Automatic Predicate Abstraction of C Programs

Thomas Ball
Microsoft

Rupak Majumdar
UC Berkeley

Todd Millstein
U Washington

Sriram K. Rajamani
Microsoft

http://research.microsoft.com/slam/

Verifying Temporal Properties of Software

do {
    AcquireLock(idev->lock);
    nPacketsOld = nPackets;
    if (request != request Keeper) {
        ReleaseLock(idev->lock);
        nPackets = nPacketsOld;
        ReleaseLock(idev->lock);
    } while (nPackets != nPacketsOld);
    ReleaseLock(idev->lock);

    Question: Is locking protocol respected?
do {
    AcquireLock(&dev->lock);
    nPacketsOld = nPackets;
    if(!b) {
        ReleaseLock(&dev->lock);
        nPackets++;
    }
} while (nPackets != nPacketsOld);
ReleaseLock(&dev->lock);

Predicate Abstraction

C program

E = \{e_1, ..., e_n\}

Predicate Discovery (Newton)

Predicate Abstraction (C2bp)

Boolean program

Model Checking (Bebop)
(PASTE 2001, Qex 2000)

no

yes

C2bp: Predicate Abstraction for C Programs

Given

- P: a C program
- E = \{e_1, ..., e_n\}: set of C boolean expressions over the variables in P
  - no side effects, no procedure calls

Produce a boolean program B

- same control-flow structure as P
- only vars are 3-valued booleans \{b_1, ..., b_n\}
- properties true of B are true of P
Contributions
- C2bp handles the constructs of C
  - pointers, recursion, structs, casts
  - modular abstraction of procedures
  - provably sound
  - successfully applied to verify properties of device drivers

C2bp Algorithm
- operates on intermediate representation
  - contains only assignments, conditionals, procedure calls, gotos
- abstracts each statement in isolation
  - no control-flow analysis
  - no need for loop invariants

Abstracting Assignments
replace assn statement s with a parallel assignment to boolean vars in scope:
- predicate e, is true after s iff wp(s, e) is true before s
  \[ b_i = wp(s, e_i); \]
- example:
  \[ wp(y=y+1, x=x*y) = (x=x*y)(y+1/y) = (x=x*y+1) \]
  - but wp(s, e) may not be expressible in terms of \((e_1, \ldots, e_n)\) ...
Strengthening

$S(e)$ is the best predicate over $\{e_1, \ldots, e_n\}$ that implies $e$:
- A minterm is a conjunction $d_1 \land \ldots \land d_n$, where $d_i = e_i$ or $d_i = \neg e_i$.
- $S(e) = \text{disjunction of all minterms that imply } e$.
- Use decision procedure to check implication.

Abstracting Assignments

- $S(wp(s, e_i))$ is true before $s$ implies predicate $e_i$ is true after $s$.
- $S(wp(s, e_i))$ is true before $s$ implies predicate $e_i$ is false after $s$.

$$b_i = S(wp(s, e_i)) \ ? \ true : S(wp(s, e_i)) \ ? \ false : *;$$

Assignment Example

<table>
<thead>
<tr>
<th>Statement in $P$:</th>
<th>Predicates in $E$:</th>
</tr>
</thead>
<tbody>
<tr>
<td>$y = y+1$;</td>
<td>$(x = y)$</td>
</tr>
</tbody>
</table>

Weakest Precondition:
$wp(y = y+1, x = y) = x = y+1$

Strengthenings:
- $S(x = y+1) = \text{false}$
- $S(x = y+1) = x = y$

Abstraction of $s$ in $B$:
- $b = b \ ? \ false : *;$
Handling Pointers

Statement in P:  
*p = 3;
(x==5)

Weakest Precondition:
wp(*p=3, x==5) = x==5  What if *p and x alias?

Correct Weakest Precondition:
(p==&x && 3==5) || (p!=&x && x==5)

We use Das’s pointer analysis [PLDI 2000] to prune disjuncts representing infeasible alias scenarios.

Abstracting Conditionals

in P:

if (expr) {...} else {...}

in B:

if (*) {assume(W(expr)); ...} else {assume(W(!expr)); ...}

weakening: W(expr) = !S(!expr)

Handling Procedures

- each predicate in E is annotated as being either global or local to a particular procedure
- procedures abstracted in two passes:
  - a signature is produced for each procedure in isolation
  - procedure calls are abstracted given the callees’ signatures
Formal Properties

- **soundness** [Ball, Milstein & Rajamani, MSR-TR]
  - B has a superset of the feasible paths in P
  - b is true (false) at some point on a path in B implies e is true (false) at that point along a corresponding path in P
- **complexity**
  - linear in size of program
  - flow analysis deferred to the model checker
  - exponential in number of predicates

Experience

- checked several device drivers for proper usage of locks
  - proper handling of interrupt request packets (IRPs)
- verified four drivers from the Windows 2000 Driver Development Kit
- found a bug in IRP handling in a Microsoft-internal floppy device driver
- ran C2bp on 6500 LOC with 23 predicates in 98 seconds
- model checking takes under 10 seconds

Conclusions

- **C2bp**, the first automatic predicate abstractor for C code
  - pointers, procedures, recursion
- provably sound
- a necessary step for making software model checking a reality
- useful for other tasks that require predicate sensitivity

http://research.microsoft.com/slam/
Predicate-sensitive Alias Analysis

```c
prev = NULL;
newl = NULL;
while (curr) {
    next = curr->next;
    if (curr->val > v) {
        if (prev)
            prev->next = next;
        curr->next = newl;
    } else
        prev = curr;
    curr = next;
}
```

L: newl = curr; (curr!=NULL) && (prev==NULL || (prev->val <= v) && (curr->val > v))

implies

(prev != curr) at label L