Software Re-Engineering

Lecture: 15

Dr. Imran Ali
School of Computing- Software Engineering
FAST-National University of Computer & Emerging Sciences, Karachi

Sequence [Todays Agenda]

Content of Lecture

- Program Slicing

Program Slicing

- It is a technique that is given a potentially large program and it extracts an interesting subset of that program.
- Given a potentially large program slicing tries to extract an executable subset of this program that (potentially) affects the values that exist at a particular program location.

Program Slicing

- It is a software analysis technique that reduces a program to a smaller, more manageable version while preserving a specific behavior or computation.
- It identifies and isolates the code statements that directly contribute to a particular variable's value or a statement's execution at a specific point in the program.
- This technique helps developers focus on relevant code sections, understand program behavior, and debug more efficiently.

Program Slicing

- Slicing criterion = program location + variable
- So given something called a slicing criterion which is essentially a point in the program so the program location and a particular variable or maybe a particular memory location that you really care about.
- A specific point in the program (a statement and a variable) that the slice focuses on.
- Slicing wants to extract that subset of the program that is responsible for computing the value of that variable at this particular program location.

Program Slicing

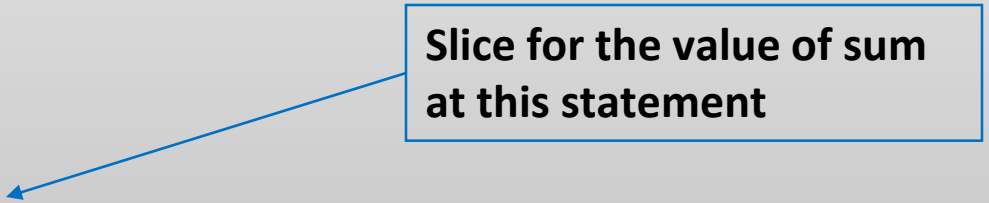
- **Example**

```
1.  var n = readInput();
2.  var I = 1;
3.  var sum = 0;
4.  var prod = 1;
5.  while (i<=n) {
6.  sum = sum + 1;
7.  prod = prod * i;
8.  i = i + 1;
9.  }
10. console.log(sum);
11. console.log(prod);
```

Program Slicing

- **Example**

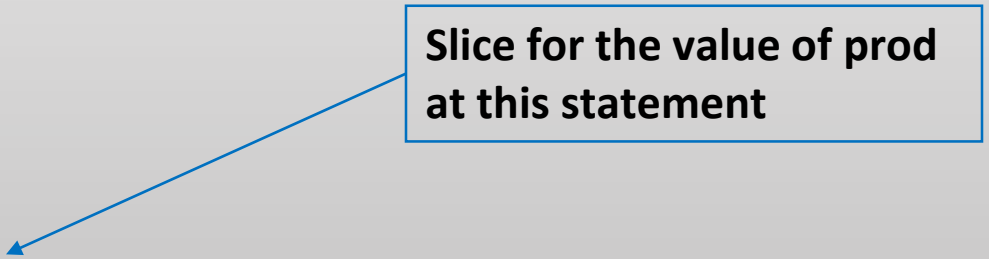
```
1.  var n = readInput();
2.  var I = 1;
3.  var sum = 0;
4.  var prod = 1;
5.  while (i<=n) {
6.  sum = sum + 1;
7.  prod = prod * i;
8.  i = i + 1;
9.  }
10. console.log(sum);
11. console.log(prod);
```



Program Slicing

- **Example**

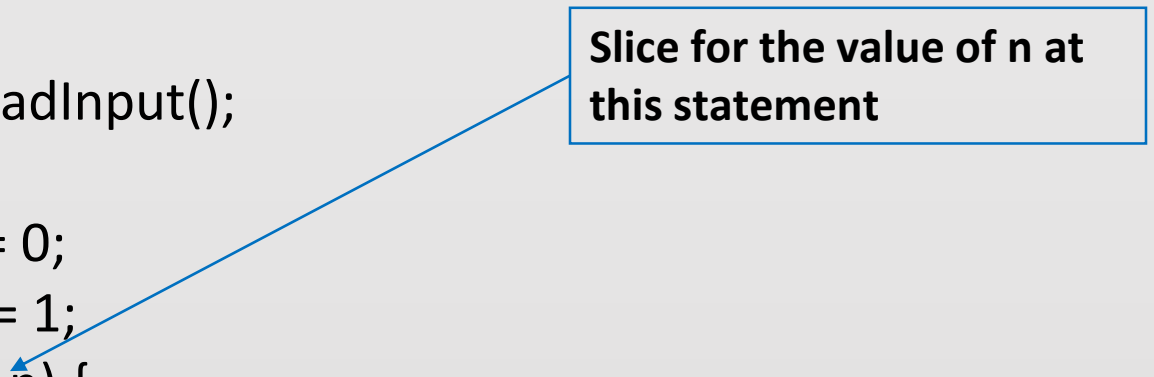
```
1.  var n = readInput();
2.  var I = 1;
3.  var sum = 0;
4.  var prod = 1;
5.  while (i<=n) {
6.  sum = sum + 1;
7.  prod = prod * i;
8.  i = i + 1;
9.  }
10. console.log(sum);
11. console.log(prod);
```



Program Slicing

- **Example**

```
1.  var n = readInput();
2.  var I = 1;
3.  var sum = 0;
4.  var prod = 1;
5.  while (i<=n) {
6.  sum = sum + 1;
7.  prod = prod * i;
8.  i = i + 1;
9.  }
10. console.log(sum);
11. console.log(prod);
```



Backward Slicing

- Statements that influencing the slicing criteria.
- Identifying statements that are affected by a given statement or variable.
- Starting from a slicing criterion we're interested in the statements that influence the slicing criterion.
- So it's basically about which other statements make us have this particular value at this particular location in the code.

Forward Slicing

- Statements that are influenced by the slicing criteria.
- Identifying statements that contribute to the value of a variable at a given point.
- Starting at a particular slicing criterion so a particular statement with a variable that has some value we are asking the question which other statements are influenced by this slicing criterion.

Forward Slicing

- Forward slicing identifies statements in a program that are affected by a particular statement.
- It helps understand the program's behavior by tracing the impact of a statement's execution on subsequent parts of the code.
- Example:

```
int sum = 0;
int i = 1;
while (i < 11) {
sum = sum + i;
i = i + 1;
}
System.out.println(sum);
```

- The forward slice would include the statements:
- “sum = sum + i;” (The statement itself)
- “i = i + 1;” (The statement that modifies the loop variable, which affects next iteration of the loop)
- “System.out.println(sum);” (The statement that prints the final value of sum, which is dependent on the loop)

Static Slicing

- Static slicing is a technique that does not execute a program but rather analyze its source code in order to extract a slice of a program.
- Static slicing involves analyzing a program's source code to identify a subset of statements that contribute to the value of a variable at a specific point.

Static Slicing

- Imagine a program that calculates both the sum and product of numbers from 1 to n.
- Let's say you want to understand how the variable product is calculated.
- Static slicing would identify all the statements that directly or indirectly affect the value of product, regardless of which execution path is taken.

Static Slicing

- Example:

```
#include <iostream>
int main() {
int n, i, sum = 0, product = 0; // Initialize product to 0
std::cout << "Enter a number: ";
std::cin >> n;
for (i = 1; i <= n; ++i) {
sum += i;
product *= i; // Incorrect: will always be 0 if product is initialized to 0
}
std::cout << "Sum: " << sum << std::endl;
std::cout << "Product: " << product << std::endl;
return 0;
}
```

- If you were to statically slice this code with the product variable at the point where it's printed, the slice would likely include the following statements:
- product = 0; (initialization)
- product *= i; (inside the loop)

Dynamic Slicing

- Dynamic slicing is a technique that execute a program and then from this execution extract an Interesting fragment of the larger program.
- Dynamic slicing extracts a subset of a program's statements that actually affect a variable's value at a specific point during a particular program execution.
- It uses the program's execution history (the sequence of statements executed) to identify the relevant statements, unlike static slicing which considers all potentially affecting statements.

Program Slicing Applications

- Debugging:
- Focus on parts of program relevant for a bug.
- If you are interested in a particular misbehavior of your program i.e, some bug, then slicing helps you to identify which parts of the program influence what you are seeing.
- You know that at a particular point in the program some variable has the wrong value then the question you are typically asking is which parts of the program influence the value of that variable that I'm seeing here and slicing can help you to answer that question by basically focusing your attention on those parts of the program that are relevant for this bug.

Program Slicing Applications

- Program understanding:
- Which statements influence this statement?
- You may just want to know which statements are actually influencing some other statements and the values that they are computed on and slicing can help you by doing.
- This you can start from one statement and then either slice backward or forward and see how this statement gets influenced or influences future statements.

Program Slicing Applications

- Change impact analysis:
- Which parts of a program are affected by a change? What should be retested?
- When a program is evolving then the code is changing then at some point you want to know which parts of the code are affected by a particular change.
- For example this is interesting to know if you want to decide which parts of a program should be retested or may be tested at all after a code change and because typically you do not have to re-test everything after you change something but only those parts of the program that are affected by a change and which parts of a program are affected by a change that's something you can compute with the help of slicing.

Program Slicing Applications

- Parallelization:
- Determine parts of program that can be computed independently of each other.
- If you have a program that is sequential but you would like to run it in parallel to speed up the execution then you need some way to determine which parts of the program can be computed independently of each other and because slicing tells you about the dependencies of different statements and parts of your program it can also help you to basically compute individual slices that are independent of each other and then you can run these slices in parallel because they anyway do not influence each other.


Thank You!