Program Slicing:

Definition

&

Elementary Techniques

source

A Survey of Program Slicing,

Binkley and Gallagher

Advances in Computers, 1996.

available on class web site

Preliminaries

- **State:** Variables => Values
- **Statement:** State => State
- **Program:** Sequence of statements
- **Slicing criterion <v,n> (of program P)**
  - **v** variable
  - **n** statement number

Finally!

- **Program Slice, S, (of P) at <v,n>**
  - **Only those statements needed to *capture the behaviour* of v at n.**
- **Every program has itself as a slice on any criteria.**
- So what does “capture the behaviour” mean?

Formally

- **Executable Program Slice, S, (of P)**
  - S can obtained from deleting 0 or more statements from P
  - If P halts on input I, then the value of **v** at statement **s** each time **s** is executed is the same in P and S.
- This is **static**
- This is also **backward**
- Look at slices from last lecture

But..

- Slices do not need to be executable
  - Executable / non-executable
  - Compliable or not.
- Slices do not need to be static
  - Static / Dynamic
  - All inputs or 1 input
- Slices do not need to be backward
  - Forward / backward
  - Or both!

Background (jargon)

- Graph: N(odes); E(dges). Digraphs
- “Immediate predecessor”
- “Immediate successor”
- Path
- Dominators
- Restrict to single-entry, single-exit

More…

- Change programs into graphs
  - Statements are nodes
  - Edges show control
- At each node:
  - **Refs(n)**, the variables referenced at n
  - **Defs(s),** the variables defined at n

How to compute relevant sets (and slices)

1. .
2. ..
3. …
4. ….
5. …..
6. …….
7. more???

Compute Relevant Sets for <a, 8>

n	stmt	defs(n)	refs(n)	relevant
1.	c = 4;	c		
2.	b = c;	b	c	
3.	a = b + c;	a	b, c	
4.	d = a + c;	d	a, c	
5.	f = d + b;	f	d, b	
6.	a = d + 8;	a	____	
7.	b = f + 30;	b	____	
8.	a = b + c:	__	____	

Computing (and using) Control Sets

- control(n): the set of predicate statements that directly control the execution of statement n.
- Whenever **n** is added to a slice, so are the members of control(n).
- **and** <refs(control(n), control(n)> is added to criteria.
- At joins, relevant(k) = relevant(succ(k))

Control Sets

n	stmt	defs(n)	refs(n)	control(n)	rel(n)
1.	b = 4;				
2.	c = 2;				
3.	d = 3;				
3.	a = d;				
5.	if (a) then				
6.	d = b + d				
7.	c = b + d				
8.	else				
9.	b = b + 1				
10.	d = d + 1				
11.	endif				
12.	a = b + c;				
13.	print a				

def ref cont rel slice

```
while(X) {
    e = d;
    d = c;
    c = b ;
    b = a;
   }
```

Loops

- For loops, iterate until slice & relevant sets stabilize.
- What is (worst case) running time of this approach?
  - Call it homework…..
- What about **return, break, exit, goto**?
  - Call it more homework…..
- And procedure/method calls?