Static Program Analysis Part IV

Program Slicing (Weiser ‘82)

- A *program slice* consists of the parts of a program that (potentially) affect the values computed at some point of interest, referred to as a *slicing criterion*
- Typically, a slicing criterion consists of a pair (line-number; variable).
- The parts of a program which have a direct or indirect effect on the values computed at a slicing criterion C are called the program slice with respect to criterion C
- A program slice is computed from the program dependency graph
- The task of computing program slices is called program slicing

Program Slicing Research

Types of slices

- Backward static slice
- Executable slice
- Forward static slice
- Dynamic slice
- Execution slice

Levels of slices

- Intra-procedural
- Inter-procedural

1. Agrawal
2. Binkley
3. Gallagher
4. Gupta
5. Horgan
6. Horwitz
7. Korel
8. Laski
9. K. Ottenstein
10. L. Ottenstein
11. Reps
12. Soffa
13. Tip
14. Weiser

Static Backward Slicing

- A *backward slice* of a program with respect to a program point **p** and set of program variables **V** consists of all statements and predicates in the program that may affect the value of variables in **V** at **p**

- The program point **p** and the variables **V** together form the *slicing criterion*, usually written <**p, V**>

Static Backward Slicing - Example

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

Criterion <9, product>

Static Backward Slicing - Example

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

Criterion <9, product>

Executable Slicing

- A slice is *executable* if the statements in the slice form a syntactically correct program that can be executed.

- If the slice is computed correctly (safely), the results of running the program that is the executable slice produces the same result for variables in **V** at **p** for all inputs.

Executable Slicing - Example

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

Criterion <9, product>

```
1.  read (n)
2.  i := 1
3.
4.  product := 1
5.  while i <= n do
6.
7.      product := product * i
8.      i := i + 1
9.
10. write (product)
```

Static Forward Slicing

- A *forward slice* of a program with respect to a program point **p** and set of program variables **V** consists of all statements and predicates in the program that may be affected by the value of variables in **V** at **p**

- The program point **p** and the variables **V** together form the *slicing criterion*, usually written <**p, V**>

Static Forward Slicing - Example

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

Criterion <3, sum>

Static Forward Slicing - Example

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

Criterion <3, sum>

Static Forward Slicing - Example

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

Criterion <1, n>

Static Forward Slicing - Example

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

Criterion <1, n>

Dynamic Slicing

- A *dynamic slice* of a program with respect to an input value of a variable **v** at a program point **p** for a particular execution **e** of the program is the set of all statements in the program that affect the value of **v** at **p.**

- The program point **p,** the variables **V,** and the input i for **e** form the *slicing criterion*, usually written <i**, v, p**>. The slicing uses the execution history or trajectory for the program with input **i.**

Dynamic Slicing - Example

```
1.  read (n)
2.  for I := 1 to n do
3.     a := 2
4.      if c1 then
5.          if c2 then
6.              a := 4
7.          else
8.              a := 6
9.          z := a
10.   write (z)
```

- Input n is 1; c1, c2 both true
- Execution history is

  $1^1, 2^1, 3^1, 4^1, 5^1, 6^1, 9^1, 2^2, 10^1$

- Criterion<1, 10^1, z>

Dynamic Slicing - Example

```
1.  read (n)
2.  for I := 1 to n do
3.      a := 2
4.       if c1 then
5.            if c2 then
6.                a := 4
7.            else
8.                a := 6
9.            z := a
10.   write (z)
```

- Input n is 1; c1, c2 both true
- Execution history is

  1^1, 2^1, 3^1, 4^1, 5^1, 6^1, 9^1, 2^2, 10^1

- Criterion<1, 10^1, z>

Dynamic Slicing - Example

```
1.  read (n)
2.  for I := 1 to n do
3.      a := 2
4.      if c1 then
5.          if c2 then
6.              a := 4
7.          else
8.              a := 6
9.          z := a
10.   write (z)
```

```
1.  read (n)
2.  for I := 1 to n do
3.      a := 2
4.      if c1 then
5.          if c2 then
6.              a := 4
7.          else
8.              a := 6
9.          z := a
10.   write (z)
```

Static slice <10, z>

Dynamic Slicing - Example

```
1.  read (n)
2.  for I := 1 to n do
3.     a := 2
4.      if c1 then
5.          if c2 then
6.              a := 4
7.          else
8.              a := 6
9.          z := a
10.   write (z)
```

- Input n is 2; c1, c2 false on first iteration and true on second iteration
- Execution history is

  $1^1, 2^1, 3^1, 4^1, 9^1, 2^2, 3^2, 4^2, 5^1, 6^1, 9^2, 2^3, 10^1>$

- Criterion<1, 10^1, z>

Dynamic Slicing - Example

```
1.  read (n)
2.  for I := 1 to n do
3.      a := 2
4.       if c1 then
5.           if c2 then
6.               a := 4
7.           else
8.               a := 6
9.           z := a
10.   write (z)
```

- Input n is 2; c1, c2 false on first iteration and true on second iteration
- Execution history is

  1^1, 2^1, 3^1, 4^1, 9^1, 2^2, 3^2, 4^2, 5^1, 6^1, 9^2, 2^3, 10^1>

- Criterion<1, 10^1, z>

Dynamic Slicing - Example

```
1.  read (n)
2.  for I := 1 to n do
3.      a := 2
4.      if c1 then
5.          if c2 then
6.              a := 4
7.          else
8.              a := 6
9.          z := a
10.   write (z)
```

```
1.  read (n)
2.  for I := 1 to n do
3.      a := 2
4.      if c1 then
5.          if c2 then
6.              a := 4
7.          else
8.              a := 6
9.          z := a
10.   write (z)
```

Static slice <10, z>

Execution Slicing

- An *execution slice* of a program with respect to an input value of a variable **v** is the set of statements in the program that are executed with input **v.**

Execution Slicing - Example

```
1.  read (n)
2.  for I := 1 to n do
3.     a := 2
4.      if c1 then
5.          if c2 then
6.              a := 4
7.          else
8.              a := 6
9.          z := a
10.   write (z)
```

- Input n is 2; c1, c2 false on first iteration and true on second iteration
- Execution history is

  $1^1, 2^1, 3^1, 4^1, 9^1, 2^2, 3^2, 4^2, 5^1, 6^1, 9^2, 2^3, 10^1>$

- Execution slice is

  1, 2, 3, 4, 5, 6, 9, 10

Execution Slicing - Example

```
1.  read (n)
2.  for I := 1 to n do
3.     a := 2
4.      if c1 then
5.          if c2 then
6.              a := 4
7.          else
8.              a := 6
9.          z := a
10.   write (z)
```

- Input n is 2; c1, c2 false on first iteration and true on second iteration
- Execution history is

  $1^1, 2^1, 3^1, 4^1, 9^1, 2^2, 3^2, 4^2, 5^1, 6^1, 9^2, 2^3, 10^1>$

- Execution slice is

  1, 2, 3, 4, 5, 6, 9, 10