SLAM
Motivation & Example

Verification & Testing

Roderick Bloem

This week: SLAM

Automatically verify properties of drivers

Part of MS Windows Driver Kit (called the *Static Driver Verifier*)

Key: automatic abstraction

Based on: Ball & Rajamani, Automatically Validating Temporal Safety Properties of Interfaces, SPIN Workshop on Software Model Checking, 2001

Why Drivers?

Drivers are…

…critical

- Run in kernel space, wreak havoc

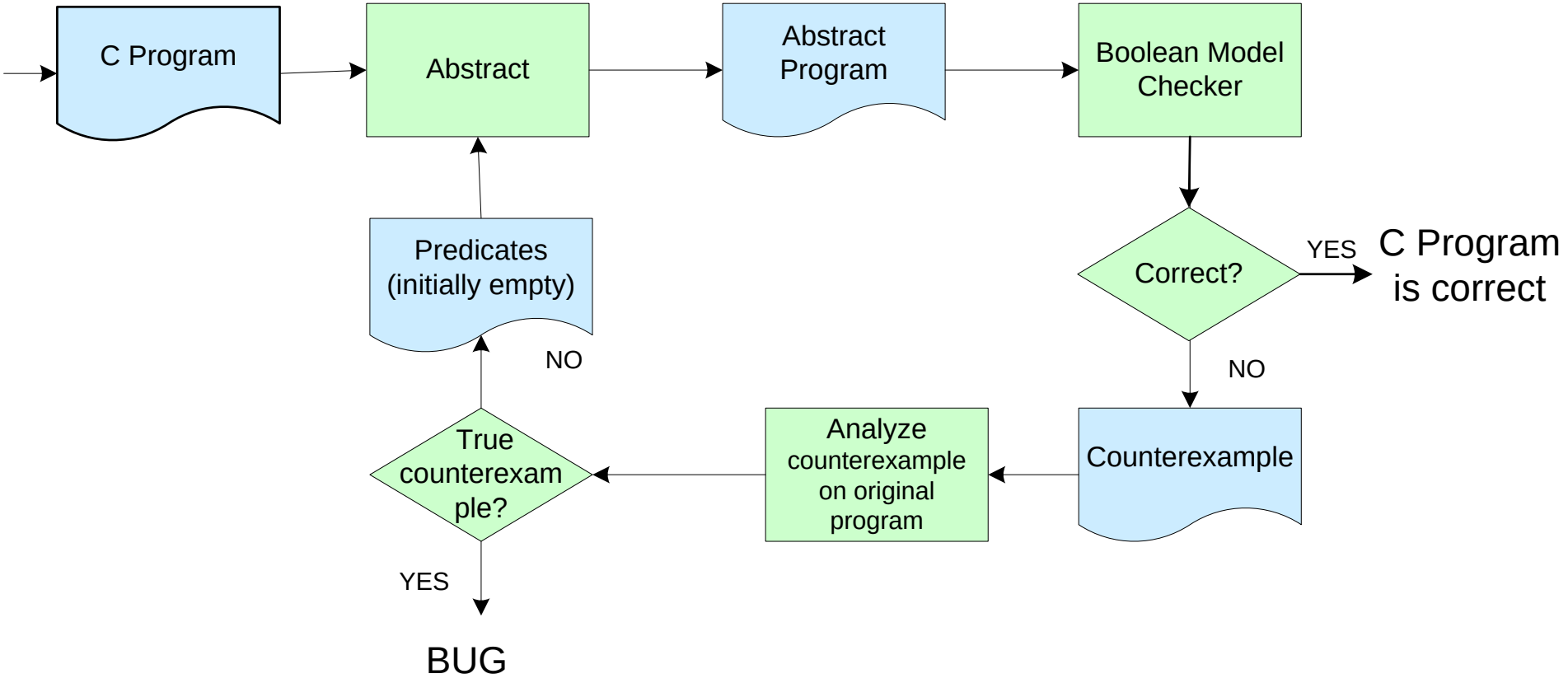
…not under MS control

- Developed by hardware companies (limited understanding of Windows)
- Cannot: verify correctness, impose coding standards, educate designers,

…simple

- Small code base
- Important properties: locking protocol, …
- Correctness may not depend on implementation details

The Approach



The Approach

- Construct abstraction of program.
  - **Abstraction adds behavior**
- Abstraction may have counterexample that is impossible on real program
- Make abstraction more precise until
  - The abstraction contains no counterexamples
  - We find a real bug
- We use predicate abstraction
- The result of the abstraction is a Boolean Program

Approach

First set of predicates is empty

Abstraction engine uses predicates to construct Boolean Program

- First approximation has only data flow, no variables

Run model checker for Boolean programs

- No counterexample? C program is correct! Stop.

Analyze counterexample

- If it is real, we have found a real bug. Stop
- If not, add predicates to make abstraction more precise. Start from 2.

There are several reasons that this may not work (undecidability!).

Boolean Programs

- Functions with parameters, recursion
- Global and local Variables

- No mallocs and frees
- Only Boolean (one-bit) variables; no integers
- Nondeterminism: *
- assert , assume

Theory: Boolean programs are pushdown automata

  – Checking Boolean programs is not hard (algorithm next week)

Some Syntax

- `a ? b : c`
  - meaning: `if a then b else c`.
  - Example: `b = c ? 2 : 3`
- `assert(e)`
  - meaning: dump core unless `e` is true
- `assume(e)`
  - meaning: executions with `e` false are irrelevant.
- `*`
  - meaning: function that evaluates to 0 or 1 nondeterministically
  - example: `if(*) b = 1; else b=2;` evaluates to `b=1` or `b=2`, but never to `b=3`.
  - `*` is a function, not a value (After `b = *, b` equals `0` or `1`, `b` cannot equal `*`)

Choose

SLAM papers use choose function:
choose(f, g) is the same as
f ? true : (g? false: *);

i.e.,

- If f is true then true,
- if g is true then false,
- if neither are true then nondeterministic
- Both are true should be impossible

Example: Specification

```
int isLocked = 0;

void lock(){
  assert(!isLocked);
  isLocked = true;
}

void release(){
  assert(isLocked);
  isLocked = false;
}
```

```
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    }
21  } while(nPackets != nPacketsOld);
22  release();
```

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.
16.
17.     skip;
18.     skip;
19.     skip;
20.     }
21.   }
22.   release();
23. }
```

```
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    }
21  } while(nPackets != nPacketsOld);
22  release();
```

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

```
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      asssume(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    }
21  } assume(nPackets == nPacketsOld);
22  release();
```

Which Predicate can Prove Counterexample Infeasible?

Which Predicate can Prove Counterexample Infeasible?

{nPackets == nPacketsOld}

b: nPackets == nPacketsOld

```
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    }
21  } while(nPackets != nPacketsOld);
22  release();
```

Second Boolean Program

```
1.  void example(){
2.    do{
3.      lock();
4.      b = true;
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.       b = b ? false: *;
20.     }// if
21.   } while(!b);
22.   release();
23. }
```

Second Boolean Program

Is correct – we are done.

The Approach