Chapter 4

Subset Based Analysis

4.1 Motivating Example

This subset based analysis is a way by which we use just a subset of field pointers used in a function to get more precise shape information.By the help of code present in Fig. 4.1 lets understand the usefulness of this technique.

Example 1 *Consider the code segment in Fig. 4.1 which has function's for searching data in a binary tree and inserting node into the same.The fields of structure Node are also shown.The structure Node has three field pointers Left,Right,Parent. In the insert function we can see a cycle getting created due to the lines S5 and S6. Take a look at Fig: 4.2(a), 4.2(b), 4.2(c) to see how the field sensitive analysis also gets the shape of p and s as a cycle at that point.*

Now come to the search function,here the Parent pointer is not at all used because just the Left and Right pointers are enough for searching.When the search function is called with the root of the tree as a parameter it's known that shape of the node at that pointer is a cycle.Its easily known that the cause of the shape of it to be cycle is *the statement $p \rightarrow Parent = s$; Going exactly by the links created in the heap it a cycle,but for the search function it's a tree only as it does not use Parent link. What if we can find the shape of node pointed to by a pointer considering only the fields which are really used in a function (in this case Left,Right). Then we can tell the shape of node pointed to by root as a tree. That is what our subset based analysis does.*

```
Struct Node {
  Struct Node *Left,*Right,*Parent;
  int key;
};
typedef Struct Node Node;

bool search(Node *root,int key){
if(root)
  return (key==s->ket)||search(s->Left,key)||search(s->Right,key);
return 0;
}

void insert(Node *root,int key){
    Node *s=root; //This pointer is used for traversing the Tree
    ..
    //New node is inserted as a child of s(s can be any node of the tree)
    Node *p;
    S1. p=(Node *)malloc(sizeof(Node));
    S2. p->Left=NULL;
    S3. p->Right=NULL;
    S4. p->key=key;
    S5. s->Left=p;
    S6. p->Parent = s;
    ..
}
```

Figure 4.1:

4.1.1 Analysis

There are two places where the Field Sensitive analysis needs to be modified for the subset based analysis to occur. First one is during pre processing of code,we parse each function separately and create a set for each of them.This set will contain

After	$Left_{p,t}$	$Parent_{t,p}$
`S1`	false	false
`S2`	false	false
`S3`	false	false
`S5`	true	false
`S6`	true	true

(a) Boolean Variables

After Stmt	D_F			I_F		
`S1`		p	s		p	s
	p	ϵ	$\emptyset$	p	$\{(\epsilon,\epsilon)\}$	$\emptyset$
	s	$\emptyset$	$\emptyset$	s	$\emptyset$	$\emptyset$
`S2`		p	s		p	s
	p	ϵ	$\emptyset$	p	$\{(\epsilon,\epsilon)\}$	$\emptyset$
	s	$\emptyset$	$\emptyset$	s	$\emptyset$	$\emptyset$
`S3`		p	s		p	s
	p	ϵ	$\emptyset$	p	$\{(\epsilon,\epsilon)\}$	$\emptyset$
	s	$\emptyset$	$\emptyset$	s	$\emptyset$	$\emptyset$
`S5`		p	s		p	s
	p	ϵ	$\emptyset$	p	$\{(\epsilon,\epsilon)\}$	$\{(\epsilon, Left^D)\}$
	s	$\{Left^D\}$	$\emptyset$	s	$\{(Left^D,\epsilon)\}$	$\emptyset$
`S6`		p	s		p	s
	p	ϵ	$\{Parent^D\}$	p	$\{(\epsilon,\epsilon)\}$	$\{(\epsilon, Left^D), (Parent^D,\epsilon)\}$
	s	$\{Left^D\}$	$\emptyset$	s	$\{(Left^D,\epsilon), (\epsilon, Parent^D)\}$	$\emptyset$

(b) Direction (D_F) and Interference (I_F) matrices

Heap Pointer	Boolean Equations
p_{cycle}	$\{\ (Parent_{p,s} \wedge s^{in}_{cycle}) \vee (Parent_{p,s} \wedge (\lvert D[s,p]\rvert >= 1))\ \} \vee \{\ Left_{s,p} \vee (\lvert D[p,s]\rvert >= 1)\ \}$
p_{dag}	$\{\ (Parent_{p,s} \wedge (\lvert I[p,s]\rvert > 1))\ \}$
s_{cycle}	$\{\ Parent_{p,s} \wedge (\lvert D[s,p]\rvert >= 1)\ \} \vee \{\ (Left_{s,p} \wedge p^{in}_{cycle}) \vee (Left_{s,p} \wedge (\lvert D[p,s]\rvert >= 1))\ \}$
s_{dag}	$\{\ (Left_{s,p} \wedge (\lvert I[s,p]\rvert > 1))\ \}$

(c) Boolean Equations after S6

Figure 4.2:

the all field pointers that are used at least once in the function.Let it be denoted by S_F for some function F.For the functions present in Fig. 4.1

$S_{insert} = \{$ Left , Right , Parent $\}$

$S_{search} = \{$ Left , Right $\}$

The other change is during the evaluation of the boolean equations.Let us say we have a reached some statement in some function F during our analysis and we are to evaluate the equation at that point.As we can see in Fig. 4.2(c) a boolean equation would contain terms like

$f_{pq} \quad, |D[p,q]| \quad, |I[p,q]| \qquad p, q \in \mathcal{H}, f \in \mathcal{F}$

Lets look at how we handle these terms when they are encountered in the evaluation.

- f_{pq} : We check whether the boolean variable f_{pq} value only if f is used at some point in function F, i.e $f \in S_F$ If its present then based on the value we proceed normally otherwise its **False**

$$f_{pq} \quad = \quad \begin{cases} \textbf{True} & f \in S_F, f_{pq} = \textbf{True} \\ \textbf{False} & \text{Otherwise} \end{cases}$$

- $|D[p,q]|$: This term indicates how many ways can a node pointed by pointer p reach the node pointed by pointer q .But for this analysis we need to count all those which reach only through fields present in S_F .To find this first count the paths possible through the fields not present in S_F and subtract it from total paths.

$$D^*[p,q] \quad = \bigcup_{f \in \mathcal{F}, f \notin S_F} \{D[p,q] \triangleright f\}$$

  $D^*[p,q]$ contains the paths which go through fields not present in S_F. Now instead of using the value of $|D[p,q]|$ we use $|D[p,q] - D^*[p,q]|$

- $|I[p,q]|$: This term indicates how many ways can a node pointed by pointer p and the node pointed by pointer q interfere .But for this analysis we need to count all those which interfere only through fields present in S_F .To find this first count the number of ways they can interfere through the fields not present in S_F and subtract it from total ways.

$$I^*[p,q] \quad = \bigcup_{f \in \mathcal{F}, f \notin S_F} \{I[p,q] \triangleright f\}$$

$I^*[p,q]$ contains the number of ways they can interfere through fields not present in S_F. Now instead of using the value of $|I[p,q]|$ we use $|I[p,q] - I^*[p,q]|$

Now that we have all the details about the analysis,lets look at how it works for the small code snippet present in Fig. 4.3(a).

Example 2 *This is just a small program with function Foo getting called from main with p as the parameter.We could see by the statements $p \rightarrow f = q$; and $q \rightarrow f = p$; that p point's to Node which is cycle.Though p is passed to the Foo ,it does not do any operation involving g. With the help of Direction,Interference matrices,Boolean Equations,boolean variables and Subsets of functions present in Fig. 4.3 lets discuss how this technique work. The Fig. 4.3(f) gives us the boolean equation of pointer a at statement S7.Lets look at each term individually and try to find its values as per this technique.As the field $g \in S_{Foo}$ and also looking at Fig. 4.3(b) (a and p are aliases) its clear that $g_{a,q}$ = **True**. Now coming to term $|I[a,q]|$,the field which is not present in S_{Foo} is f so as discussed earlier the entry present in $I^*[a,q]$ is $\{(f^D, \epsilon)\}$*

$$\begin{aligned} I[a,q] &= \{(f^D,\epsilon),(g^D,\epsilon)\} \\ I^*[a,q] &= \{(f^D,\epsilon)\} \\ I[a,q] - I^*[a,q] &= \{(g^D,\epsilon)\} \\ |I[a,q] - I^*[a,q]| &= 1 \end{aligned}$$

As $f \notin S_{Foo}$ we simple consider its value to be ***False*** *.If we substitute all these calculated values we get a_{dag} as 0.Meaning its not a Dag.As we already know it cannot be a cycle at all any way we haven't considered a_{cycle}.the shape of pointers pand qafter each statement is shown in Fig. 4.3(d)*

```
main()
{
  Node *p,*q;
  S1 p=(Node *)malloc(sizeof(Node);
  S2 q=(Node *)malloc(sizeof(Node);
  S3 p->f=q;
  S4 p->g=q;
  S5 Foo(p);
  S6 p=NULL;
}
```

```
struct Node{
  struct Node *f,*g;
};
typedef struct Node Node;
void Foo(Node *a)
{
S7 Node *b=malloc();
S8 b->g=a;
}
```

(a) Example Code

After	$f_{p,q}$	$g_{p,q}$
S1	false	false
S2	false	false
S3	true	false
S4	true	true
S7	true	true
S8	true	true

(b) Boolean variables

Function	S_F
main	$\{ f , g \}$
Foo	$\{ g \}$

(c) Field Subsets

After	p	q	a
S1	Tree	Tree	Tree
S2	Tree	Tree	Tree
S3	Tree	Tree	Tree
S4	**Dag**	Tree	Tree
S7	**Tree**	Tree	**Tree**
S8	**Tree**	Tree	**Tree**
S8	**Dag**	Tree	Dag

(d)Shape after each statement

After Stmt	D_F			I_F		
$S1$		p	q		p	q
	p	ϵ	$\emptyset$	p	$\{(\epsilon, \epsilon)\}$	$\emptyset$
	q	$\emptyset$	$\emptyset$	q	$\emptyset$	$\emptyset$
$S2$		p	q		p	q
	p	ϵ	$\emptyset$	p	$\{(\epsilon, \epsilon)\}$	$\emptyset$
	q	$\emptyset$	ϵ	q	$\emptyset$	$\{(\epsilon, \epsilon)\}$
$S3$		p	q		p	q
	p	ϵ	$\{f^D\}$	p	$\{(\epsilon, \epsilon)\}$	$\{(f^D, \epsilon)\}$
	q	$\emptyset$	ϵ	q	$\{\epsilon), (f^D\}$	$\{(\epsilon, \epsilon)\}$
$S4$		p	q		p	q
	p	ϵ	$\{f^D, g^D\}$	p	$\{(\epsilon, \epsilon)\}$	$\{(f^D, \epsilon), (g^D, \epsilon)\}$
	q	$\emptyset$	ϵ	q	$\{\epsilon, (f^D), (\epsilon, g^D)\}$	$\{(\epsilon, \epsilon)\}$
$S7^*$		a	s		a	s
	a	ϵ	$\{f^D, g^D\}$	a	$\{(\epsilon, \epsilon)\}$	$\{(f^D, \epsilon), (g^D, \epsilon)\}$
	q	$\emptyset$	ϵ	q	$\{\epsilon, (f^D), (\epsilon, g^D)\}$	$\{(\epsilon, \epsilon)\}$

(e) Direction (D_F) and Interference (I_F) matrices

Heap Pointer	Boolean Equation
a	$\{ \ (g_{a,q} \wedge (\lvert I[a,q] \rvert > 1)) \vee (f_{a,q} \wedge (\lvert I[a,q] \rvert > 1)) \ \}$

(f) a's Dag Boolean Equation at statement S7

Figure 4.3: