Software Engineering: Where are we? And where do we go from here?

V22.0474-001 Software Engineering
Lecture 23

Clark Barrett
New York University
4/17/2006
• Between 1985 and 1987, at least 6 accidental radiation overdoses were administered.
• All the victims were injured, and 3 of them later died.
In 1996, an unmanned Ariane 5 rocket launched the European ace shortly after its lift off.

A value of rocket and cargo: 0.5 million
• In August, the largest loss out in our country's history occurred.
• Estimated cost to fix 3 or it alone: $1.1 billion.
Caused by Software Bugs!

- Each of the overdoses from the 5 herac 5 was the result of a user in the controlling software.

- The Ariane 5 explosion was the result of an unsafe floating oint to integer conversion in the rocket's software system.

- A software user caused an alarm system failure at 61rst near in Akron, Ohio. An early response to those alarms would likely have prevented the lac out.
Top Oxymoron from OxymoronList.com: Microsoft Works
- 7 hen we uild a rid e or a uildin , we don't e ect it to crum le and have to re uilt twice a wee . 7 h is software so much less relia le than rid es or uildin s8

- 9 o ou have the nowled e and s ills ou need to create ualit software8

- 7 hat have ou learned in this class that can hel 8

- 7 hat tools and techni ues do ou thin future software en ineers will use to create more relia le s stems8
• Formal software verification has been the old rail of computer science for many decades.

• Formal verification techniques can be used to prove that a piece of software is correct.

• There are still many challenges in making formal methods practical, but there are also some success stories.
- What is Formal Verification?
- Model Checking
- Theorem Proving
- Systems and Tools
- Create a mathematical model of the system
  - An inaccurate model can introduce or mas u s.
  - 6 ortunatel , this can often e done automaticall.

- ecif formall what the ro erties of the s stem should e

- rove that the model has the desired ro erties
  - uch etter than an testin method
  - overs all ossi le cases
  - 5 his is the hard art

- 5 here are a variet of tools and techni ues
• odel hec in
  ◆ 5 icall relies on low level Boolean logic
  ◆ roof is full automatic
  ◆ 9 oes not scale to lar e s stems

• 5 heorem rovin
  ◆ 5 icall uses more e ressive lo ic hi her order lo ic
  ◆ roof is manuall directed
  ◆ nlimited scala ilit

• Advanced techni ues com ine elements of oth
- What is Formal Verification?
- Model Checking
- Theorem Proving
- Systems and Tools
• 5 icall, a formal model is a graph in which each vertex represents a state of the system, and each edge represents a transition from one state to another.

• Consider this simple system:

```c
int x, y;
x = 0;
y = 0;
while (x < 3) {
    x++;
    y = y + x;
}
```

• The states of this system are all possible states of the variables and .

• Fortunately, we can restrict our attention to the states.
• 5 icall, a formal model is a rah in which each verte re represents a e of the ro ram, and each ed e re represents a ro from one state to another.
7. We can check a root vertex verification that it is reachable state. If the root is e, then there is a u.

Initial State

Final State

int x, y;
x = 0;
y = 0;
while (x < 3) {
    x++;
y = y + x;
}
7 e can check a root verifies if it is root in ever reachable state. If the root is e, then there is a unique.

```
int x, y;
x = 0;
y = 0;
while (x < 3) {
    x++;
    y = y + x;
}
```
7e can check a root verification that it is root in every reachable state. If the root is e, then there is a u.

```
int x, y;
x = 0;
y = 0;
while (x < 3) {
    x++;
    y = y + x;
}
```
7 e can check a ro e r t verif in that it is ro e in ever reacha le state. f the ro e r t is e, then there is a u.

\[ y = \frac{x(x+1)}{2} \]

Initial State

Final State

```plaintext
int x, y;
0 x = 0;
1 y = 0;
3 while (x < 3) {
  x++;
  y = y + x;
}
```
In practice, models of real ro rams would have too man states to modelcheck.

5 here are a num er of techni ues which can hel

- A straction
- 9 ecom osition
- m olic model check in

Itimatel , model check in alone cannot rove ro erties of lar e ro rams.
- What is Formal Verification?
- Model Checking
- Theorem Proving
- Systems and Tools
Theorem rovin relies on human in enuit and s m olic mani ulation to rove that a ro ram satisfies some ro ert.

5 icall, rovin a sin le ro ert a out a ro ram will re uire rovin man other ro erties as well.

ne a roach is to annotate the ro ram with theorems to e roved also called vr or er o, and then rove that each theorem reall does hold.
Consider a slightly modified version of our simple read ram from before this time there are many more reachable states.

Use we wish to prove that at the end of the read ram, \( x(x+1) \)

We can annotate the end of the read ram with this robust and worse towards from there.

```plaintext
int x, y;
x = 0;
y = 0;
while (x < 30) {
    x++;
    y = y + x;
}
```
5 o show this root, we must look at the two possible previous locations in the root.

6 or these two locations, it will be sufficient to prove that the root either doesn’t end or that the root holds.

7 with a bit of insight, we can see that these two formulas are more complicated than necessary. We can rewrite a formula relating it with a formula which implies it.

```c
int x, y;
x = 0;
y = 0;
x < 30 ? x = x(x + 1) : x;
while (x < 30) {
    x++;
    y = y + x;
    x < 30 ? x = x(x + 1) : x;
}
x(x)
```
5. To show this, we must look at the two possible previous locations in the RAM.

6. For these two locations, it will be sufficient to prove that the RAM either doesn't end or that the other holds.

7. With a bit of insight, we can see that these two formulas are more complicated than necessary. We can replace a formula by a formula which implies it.

```java
int x, y;
x = 0;
y = 0;
x <= 30 \rightarrow y = x(x + 1)
while (x < 30) {
    x++;
    y = y + x;
}
```

```java
x(x + 1)
```
7 hat is the condition that will guarantee the reen assertion after execute $y = y + x$

5 o find out, we ima ine tr in to rove the reen condition usin rimed varia les to re resent the value after $y = y + x$ and un rimed varia les for the value efore

$(0) \land \implies +x \land x = x \implies =x (x + )$

$(0) \implies x (x+ )$

$(0) \implies x(x )$

```c
int x, y;
x = 0;
y = 0;
x=0
while (x < 30) {
    x++;
y = y + x;
    =x(x+ )
}

x(x )
```
ow we must find an assertion which is im lied the loop end condition and the re loop condition, and which im lies the reen condition after e ecutin \( x++ \).

5 o do this, we first must stren then the re loop condition.

```cpp
int x, y;
x = 0;
y = 0;
x = 0 ∧ y = 0
while (x < 30) {
    x++;
    =x(x )
y = y + x;
    =x(x+ )
}
x(x )
```
ow we must find an assertion which is implied the loop end condition and the re loop condition, and which implies the reen condition after executing $x++$.

5 o do this, we first must strengthen then the re loop condition.

7 e finish with a set of assertions, each of which can be proven to follow from the annotations at all ossi le revious points in the pro ram.

```c
int x, y;
x = 0;
y = 0;
x = 0 \wedge y = 0
while (x < 30) {
    x += 1;
    y = y + x;
}
```
Notice that the final set of conditions on the number of loop iterations.

In fact, this same roof can be used regardless of what the loop condition is.

This is one advantage of theorem proving has over model checking.
ach roof from the assertion before a statement to the assertion after the statement is called a ver o 
o o .

, erification conditions can e roved usin an o e eore prover.

however, comin u with the assertions usuall re uires human uidance and can e quite challen in .
• What is Formal Verification?
• Model Checking
• Theorem Proving
• Systems and Tools
odel checker for finite state stems
Based on extremely efficient data structures for representing Boolean logic
, er successful for hardware

odel checker for parallel stems
imited to small state aces
Theorem

- Some interactive 5 theorem rovers
  - ,
  - A
  - sa elle

- Some automated domain specific theorem rovers
  - im lif
  - ,
stems esearch enter at

formerl om a , formerl 9

5 heorem rovin a roach for sim le roerties in ava

ser annotates code with e ected invariants

nvariants are verified using automated theorem rover im lif
lever combination of model checking and automated theorem proving

- An abstract RAM is created in which all conditions are replaced with Boolean variables
- Resulting Boolean RAM is model checked
- If model checking fails, the potential error is checked in the original RAM using an automated theorem prover

- Successful used to find issues in 7 indows drivers.
  - reducing the frequency of Blue screens of deathA
• Formal software, verification is starting to become tactical

• till lots of work to be done

• own can it make or a etter ro rammer8
  ◆ 9 ocument our code with the roerties and invariants that ou thin should be true
  ◆ 7 hen ou modif code, convince ourself that ou are not rea in an invariants
  ◆ earn more a out formal verification

• o efull, someday software will be as safe and relia le as the other objects uilt en ineers