Upcoming

• Homework 3 due March 22
• Literature review due today March 20
• Project plan assignment posted, due April 10

• Paper presentation instructions:

Repairing Automated Repair

Generalizing

• How many games are there in a 78-team bracket?

• What about an n-team bracket?
Repairing Automated Repair

What do cobras have to do with automated program repair?

Automated Program Repair

the many repair tools

Potential problem

the patched program may pass all given tests, but break other functionality
COMPUTE THE MEDIAN OF THREE NUMBERS

```c
int median(int a, int b, int c) {
    int result = 0;
    if ((b<=a && a<=c) ||
        (c<=a && a<=b))
        result = a;
    if ((a<b && b<=c) ||
        (c<=b && b<a))
        result = b;
    if ((a<c && c<b) ||
        (b<c && c<a))
        result = c;
    return result;
}
```
```c
int median(int a, int b, int c) {
    int result = 0;
    if ((b<=a && a<=c) ||
        (c<=a && a<=b))
        result = a;
    if ((a<b && b<=c) ||
        (c<=b && b<=a))
        result = b;
    if ((a<c && c<=b) ||
        (b<=c && c<=a))
        result = c;
    return result;
}
```

```c
int med_broken(int a, int b, int c) {
    int result;
    if ((a==b) || (a==c) ||
        (b<=a && a<=c) ||
        (c<=a && a<=b))
        result = a;
    else if ((b==c) || (a<b && b<=c) ||
              (c<b && b<=a))
        result = b;
    else if (a<c && c<b)
        result = c;
    return result;
}
```
int medBroken(int a, int b, int c) {
    int result;
    if ((a == b) || (a == c) ||
        (b < a && a < c))
        result = a;
    else if ((b == c) || (a < b && b < c) ||
             (c < b && b < a))
        result = b;
    else if (a < c && c < b)
        result = c;
    return result;
}

int medBroken(int a, int b, int c) {
    int result;
    if ((a == b) || (a == c) ||
        (b < a && a < c))
        result = a;
    else if ((b == c) || (a < b && b < c) ||
             (c < b && b < a))
        result = b;
    else if (a < c && c < b)
        result = c;
    return result;
}

int medBroken(int a, int b, int c) {
    int result;
    if ((a == b) || (a == c) ||
        (b < a && a < c))
        result = a;
    else if ((b == c) || (a < b && b < c) ||
             (c < b && b < a))
        result = b;
    else if (a < c && c < b)
        result = c;
    return result;
}

int medBroken(int a, int b, int c) {
    int result;
    if ((a == b) || (a == c) ||
        (b < a && a < c))
        result = a;
    else if ((b == c) || (a < b && b < c) ||
             (c < b && b < a))
        result = b;
    else if (a < c && c < b)
        result = c;
    return result;
}
**Potential solution**

- Use an independent test suite to measure quality of the patch.

**Focus of prior evaluations**

- Most evaluations are interested in whether tools work — produce patches.
- Some interest in other factors — human acceptance of patches, plausibility.
- ...but these don't fully assess functional correctness.
- No evaluations test functional correctness of repair outputs independently of repair inputs.

**What do we need?**

- We need bugs with 2 test suites — and the test suites need to be good.
- It's hard enough to find one good test suite, good luck finding programs with two.

**Why?**

Some programs fail some wb tests, others bb tests, others, some of both.

**Make your own!**

[http://repairbenchmarks.cs.umass.edu](http://repairbenchmarks.cs.umass.edu)

- 998 student-written buggy C programs
- Simple (very small)
- Have 2 test suites
  - White-box (generated by KLEE)
  - Black-box (written by instructor)

**RQ1**: What is the base incidence of overfitting?

Give a repair tool the buggy program and the black-box test suite, try to repair it, see what fraction of the white-box tests the patches pass.

**RQ1**: What is the base incidence of overfitting?

But first, how often can we actually generate patches?

<table>
<thead>
<tr>
<th>Repair Tool</th>
<th>Patch Production %</th>
</tr>
</thead>
<tbody>
<tr>
<td>GenProg</td>
<td>466/778 = 59.9%</td>
</tr>
<tr>
<td>TrpAutoRepair</td>
<td>444/778 = 57.1%</td>
</tr>
</tbody>
</table>
RQ1: What is the base incidence of overfitting?

RQ2: What effect do pre-repair test failures have on overfitting?

RQ3: What effect does test suite coverage have on overfitting?

RQ4: What effect does test suite provenance have on overfitting?

- So far, all experiments have used human-written black-box tests to build repairs
- Switch to using KLEE-generated white-box tests
- Attempt to repair programs
- If a repair is found, measure correctness of repair – this time with black-box tests
RQ4: What effect does test suite provenance have on overfitting?

Automatically generated tests produced significantly buggier repairs compared to human-written tests.

RQ4: Do tools do better than novices?

Summary

- Overfitting is a real concern
  - median patch for either tool passed only 75% of evaluation suite
- Overfitting is hard to avoid
  - minimization doesn’t help on this dataset
  - N-version voting only works in extreme cases
- Program repair is harder for buggier programs, but likely to break more correct programs
- Novice developers don’t significantly beat repair tools

So is there no hope?

- SearchRepair, a brand new technique, reduces overfitting to 97.2%.
- Most SearchRepair repairs pass 100% of the held-out test suite. (Select few poor repairs drop the overall rate.)

Read more about SearchRepair: