Program Analysis

Program Slicing (Part 4)

Prof. Dr. Michael Pradel

Software Lab, University of Stuttgart

Winter 2020/2021

Outline

1. **Introduction**
2. **Static Slicing**
3. **Thin Slicing**
4. **Dynamic Slicing** 

Mostly based on these papers:

- *Program Slicing*, Weiser., IEEE TSE, 1984
- *Thin Slicing*, Sridharan et al., PLDI 2007
- *Dynamic Program Slicing*, Agrawal and Horgan, PLDI 1990
- *A Survey of Program Slicing Techniques*, Tip, J Prog Lang 1995

Dynamic Slicing

- Various definitions

  Here: Agrawal & Horgan, PLDI 1990

- Dynamic slice: Statements of an execution that must be executed to give a variable a particular value
  - For an execution, i.e., a particular input
  - Slice for one input may be different from slice for another input

- Useful, e.g., for debugging: Get a reduced program that leads to the unexpected value

Dynamic Slice (Simple Approach)

- **Given: Execution history**
  - Sequence of PDG nodes that are executed
- **Slice for statement n and variable v:**
  - Keep PDG nodes only if there are in history
  - Use static slicing approach (= graph reachability) on reduced PDG

Example 1

```
var x = readInput();
if (x < 0) {
  y = x + 1;
  z = x + 2;
} else {
  if (x === 0) {
    y = x + 3;
    z = x + 4;
  } else {
    y = x + 5;
    z = x + 6;
  }
}
console.log(y);
console.log(z);
```

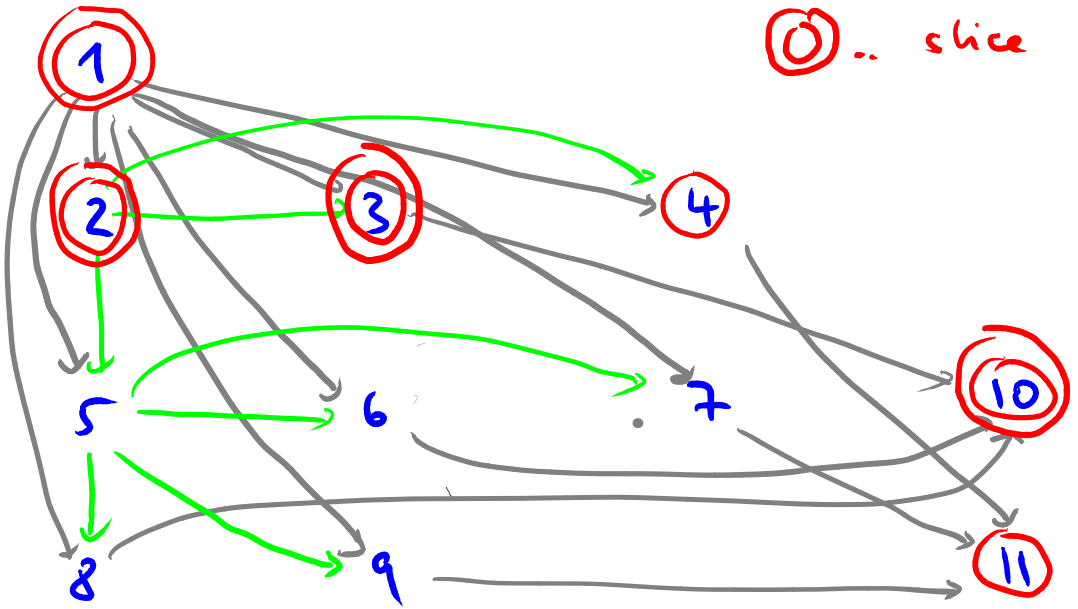
Example: Dynamic Slicing (Simple Approach)

```
1  var x = readInput();
2  if (x < 0) {
3    y = x + 1;
4    z = x + 2;
   } else {
5    if (x === 0) {
6      y = x + 3;
7      z = x + 4;
     } else {
8      y = x + 5;
9      z = x + 6;
     }
   }
10 console.log(y);
11 console.log(z);
```

Input: −1

History: 1, 2, 3, 4, 10, 11

PDG:



O .. executed

O .. slice (10, {y})

data dep. •→

control dep. →

Example 2: Quiz

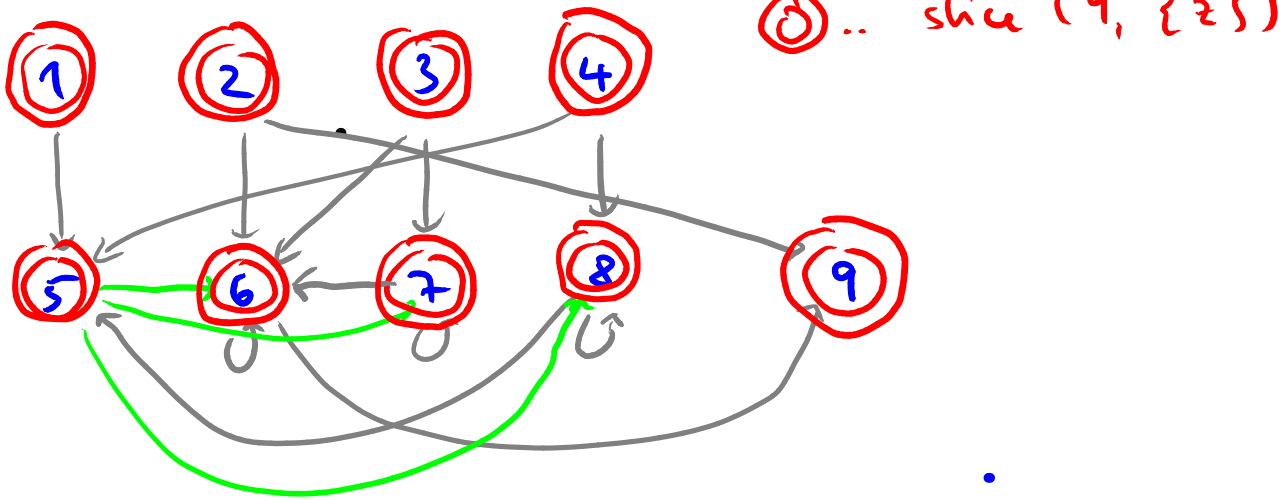
```
var n = readInput(); // 1
var z = 0;           // 2
var y = 0;           // 3
var i = 1;           // 4
while (i <= n) {     // 5
  z = z + y;         // 6
  y = y + 1;         // 7
  i = i + 1;         // 8
}
console.log(z);      // 9
```

Example 2 (Quiz)

Input: 1

History: 1, 2, 3, 4, 5, 6, 7, 8, 5, 9

```
1  var n = readInput();
2  var z = 0;
3  var y = 0;
4  var i = 1;
5  while (i <= n) {
6    z = z + y;
7    y = y + 1;
8    i = i + 1;
   }
9  console.log(z);
```



○ .. in history

○ .. slice (9, {z})

Limitations of Simple Approach

- Multiple occurrences of a single statement are represented as a single PDG node
- Difference occurrences of a statement may have different dependences
  - All occurrences get conflated
- Slices may be larger than necessary

Dynamic Slice (Revised Approach)

Dynamic dependence graph

- Nodes: Occurrences of nodes of static PDG
- Edges: Dynamic data and control flow dependences

Slice for statement n and variables V that are defined or used at n:

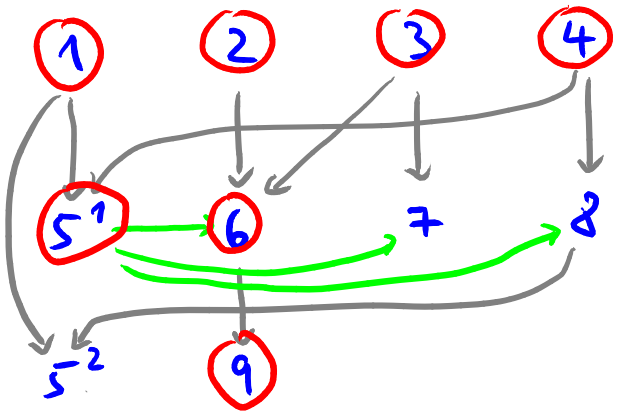
- Compute nodes S_{dyn} that can reach any of the nodes that represent occurrences of n
- Slice = statements with at least one node in S_{dyn}

Example 2 (revised approach)

Input: 1

History: 1, 2, 3, 4, 5, 6, 7, 8, 5, 9

```
1  var n = readInput();
2  var z = 0;
3  var y = 0;
4  var i = 1;
5  while (i <= n) {
6    z = z + y;
7    y = y + 1;
8    i = i + 1;
   }
9  console.log(z);
```



→ data

→ control

O .. slice(9, {z})

Discussion: Dynamic Slicing

- May yield a program that, if executed with another input, does not give the same value for the slicing criterion than the original program
- Instead: Focuses on isolating statements that affect a particular value
  - Useful, e.g., for debugging and program understanding
- Other approaches exist, see F. Tip's survey (1995) for an overview