				x
9^8	while (x <= t)				x
15^9	write (z)				

Slice={3,4,6,9,11,12}