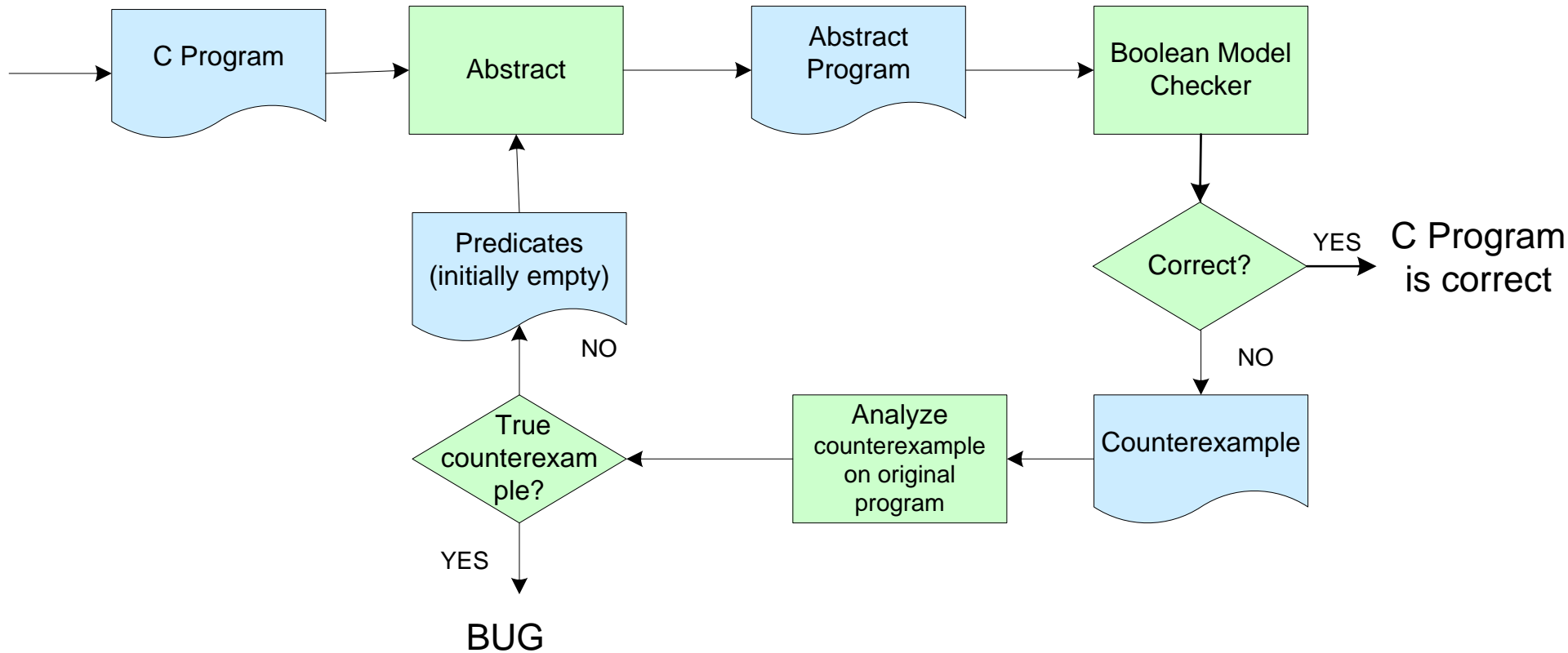


SLAM III: Discovering Predicates

V&T

Roderick Bloem

The Approach



Today's Problem

- Figuring out that a counterexample is infeasible
- Finding new predicates

First, recapitulate example

```
1. void example(){
2.     do{
3.         lock();
4.         nPacketsOld = nPackets;
5.         req = devExt->WLHV;
6.         if(req && req-> status){
7.             devExt->WLHV = req->next;
8.             release();
9.             irp = req->irp;
10.            if(req->status > 0){
11.                irp->IoS.status = SUCCESS;
12.                irp->IoS.Info = req->Stat;
13.            } else {
14.                irp->IoS.status = FAIL;
15.                irp->IoS.Info = req->Stat;
16.            }
17.            smartDevFreeBlock(req);
18.            IoCompleteRequest(irp);
19.            nPackets++;
20.        } // if req
21.    } while(nPackets != nPacketsOld);
22.    release();
23. }
```

Program

```
1. void example() {
2.     do{
3.         lock();
4.         skip;
5.         skip;
6.         if(*) {
7.             skip;
8.         release();
9.         skip;
10.        if(*) {
11.            skip;
12.            skip;
13.        } else {
14.            skip;
15.            skip;
16.        }
17.        skip;
18.        skip;
19.        skip;
20.    } // if
21. } while(*);
22. release();
23. }
```

First Boolean Program

Boolean Counterexample

```

1. void example(){
2.     {
3.         lock();
4.         skip;
5.         skip;
6.         {
7.             skip;
8.             release();
9.             skip;
10.        {
11.            skip;
12.            skip;
13.        }
14.
15.        skip;
16.        skip;
17.        skip;
18.    }
19.    release();
20. }

```

```

1. void example(){
2.     do{
3.         lock();
4.         nPacketsOld = nPackets;
5.         req = devExt->WLHV;
6.         if(req && req-> status){
7.             devExt->WLHV = req->next;
8.             release();
9.             irp = req->irp;
10.            if(req->status > 0){
11.                irp->IoS.status = SUCCESS;
12.                irp->IoS.Info = req->Stat;
13.            } else {
14.                irp->IoS.status = FAIL;
15.                irp->IoS.Info = req->Stat;
16.            }
17.            smartDevFreeBlock(req);
18.            IoCompleteRequest(irp);
19.            nPackets++;
20.        } // if req
21.    } while(nPackets != nPacketsOld);
22.    release();
23. }

```

Counterexample in C

```
1. void example() {
2.     {
3.         lock();
4.         nPacketsOld = nPackets;
5.         req = devExt->WLHV;
6.         {assume(req && req->status);
7.             devExt->WLHV = req->next;
8.             release();
9.             irp = req->irp;
10.        {assume(req->status > 0)
11.            irp->IoS.status = SUCCESS;
12.            irp->IoS.Info = req->Stat;
13.        }
14.
15.
16.
17.        smartDevFreeBlock(req);
18.        IoCompleteRequest(irp);
19.        nPackets++;
20.    }
21.    } assume(nPackets == nPacketsOld);
22.    release();
23. }
```

How do we show this counterexample is infeasible?

Reminder: Hoare Axiom for Assignment

An example:

$$\{b+c = 5\}$$
$$a = b + c$$
$$\{a = 5\}$$

This is the *weakest precondition*:

$$Q[x \rightarrow e] \text{ holds before } x = e$$

if and only if

$$Q \text{ holds afterwards}$$

In general:

$$\{Q[x \rightarrow e]\}$$
$$x = e;$$
$$\{Q\}$$

$[x \rightarrow e]$ means that x is replaced
by e


```
lock () ;
```

```
nPacketsOld = nPackets;
```

```
req = devExt->WLHV;
```

```
assume(req && req->status)
```

```
devExt->WLHV = req->next;
```

```
release () ;
```

```
irp = req->irp;
```

```
assume(req->status > 0)
```

```
irp->IoS.status = SUCCESS;
```

```
irp->IoS.Info = req->Stat;
```

```
smartDevFreeBlock(req);
```

```
IoCompleteRequest(irp);
```

```
nPackets = nPackets + 1;
```

```
assume(nPackets == nPacketsOld)
```

```
release () ;
```

```

lock();
% FALSE && devExt->WLHV->status>0
% && devExt->WLHV && devExt->WLHV->status
npo = np;
% np+1==npo && devExt->WLHV->status>0
% && devExt->WLHV && devExt->WLHV->status
req = devExt->WLHV;
% np+1==npo && req->status>0 &&
% req && req->status
assume(req && req->status)
% np+1==npo && req->status>0
devExt->WLHV = req->next;
% np+1==npo && req->status>0
release();
% np+1==npo && req->status>0
irp = req->irp;
% np+1==npo && req->status>0
assume(req->status > 0)

```

```

% np+1==npo
irp->IoS.status = SUCCESS;
% np+1==npo
irp->IoS.Info = req->Stat;
% np+1==npo
smartDevFreeBlock(req);
% np+1==npo
IoCompleteRequest(irp);
% np+1==npo
np = np + 1;
% np==npo
assume(np == npo)
%TRUE
release();
% TRUE

```

Computed from bottom to top

Note the use of colors!

Assumption: function calls do not affect predicates

```
lock () ;
```

```
nPacketsOld = nPackets;
```

```
req = devExt->WLHV;
```

```
assume(req && req->status)
```

```
devExt->WLHV = req->next;
```

```
release () ;
```

```
irp = req->irp;
```

```
assume(req->status > 0)
```

```
irp->IoS.status = SUCCESS;
```

```
irp->IoS.Info = req->Stat;
```

```
smartDevFreeBlock(req);
```

```
IoCompleteRequest(irp);
```

```
nPackets = nPackets + 1;
```

```
assume(nPackets == nPackets0)
```

```
release () ;
```

Adding Predicates

if we add

- `{nPackets==nPacketsOld}` and
- `{nPackets+1==nPacketsOld}`

Red predicate is false at every line → path is no longer possible in abstract program.

Note:

- Infeasibility is computed bottom-up,
- Functions (`smartDevFreeBlock`, etc.) must inlined or analyzed
- Path consists of
 - assignments (change the predicates)
 - assumes (derive from if, while, add a predicate)

Dealing with assertions

Consider this program

```
1. x = 4;  
2. if (x == 4) {  
3.   x = x + 1;  
4. }  
5. assert (x==5);
```

and the predicate $b: \{x==5\}$

Abstract program:

Counterexample:

```
x = 4;
```

```
assume (x == 4)
```

```
x = x + 1;
```

```
assume (x != 5) ;
```

New predicate:

Dealing with asserts

Consider this program

```
1. x = 4;  
2. if (x == 4) {  
3.   x = x + 1;  
4. }  
5. assert (x==5);
```

and the predicate $b : \{x==5\}$

The abstract program is:

```
1. b = FALSE;  
2. if (b?false:*) {  
3.   b = b? FALSE : *;  
5. }  
6. assert (b);
```

A counterexample:

1, 2, 3, 4, 5:

```
1. x = 4;  
2. assume (x == 4)  
3. x = x + 1;  
4.  
5. assume (x!=5);
```

From here you can compute new predicate. (see next slide)

Note: The broken assertion becomes an assumption that the condition is false

Computing Infeasible Paths

In: a path P with l lines.

```

lp := l + 1 // line pointer
condition[lp] := true; // condition for line lp, a condition is a
                        // set of predicates

while(lp ≠ 0 && condition[lp] ≠ false){
  lp--
  if(statement(lp) = "assume(p)") {
    condition[lp] := "condition[lp+1] && p", use new color for p
  } else if(statement(lp) = "x := e"){
    condition[lp] := condition[lp+1][x → e]
  }
  simplify condition
}

if(condition[lp]= ... &&false&&...){
  report INFEASIBLE!
  add all predicates from all condition[i] with same color as false
} else {
  report FEASIBLE: BUG FOUND
}

```


Excluding paths

Guarantee: New predicates exclude the infeasible path

Hope: They also exclude other paths

How do you check whether expression equals FALSE?

- SLAM pretends that `ints` are integers and `floats` are real numbers, uses theorem prover.
 - Not precise (overflows, limited precision for floats)
 - Theorems over integers are undecidable
 - Theorem provers are relatively fast.
- Better: `ints` are bit vectors
 - SMT solvers are fast for bit vectors
 - Decidable, precise

Undecidability

- Equality of integer formulas including multiplication is undecidable!
 - There is no algorithm to decide formulas like
 - Are there x, y, z such that $x*x*x + y*y*y = z*z*z$?
 - Are there $x, y, z, n > 2$ such that $x^n + y^n = z^n$? (Fermat's theorem)
- Without multiplication everything is OK.
- Use a theorem prover to prove equivalence, theorem prover may never return an answer, or say “I don't know”

Conclusion

You can find infeasible paths by repeated application of Hoare's axiom for assignment

You color each predicate that you find, if one predicate is transformed to false, add all predicates of that color.