

Program Slicing

Last time

- Interprocedural pointer analysis

Today

- Program slicing [Weiser 84]
- Uses of program slicing

Thanks to Grammatech and Tim Teitelbaum for content that I've borrowed

Program Slicing

Backward slice

- The **backward slice** at program point p is the program subset that may affect p

Forward slice

- The **forward slice** at program point p is the program subset that may be affected by 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

Backward Slice

Example

```
int main()
{
    int sum = 0;
    int i = 1;
    while (i<11) {
        sum = sum+i;
        i = i + 1;
    }
    printf("%d \n", sum);
    printf("d \n", i);
}
```

The program subset that may affect `printf("d \n", i);`

Forward Slice

Example

```
int main()
{
    int sum = 0;
    int i = 1;
    while (i<11) {
        sum = sum+i;
        i = i + 1;
    }
    printf("%d \n", sum);
    printf("d \n", i);
}
```

The program subset that may be affected by `sum = 0;`

Chop

Example

```
int main()
{
  → int sum = 0;
    int i = 1;
    while (i<11) {
      sum = sum+i;
      i = i + 1;
    }
  → printf("%d \n", sum);
    printf("d \n", i);
}
```

The chop is empty
There is no data flow between the two statements

The program subset that may be affected by `sum = 0;` and that may affect `printf("%d \n", sum);`

Uses of Program Slicing

Program understanding

- What is affected by what?

Program restructuring

- Isolate functionally distinct pieces of code

Program specialization and reuse

- Use slices to represent specialized pieces of code
- Only reuse relevant slices

Program differencing

- Compare slices to identify program changes

Uses of Program Slicing

Test coverage

- What new test cases would improve code coverage?
- What regression tests should be run after a change?

Model checking

- Reduce state space by removing irrelevant parts of the program

Automatic differentiation

- Activity analysis– what variables contribute to the derivative of a function?

Specialization Example

Given

- A line-and-character-count program

Produce

- A line-count program
- A character-count program

Line-and-Character-Count Program

```
void line_char_count (FILE *f)
{
    int lines = 0;
    int chars;
    BOOL eof_flag = FALSE;
    int n;
    extern void scan_line(FILE *f, BOOL *bptr, int, *iptr);
    scan_line(f, &eof_flag, &n);
    chars = n;
    while (eof_flag == FALSE) {
        lines = lines + 1;
        scan_line(f, &eof_flag, &n);
        chars = chars + n;
    }
    printf("lines = %d \n", lines);
    printf("char s= d \n", chars);
}
```

CIS 570 Lecture 14

Program Slicing

10

Character-Count Program

```
void line_char_count (FILE *f)
{
    int lines = 0;
    int chars;
    BOOL eof_flag = FALSE;
    int n;
    extern void scan_line(FILE *f, BOOL *bptr, int, *iptr);
    scan_line(f, &eof_flag, &n);
    chars = n;
    while (eof_flag == FALSE) {
        lines = lines + 1;
        scan_line(f, &eof_flag, &n);
        chars = chars + n;
    }
    printf("lines = %d \n", lines);
    → printf("chars = d \n", chars);
}
```

CIS 570 Lecture 14

Program Slicing

11

Line-Count Program

```
void line_char_count (FILE *f)
{
    int lines = 0;
    int chars;
    BOOL eof_flag = FALSE;
    int n;
    extern void scan_line(FILE *f, BOOL *bptr, int, *iptr);
    scan_line(f, &eof_flag, &n);
    chars = n;
    while (eof_flag == FALSE) {
        lines = lines + 1;
        scan_line(f, &eof_flag, &n);
        chars = chars + n;
    }
    printf("lines = %d \n", lines);
    printf("chars = d \n", chars);
}
```

CIS 570 Lecture 14

Program Slicing

12

Line-Count Program

```
void line_char_count (FILE *f)
{
    int lines = 0;

    BOOL eof_flag = FALSE;

    extern void scan_line2(FILE *f, BOOL *bptr, int);
    scan_line2(f, &eof_flag);

    while (eof_flag == FALSE) {
        lines = lines + 1;
        scan_line2(f, &eof_flag);
    }
    printf("lines = %d \n", lines);
}
```

CIS 570 Lecture 14

Program Slicing

13

How Do We Compute Slices?

Reachability in a dependence graph

Program Dependence Graph (PDG)

- Represents dependences within one procedure
- Intraprocedural slicing is reachability in one PDG

System Dependence Graph (SDG)

- Represents dependences within entire system
- Interprocedural slicing is reachability in the SDG

Intraprocedural Slicing

Program Dependence Graph (PDG)

- Nodes are statements
- Edges represent either:
 - Control dependence
 - Data dependence

Backward slice

- To compute a backward slice from point p, compute backward reachability in the PDG from node p

Forward slice

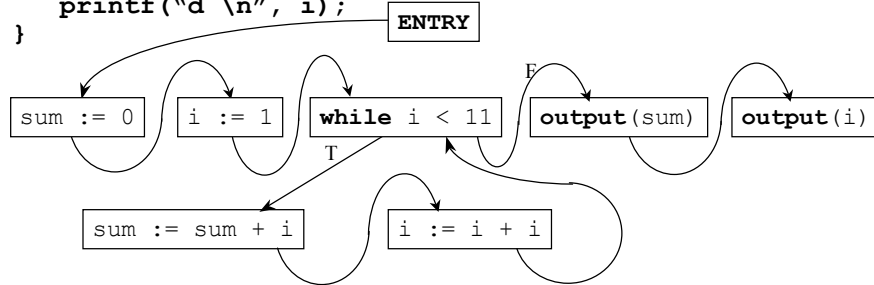
- To compute a forward slice from point p, compute forward reachability in the PDG from node p

Chop

- To compute the chop between points p and q, identify all paths between p and q

Control Flow Graph

```
int main(){
  int sum = 0;
  int i = 1;
  while (i<11) {
    sum = sum+i;
    i = i + 1;
  }
  printf("%d \n", sum);
  printf("%d \n", i);
}
```



CIS 570 Lecture 14

Program Slicing

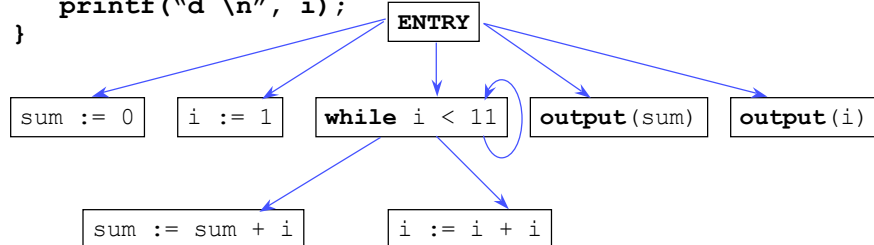
16

Control Dependence Graph

```
int main(){
  int sum = 0;
  int i = 1;
  while (i<11) {
    sum = sum+i;
    i = i + 1;
  }
  printf("%d \n", sum);
  printf("%d \n", i);
}
```

Control dependence

$p \rightarrow q$ if p branches one way, q eventually reached; p branches other way, q may not be reached



CIS 570 Lecture 14

Program Slicing

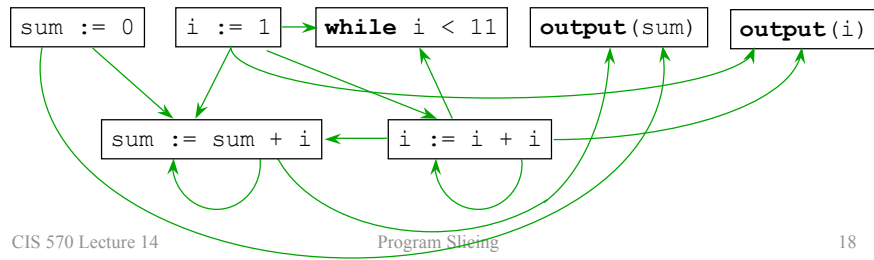
17

Flow Dependence Graph

```
int main(){
  int sum = 0;
  int i = 1;
  while (i<11) {
    sum = sum+i;
    i = i + 1;
  }
  printf("%d \n", sum);
  printf("%d \n", i);
}
```

Data dependence

$p \rightarrow q$ Value of variable assigned at p may be used at q .



CIS 570 Lecture 14

Program Slicing

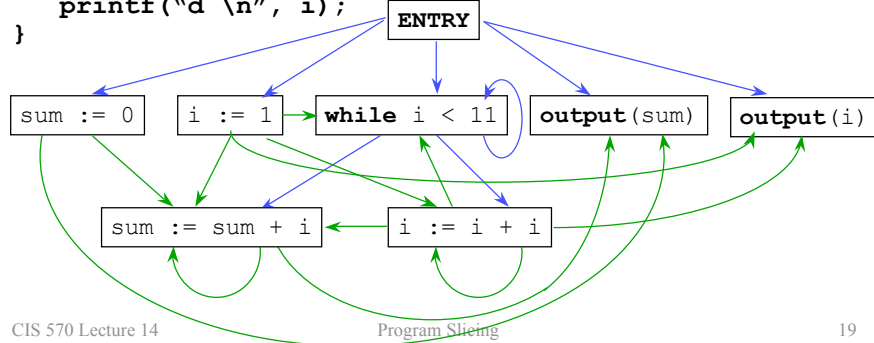
18

Program Dependence Graph

```
int main(){
  int sum = 0;
  int i = 1;
  while (i<11) {
    sum = sum+i;
    i = i + 1;
  }
  printf("%d \n", sum);
  printf("%d \n", i);
}
```

Control dependence

Data dependence



CIS 570 Lecture 14

Program Slicing

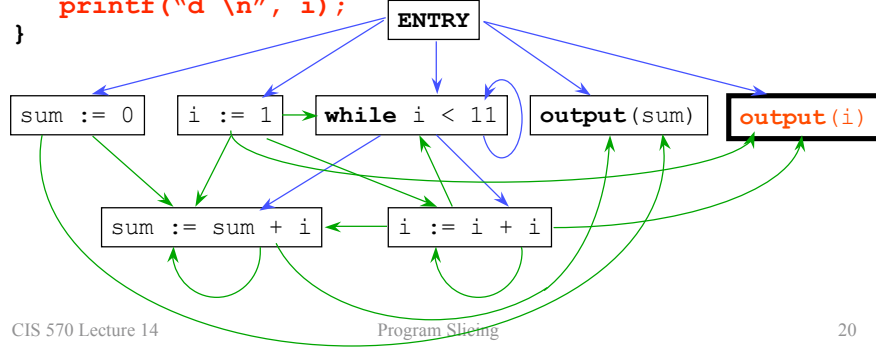
19

Backward Slice

```

int main(){
  int sum = 0;
  int i = 1;
  while (i<11) {
    sum = sum+i;
    i = i + 1;
  }
  printf("%d \n", sum);
  printf("%d \n", i);
}

```



CIS 570 Lecture 14

Program Slicing

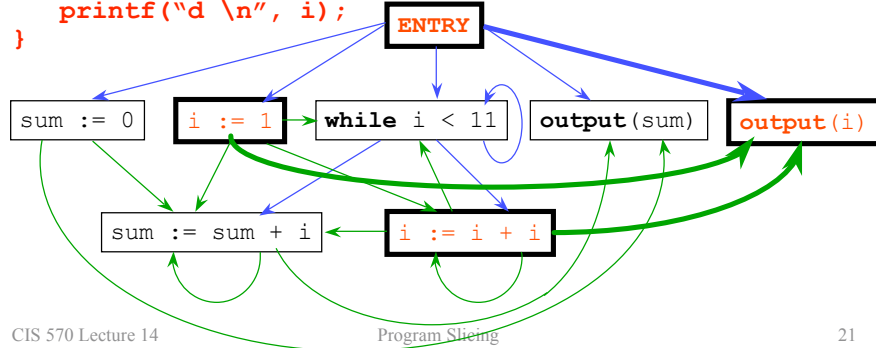
20

Backward Slice

```

int main(){
  int sum = 0;
  int i = 1;
  while (i<11) {
    sum = sum+i;
    i = i + 1;
  }
  printf("%d \n", sum);
  printf("%d \n", i);
}

```



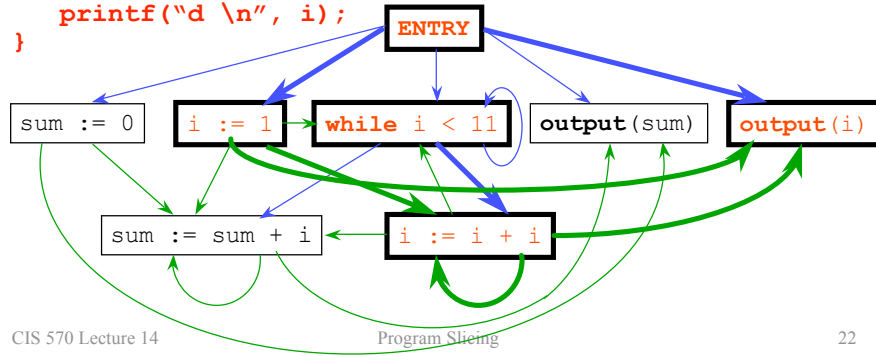
CIS 570 Lecture 14

Program Slicing

21

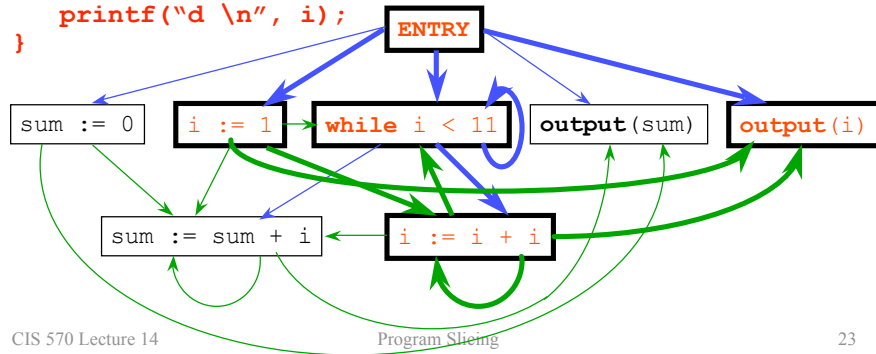
Backward Slice

```
int main(){
  int sum = 0;
  int i = 1;
  while (i<11) {
    sum = sum+i;
    i = i + 1;
  }
  printf("%d \n", sum);
  printf("%d \n", i);
}
```



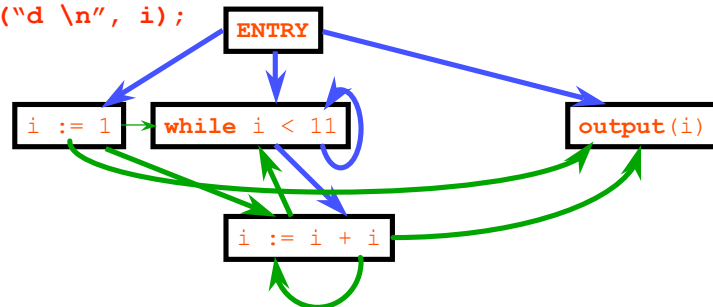
Backward Slice

```
int main(){
  int sum = 0;
  int i = 1;
  while (i<11) {
    sum = sum+i;
    i = i + 1;
  }
  printf("%d \n", sum);
  printf("%d \n", i);
}
```



Slice Extraction

```
int main(){  
    int i = 1;  
    while (i<11) {  
        i = i + 1;  
    }  
    printf("\d \n", i);  
}
```



Interprocedural Slice

```
int main(){  
    int sum = 0;  
    int i = 1;  
    while (i<11) {  
        add(sum,i);  
        add(i, 1);  
    }  
    printf("%d \n", sum);  
    printf("\d \n", i);  
}
```

```
int add(int x, y){  
    return x + y;  
}
```

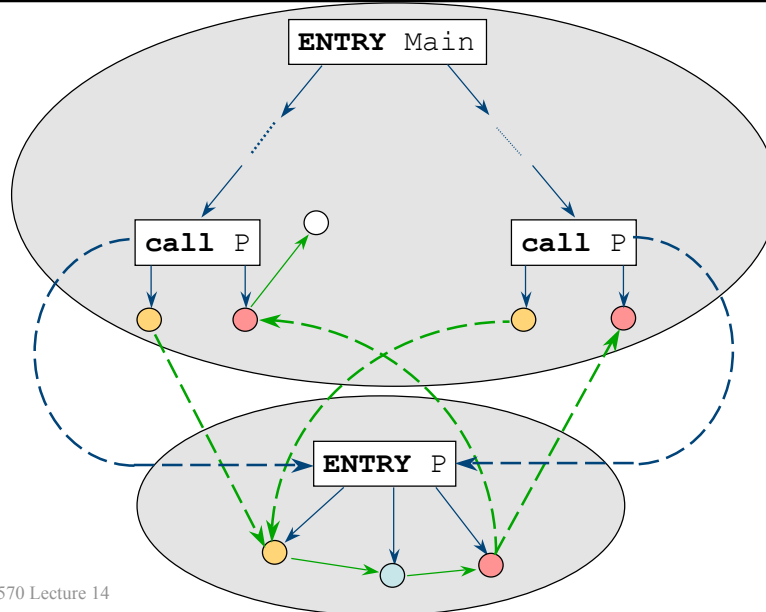
Should we include `add(sum, i)`?

Interprocedural Slicing

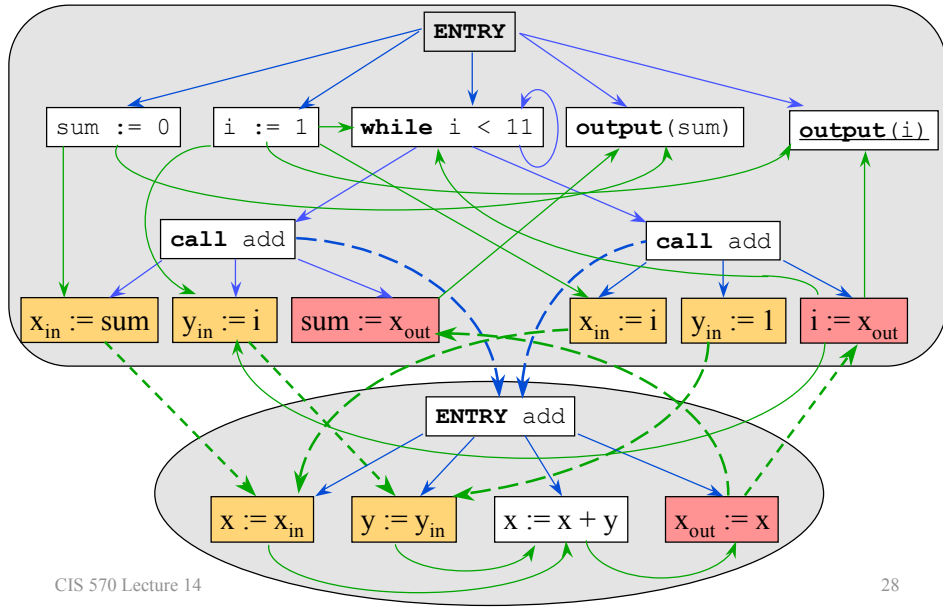
System Dependence Graph (SDG)

- One PDG for each procedure
- Additional edges
 - Connect calls to entries
 - Connect actual parameters to formal parameters
 - Connect procedure results to call-site return values

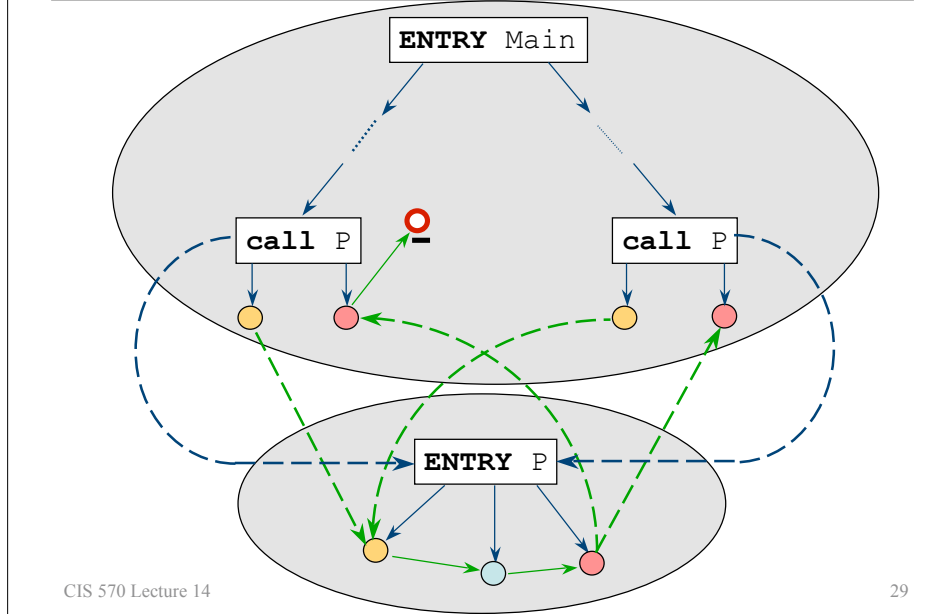
System Dependency Graph (SDG)



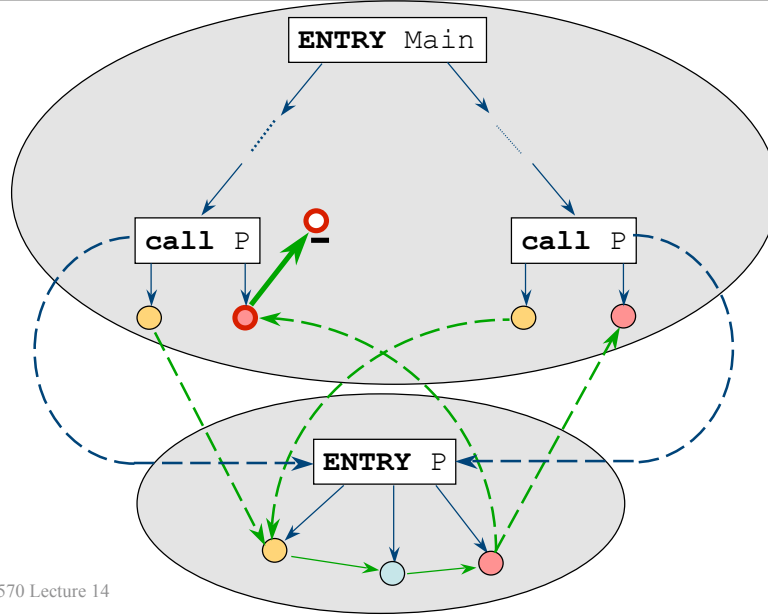
Example SDG



Interprocedural Backward Slice



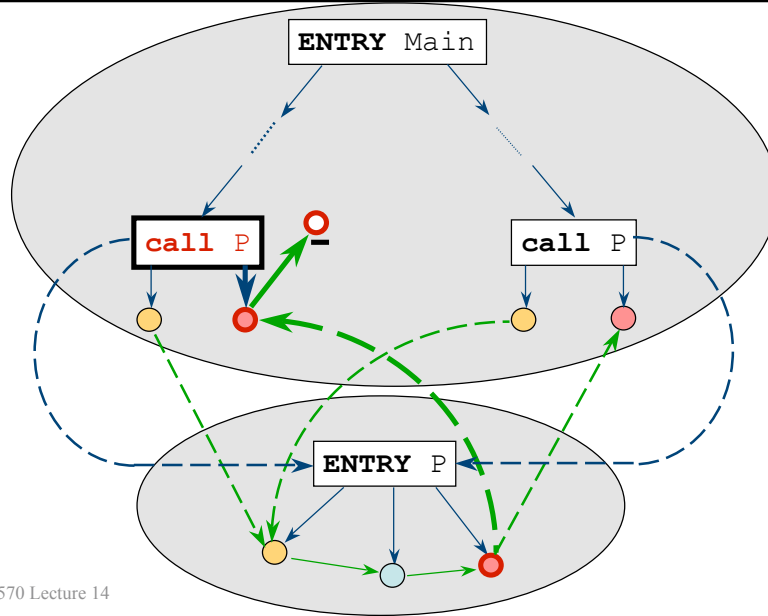
Interprocedural Backward Slice



CIS 570 Lecture 14

30

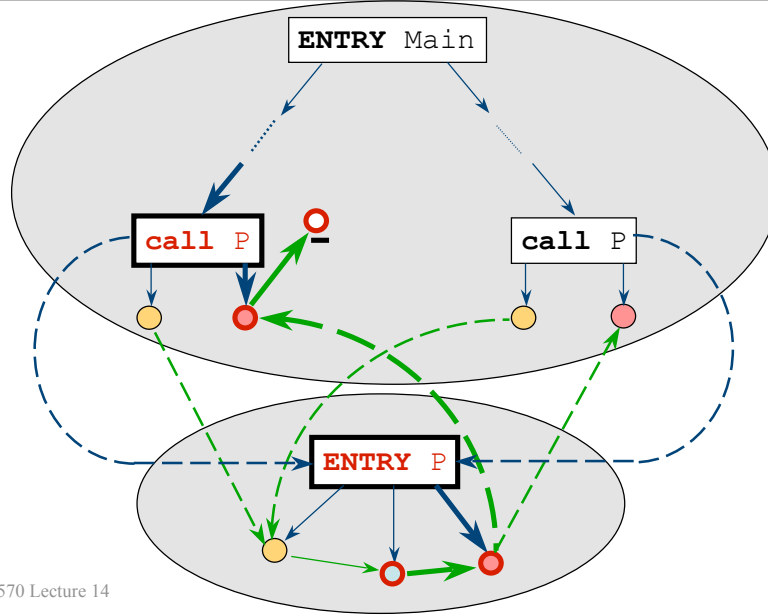
Interprocedural Backward Slice



CIS 570 Lecture 14

31

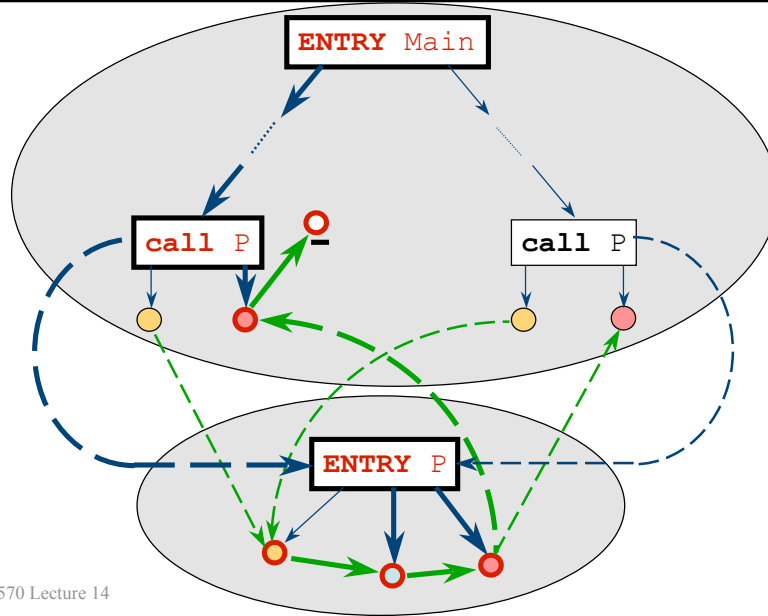
Interprocedural Backward Slice



CIS 570 Lecture 14

32

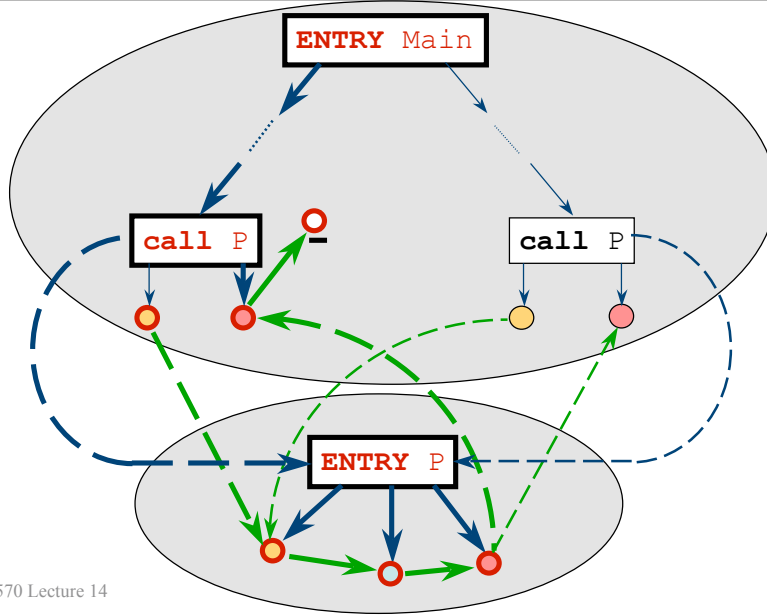
Interprocedural Backward Slice



CIS 570 Lecture 14

33

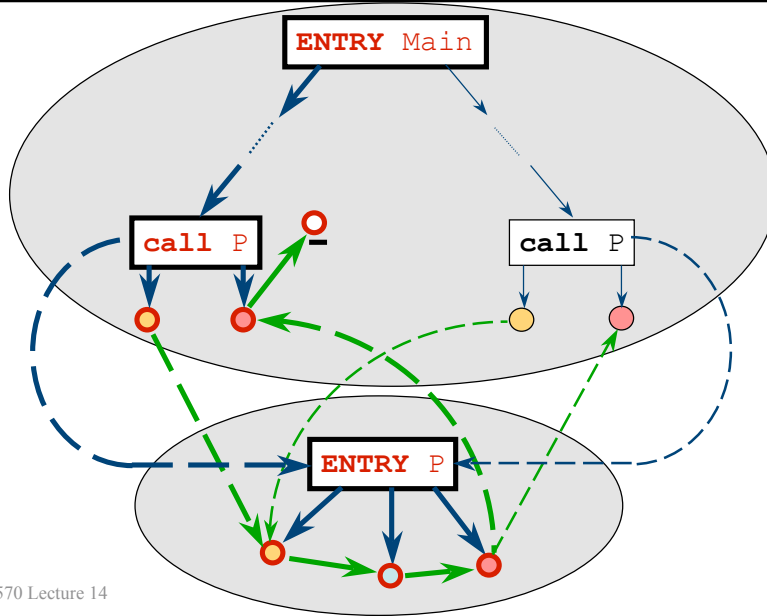
Interprocedural Backward Slice



CIS 570 Lecture 14

34

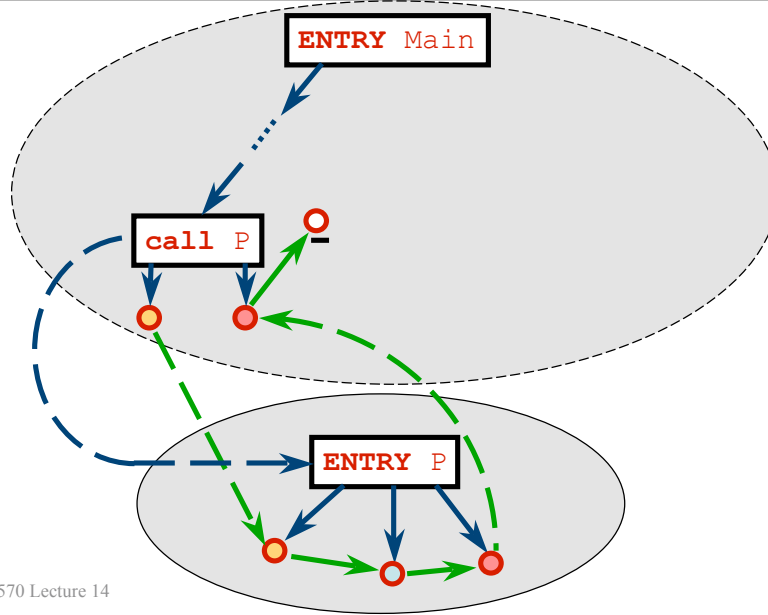
Interprocedural Backward Slice



CIS 570 Lecture 14

35

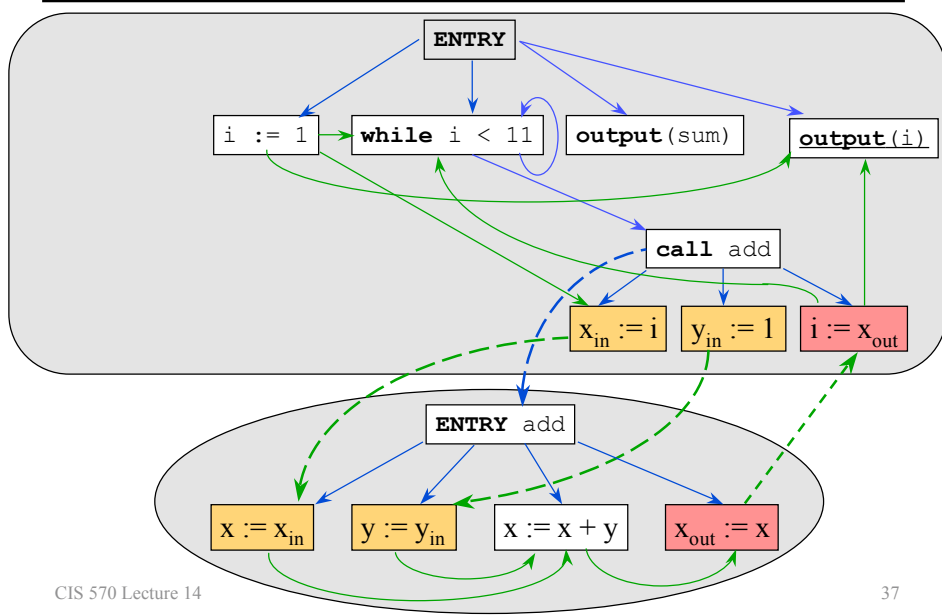
Interprocedural Slice Extraction



CIS 570 Lecture 14

36

Slice of the Sum Program



CIS 570 Lecture 14

37

Concepts

Program slicing

- Backward slice
- Forward slice
- Chopping

Program representations

- Program Dependence Graph
- System Dependence Graph

Next Time

Next lecture

- More modern uses of compilers