

Program Slicing

Collection of Examples

Jens Knoop, Markus Schordan, Gergő Barany

Institut für Computersprachen
Technische Universität Wien

Overview

- Definition of executable slice (liveness analysis)
- Example with scalar variables
- Example with pointers to stack-allocated variables
- Example with strong and weak update in alias analysis
- Example with non-cyclic dynamic data structures
- Example with cyclic data structures
- Comparison of slice size

Reaching Defs with/out Pointers

$ReachingDefs(var, n, F) =$ [without pointers]
let
 $F = (V, A, En, Ex), A' = A \setminus \{(x, n)\}$
return $\bigcup_{(x, n) \in A} \text{if } var \notin def(x) \text{ then } ReachingDefs(var, x, (V, A', En, Ex))$
else $\{x\}$

$ReachingDefs(var, n, F) =$ [with pointers]
let
 $F = (V, A, En, Ex), A' = A \setminus \{(x, n)\}$
return $\bigcup_{(x, n) \in A} \text{if } var \notin def(x) \text{ then } ReachingDefs(var, x, (V, A', En, Ex))$
else (if $\#def(x) = 1$ then $\{x\}$)
else $\{x\} \cup ReachingDefs(var, x, (V, A', En, Ex))$

Program Dependence Graph

$DataDep(P) =$
let $CFG(P) = (V, A, En, Ex)$
 $D = \bigcup_{n \in V, var \in use(n), x \in ReachingDefs(var, n, CFG(P))} \{(n, x)\}$
return (V, D)

$ProgramDep(P) =$
let $DataDep(P) = (V, D),$
 $ControlDep(P) = (V, C, In),$
return $(V, D \cup C)$

Static Slice

$ReachableNodes(v, G) =$
let $G = (V, A)$
return $\{v\} \cup \bigcup_{(v, x) \in A} ReachableNodes(x, (V, A \setminus \{(v, x)\}))$

$StaticSlice(P, n, Vars) =$
let $F = CFG(P)$
 $D = ProgramDep(P)$
return $\bigcup_{var \in Vars} \bigcup_{x \in ReachingDefs(var, n, F)} ReachableNodes(x, D)$

Example: Artificial Sum (only scalar vars)

```
main() {
0  int a,b,i,j,n,y;
1  n=read();
6  a=0;
7  b=0;
8  i=n;
9  j=n;
10 while (i>0) {
11  a=a+1;
12  i=i-1;
13  j=i;
14  while (j>0) {
15    b=b+1;
16    j=j-1;
17  };
18  y=a+b;
19  write(y);
}
```

$$y = n + \sum_{i=1}^{n-1} i = \sum_{i=1}^n i$$

Example: With Pointers

```

main() {
0  int a,b,i,j,n, *ap,*bp,**cp;
1  n=read();
2  cp=&bp;
3  ap=&a;
4  bp=ap;
5  *cp=&b;
6  a=0;
7  b=0;
8  i=n;
9  j=n;
10 while (i>0) {
11  *ap=*ap+1;
12  i=i-1;
13  j=i;
14  while (j>0) {
15    *bp=*bp+1;
16    j=j-1;
17  }
18  y=*ap + *bp;
19  write(y);
20 }

```

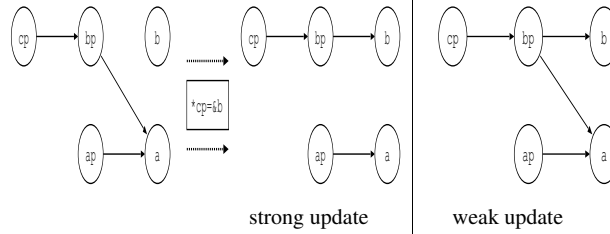
Slicing criterion 17: *ap

Example: Strong vs Weak Update

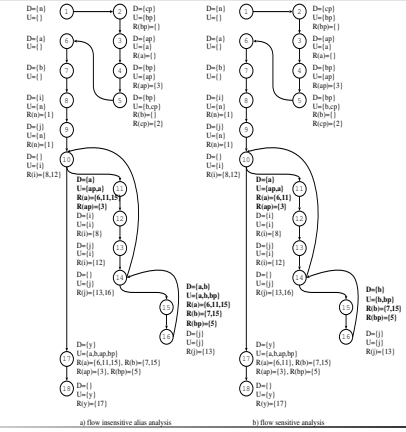
```

0  int a,b,i,j,*ap,*bp,**cp;
1  cp=&bp;
2  ap=&a;
3  bp=ap;
4  *cp=&b;

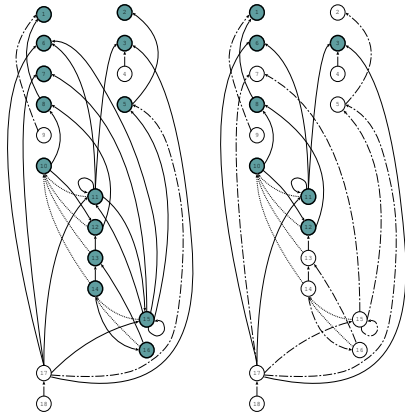
```



Example: Reaching Definitions



Example: Computation of Slice



Example: With Pointers (Slices)

```

main() {
1  n=read();
2  cp=&bp;
3  ap=&a;
4  bp=ap;
5  *cp=&b;
6  a=0;
7  b=0;
8  i=n;
9  j=n;
10 while (i>0) {
11  *ap=*ap+1;
12  i=i-1;
13  j=i;
14  while (j>0) {
15    *bp=*bp+1;
16    j=j-1;
17  }
18  y=*ap + *bp;
19  write(y);
20 }

```

Slicing criterion 17: *ap

Example: Comparison

Program	Aliasanalysis	Slice	S
Sum	none	{1, 6, 8, 10, 11, 12, 17}	
Sum (ptrs)	flow insensitive	{1,2,3,5,6,7,8,10,11,12,13,14,15,16,17}	
Sum (ptrs)	flow sensitive	{1, 3, 6, 8, 10, 11, 12, 17}	

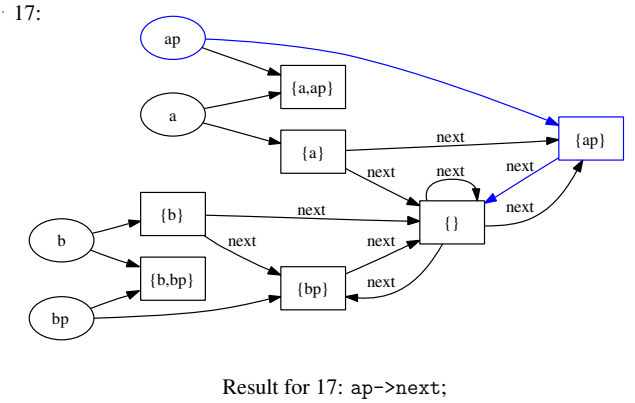
Example: With Dynamic Data Structures

```

main() {
  List *a,*b,*ap,*bp;
  int i,j,n;
1 n=read();
2 a=new List();
3 b=new List();
4 ap=a;
5 bp=b;
8 i=n;
9 j=n;
10 while (i>0) {
11   ap->next=new List();
11b  ap=ap->next;
12   i=i-1;
13   j=i;
14   while (j>0) {
15     bp->next=new List();
15b    bp=bp->next;
16     j=j-1;
17   }
17  ap->next = b; // conc
17b  y=a;
18  write(length(y)-2)
}

```

Example: Shape Analysis



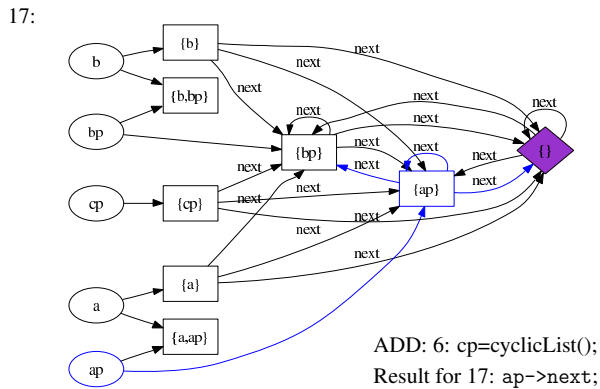
Example: With Dynamic Data Structures

```

main() {
  List *a,*b,*ap,*bp;
  int i,j,n;
1 n=read();
2 a=new List();
3 b=new List();
4 ap=a;
5 bp=b;
8 i=n;
9 j=n;
10 while (i>0) {
11   ap->next=new List();
11b  ap=ap->next;
12   i=i-1;
13   j=i;
14   while (j>0) {
15     bp->next=new Lis
15b    bp=bp->next;
16     j=j-1;
17   }
17  ap->next = b; // con
17b  y=a;
18  write(length(y)-2)
}

```

Example: Shape Analysis with Cycle



Example: Comparison

Program	Aliasanalysis	Inner Loop in Slice
Sum	none	No
Sum (ptrs)	flow insensitive (weak update)	Yes
Sum (ptrs)	flow sensitive (strong update)	No
Sum (dyn)	heap represented by 1 node only	Yes
Sum (dyn)	shape analysis (strong update)	No
Sum (dyn+cycle)	shape analysis	Yes