

Slicing

Rupesh Nasre.

CS6843 Program Analysis
IIT Madras
Jan 2015

Applications

- Program understanding / debugging
- Program restructuring
- Program differencing
- Test coverage
- Model checking

4

Outline

- Introduction and applications
- Application time
 - static
 - dynamic
- Direction
 - forward
 - backward

2

Program with Multiple Functionality

```
Line + Character counting
extern void scanLine(FILE *f, bool *eof, int *nread);
void lineCharCount(FILE *f) {
    int nlines = 0;
    int nchars = 0;
    int nread;
    bool eof = false;

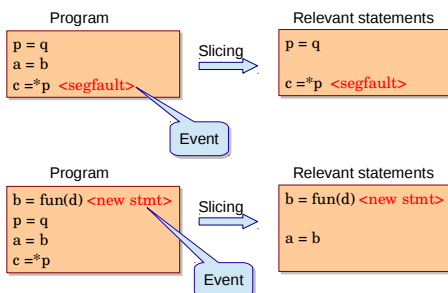
    do {
        scanLine(f, &eof, &nread);
        ++nlines;
        nchars += nread;
    } while (!eof);

    printf("nlines = %d\n", nlines);
    printf("nchars = %d\n", nchars);
}
```

5

Definition

- A slice is a subset of program statements (possibly) relevant to an event of interest.



3

Program with Single Functionality

```
Line + Character counting
extern void scanLine(FILE *f, bool *eof,
                    int *nread);
void lineCharCount(FILE *f) {
    int nlines = 0;
    int nchars = 0;
    int nread;
    bool eof = false;

    do {
        scanLine(f, &eof, &nread);
        ++nlines;
        nchars += nread;
    } while (!eof);

    printf("nlines = %d\n", nlines);
    printf("nchars = %d\n", nchars);
}
```

```
Line + Character counting
extern void scanLine(FILE *f, bool *eof,
                    int *nread);
void lineCharCount(FILE *f) {
    int nlines = 0;
    int nchars = 0;
    int nread;
    bool eof = false;

    do {
        scanLine(f, &eof, &nread);
        ++nlines;
        nchars += nread;
    } while (!eof);

    printf("nlines = %d\n", nlines);
    printf("nchars = %d\n", nchars);
}
```

6

Program with Single Functionality

Line + Character counting

```
extern void scanLine2(FILE *f, bool *eof,
                    int *nread);
void lineCharCount(FILE *f) {
    int nlines = 0;
    int nchars = 0;
    int nread;
    bool eof = false;

    do {
        scanLine2(f, &eof, &nread);
        ++nlines;
        nchars += nread;
    } while (!eof);

    printf("nlines = %d\n", nlines);
    printf("nchars = %d\n", nchars);
}
```

Line + Character counting

```
extern void scanLine(FILE *f, bool *eof,
                    int *nread);
void lineCharCount(FILE *f) {
    int nlines = 0;
    int nchars = 0;
    int nread;
    bool eof = false;

    do {
        scanLine(f, &eof, &nread);
        ++nlines;
        nchars += nread;
    } while (!eof);

    printf("nlines = %d\n", nlines);
    printf("nchars = %d\n", nchars);
}
```

7

Forward Slice

```
void main() {
    int sum = 0;
    int i = 0;

    while (i < 11) {
        sum += i;
        ++i;
    }
    printf("sum = %d\n", sum);
    printf("i = %d\n", i);
}
```

Slicing criteria

10

Slicing Types

- **Forward**
 - The forward slice at program point p is the program subset that may be affected by p
- **Backward**
 - The backward slice at program point p is the program subset that may affect p
- **Chop**
 - The chop between program points p and q is the program subset that may be affected by p and that may affect q

8

Backward Slice

```
void main() {
    int sum = 0;
    int i = 0;

    while (i < 11) {
        sum += i;
        ++i;
    }
    printf("sum = %d\n", sum);
    printf("i = %d\n", i);
}
```

Slicing criteria

11

Forward Slice

```
void main() {
    int sum = 0;
    int i = 0;

    while (i < 11) {
        sum += i;
        ++i;
    }
    printf("sum = %d\n", sum);
    printf("i = %d\n", i);
}
```

Slicing criteria

9

Backward Slice

```
void main() {
    int sum = 0;
    int i = 0;

    while (i < 11) {
        sum += i;
        ++i;
    }
    printf("sum = %d\n", sum);
    printf("i = %d\n", i);
}
```

Slicing criteria

12

Chop

```
void main() {
  int sum = 0;
  int i = 0;

  while (i < 11) {
    sum += i;
    ++i;
  }
  printf("sum = %d\n", sum);
  printf("i = %d\n", i);
}
```

Slicing / chopping criteria

Slicing / chopping criteria

13

Slicing Types

- **Static**
 - Compile time, without making assumptions about the program inputs
- **Dynamic**
 - Execution time or compile time, relying on specific test cases

Language features such as functions, unstructured control flow, composite data types, pointers, and concurrency require specific extensions to the slicing algorithms.

16

Chop

```
void main() {
  int sum = 0;
  int i = 0;

  while (i < 11) {
    sum += i;
    ++i;
  }
  printf("sum = %d\n", sum);
  printf("i = %d\n", i);
}
```

Slicing / chopping criteria

Slicing / chopping criteria

START

STOP

14

Static Slicing

- Whatever we have been looking at so far...
- There could be multiple valid slices for a given criteria, depending upon analysis precision.
- There exists at least one slice for a criteria (the program itself).
- The analysis implicitly assumes that the program halts.
- Finding a statement-minimal slice is impossible (halting problem).

17

Chop

```
void main() {
  int sum = 0;
  int i = 0;

  while (i < 11) {
    sum += i;
    ++i;
  }
  printf("sum = %d\n", sum);
  printf("i = %d\n", i);
}
```

Slicing / chopping criteria

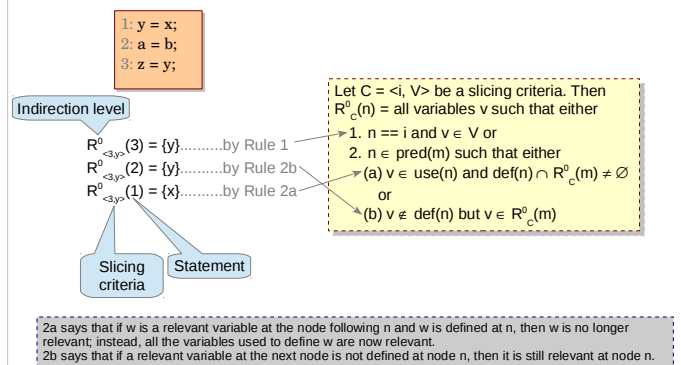
Slicing / chopping criteria

START

STOP

15

Computing Relevant Variables



Computing Relevant Statements

- The relevant statements define relevant variables.

S_c^0 = all statements n such that $\text{def}(n) \cap R_c^0(n+1) \neq \emptyset$.

19

Slicing as a DFA

```
1: a = 1;
2: while (f(k)) {
3:   if (g(c)) {
4:     b = a;
5:     x = 2;
6:   } else {
7:     c = b;
8:     y = 3;
9:   }
10: k = k + 1;
11: z = x + y;
```

Slicing criterion = $\langle 10, z \rangle$

Is statement 1 included in the slice?

22

But what about Control Dependence?

- Let's define a set of statements that influence a set of statements
- INFL(b) is the set of statements which directly affect execution of statement b .

$B_c^0 = \bigcup \text{INFL}(n)$ such that $n \in S_c^0$

Branch statements with indirect relevance to a slice

To include **all** the indirect influences, the statements with direct influence on B_c^0 must now be considered, and then the branch statements influencing those new statements, and so on.

20

Slicing as a DFA may not be minimal

```
1: a = 1;
2: while (f(k)) {
3:   if (g(c)) {
4:     b = a;
5:     x = 2;
6:   } else {
7:     c = b;
8:     y = 3;
9:   }
10: k = k + 1;
11: z = x + y;
```

Slicing criterion = $\langle 10, z \rangle$

Is statement 1 included in the slice?

Statements 1 and 4 would not be part of the slice as there is a guarantee that after if block is executed, else block will definitely be not executed. Therefore, there is no dependence from statement 4 to statement 7.

23

Computing Static Slice

Relevance at level n is defined as below

Next set of relevant variables = current set of relevant variables $\cup$ relevant variables in current set of relevant branch statements

for all $i \geq 0$

$R_c^{i+1}(n) = R_c^i(n) \cup R_{BC(b)}^0(n)$ for $b \in B_c^i$
where $BC(b)$ is a branch statement criterion defined as $\langle b, \text{use}(b) \rangle$

$B_c^{i+1} = \bigcup \text{INFL}(n)$ for $n \in S_c^{i+1}$

$S_c^{i+1} =$ all statements n such that $\text{def}(n) \cap R_c^{i+1}(n+1) \neq \emptyset$ or $n \in B_c^i$

21

Dynamic Slicing

Slicing criteria
(9, x, n=2)

```
1: read(n);
2: i = 1;
3: while (i <= n) {
4:   if (i % 2 == 0)
5:     x = 7;
6:   else
7:     x = 16;
8:   ++i;
9: write(x);
```

```
1: read(n);
2: i = 1;
3: while (i <= n) {
4:   if (i % 2 == 0)
5:     x = 7;
6:   else
7:     ;
8:   ++i;
9: write(x);
```

The else branch may be omitted from the dynamic slice because the assignment of 16 to x in the first iteration is killed by the assignment of 7 in the second iteration.

24

Static vs. Dynamic Slicing

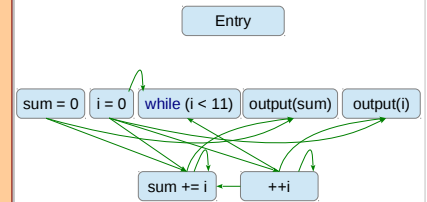
- | | |
|--|---|
| • Across all inputs | • Specific set of inputs |
| • Less precise | • More precise |
| • Source code | • Source code or trace |
| • Slicing criteria contains statement and variables. | • Slicing criteria contains statement instance, variables and inputs. |

25

Data Dependence Graph

```
void main() {
    int sum = 0;
    int i = 0;

    while (i < 11) {
        sum += i;
        ++i;
    }
    printf("sum = %d\n", sum);
    printf("i = %d\n", i);
}
```



→ Data dependence

28

Computing Slices

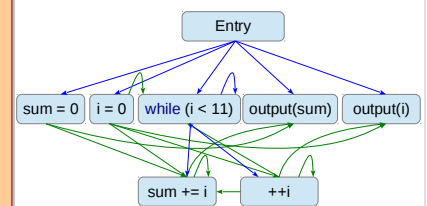
- Reachability in a dependence graph
 - PDG (intra-procedural)
 - SDG (inter-procedural)
- Dependence graph
 - Control dependence (is it the same as a CFG?)
 - Data dependence

26

Program Dependence Graph

```
void main() {
    int sum = 0;
    int i = 0;

    while (i < 11) {
        sum += i;
        ++i;
    }
    printf("sum = %d\n", sum);
    printf("i = %d\n", i);
}
```



→ Control dependence

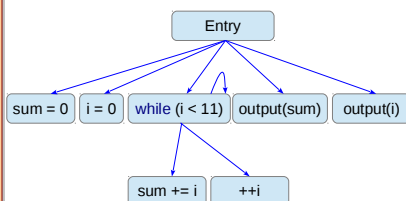
→ Data dependence

29

Control Dependence Graph

```
void main() {
    int sum = 0;
    int i = 0;

    while (i < 11) {
        sum += i;
        ++i;
    }
    printf("sum = %d\n", sum);
    printf("i = %d\n", i);
}
```



→ Control dependence

27

Syntactically Valid Slices

- Include additional statements or reorder statements to make the slice syntactically valid.
- Operates at the source code level not IR.

```
read(n);
i = 1;
if (p == null)
    x = x + 1;
else
    n = n * 2;
write(n);
```

Slicing →

```
read(n);
if (p == null)
    else
        n = n * 2;
write(n);
```

Syntax error

Situation is simpler in C due to ; as statement terminator.

30

Acknowledgments

- Christopher Lewis (upenn)
- Laurie Hendren (mcgill)
- Frank Tip (waterloo)