Repairing Automated Repair

Course updates

- Literature review was due today
- Project plan assignment is posted, due April 11
- Homework 3 due this Thursday, 9 AM EDT
- Homework 4 in posted, due April 6

Generalizing

- How many games are there in a 78-team bracket?
- What about an n-team bracket?

Repairing Automated Repair

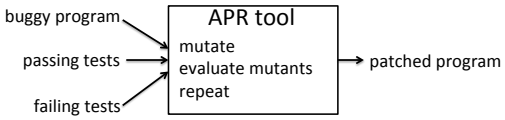
Cobra effect

What do cobras have to do with automated program repair?

repairing python programs?

Automated Program Repair

basic idea:



the many repair tools

ClearView [Perkinds et al. 2009] GenProg [Weimer et al. 2009]

Prophet [Long and Rinard 2015] SPR [Long and Rinard 2015]

TDS [Perelman et al. 2014]

Par [Kim et al. 2013] AE [Weimer et al. 2013]

SemFix [Nguyen et al. 2013] AutoFix-E [Wei et al. 2010]

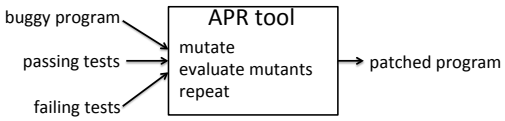
[Carzaniga et al. 2010] [Carzaniga et al. 2013]

[Jin et al. 2011] Coker and Hafiz et al. 2013]

[Debroy and Wong et al. 2010] [Lin and Ernst et al. 2004]

[Forrest et al. 2009] [Novark et al. 2007] [Demsky et al. 2006]

Potential problem



the patched program may pass all given tests, but break other functionality

COMPUTE THE MEDIAN OF THREE NUMBERS

```
int median(int a, int b, int c) {
  int result;
  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;
}
```

```
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;
}
```

```
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;
}
```

```
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;
}
```

```
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;
}
```

```
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;
}
```

```
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;
}
```

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

```
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 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 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 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;
}
```

Input	Expected	Pass?
0,0,0	0	✓
2,0,1	1	X
0,0,1	0	✓
0,1,0	0	✓
0,2,1	1	✓
0,2,3	2	✓

```
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;
  if (b < a)
    result = c;
  else if (b<a) (b==c) || (a<b && b<c) ||
             (c<b && b<a))
    result = b;
  else if (a<c && c<b)
    result = c;
  return result;
}
```

Input	Expected	Pass?
0,0,0	0	✓
2,0,1	1	X
0,0,1	0	✓
0,1,0	0	✓
0,2,1	1	✓
0,2,3	2	✓

```
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;
  if (b < a)
    result = c;
  if (b<a) (b==c) || (a<b && b<c) ||
             (c<b && b<a))
    result = b;
  if (a<c && c<b)
    result = c;
  return result;
}
```

Input	Expected	Pass?
0,0,0	0	✓
2,0,1	1	X
0,0,1	0	✓
0,1,0	0	✓
0,2,1	1	✓
0,2,3	2	✓

```
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;
  if (b < a)
    result = c;
  else if (b<a) (b==c) || (a<b && b<c) ||
             (c<b && b<a))
    result = b;
  else if (a<c && c<b)
    result = c;
  return result;
}
```

Input	Expected	Pass?
0,0,0	0	✓
2,0,1	1	✓
0,0,1	0	✓
0,1,0	0	✓
0,2,1	1	✓
0,2,3	2	✓

```
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;
  if ((b==c) || (a<b && b<c) ||
             (c<b && b<a))
    result = b;
  if (a<c && c<b)
    result = c;
  return result;
}
```

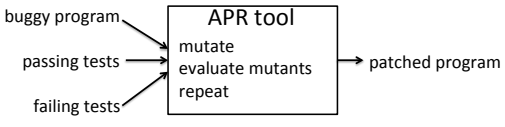
Input	Expected	Pass?
2,6,8	6	✓
2,8,6	6	✓
6,2,8	6	✓
6,8,2	6	✓
8,2,6	6	X
8,6,2	6	✓
9,9,9	9	✓

```
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;
  if (b < a)
    result = c;
  else if (b<a) (b==c) || (a<b && b<c) ||
             (c<b && b<a))
    result = b;
  else if (a<c && c<b)
    result = c;
  return result;
}
```

Input	Expected	Pass?
0,0,0	0	✓
2,0,1	1	✓
0,0,1	0	✓
0,1,0	0	✓
0,2,1	1	✓
0,2,3	2	✓

Input	Expected	Pass?
2,6,8	6	✓
2,8,6	6	✓
6,2,8	6	X
6,8,2	6	✓
8,2,6	6	✓
8,6,2	6	X
9,9,9	9	✓

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 [Durieux et al. 2015] [Fry et al. 2012] [Kim et al. 2013]
  - plausibility [Qi et al. 2015]
  - …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

Why?

- it's hard enough to find one good test suite, good luck finding programs with two

Make your own!

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)

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

RQ1: What is the base incidence of overfitting?

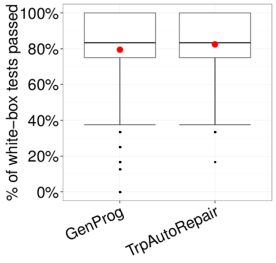
Give a repair tool the buggy program and the black-box test suite, try to repair it, see what fraction of the white-box tests the patches pass.

RQ1: What is the base incidence of overfitting?

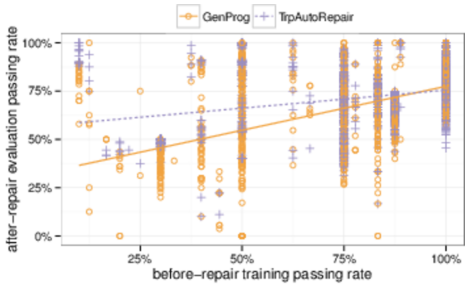
but first, how often can we actually generate patches?

repair tool	patch production %
GenProg	466/778 = 59.9%
TrpAutoRepair	444/778 = 57.1%

RQ1: What is the base incidence of overfitting?

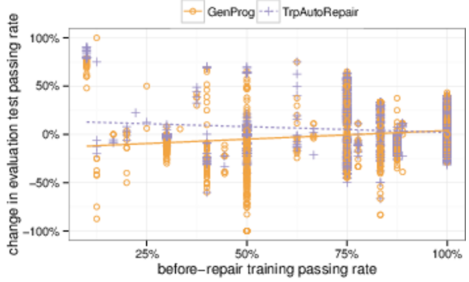


RQ2: What effect do pre-repair test failures have on overfitting?



Programs that fail more tests before repair still fail more tests after repair

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

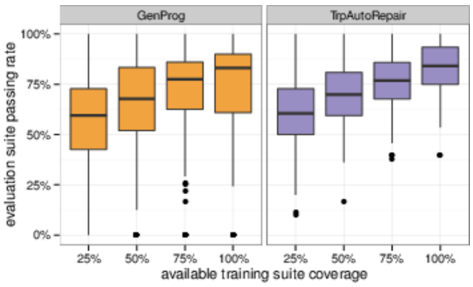


Repair is at best unlikely to improve correctness, at worst likely to worsen it

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

- Randomly sample 25%, 50%, and 75% of passing and failing tests for each buggy program
- Attempt to repair programs
  - with each level of test coverage
- If a repair is found, measure correctness of repair

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

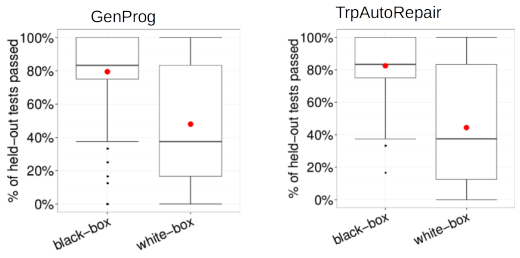


Lower test suite coverage leads to more overfitting

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

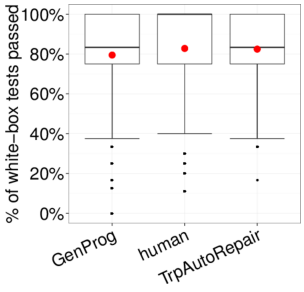
- So far, all experiments have used human-written *black-box* tests to build repairs
- Switch to using KLEE-generated *white-box* tests
- Attempt to repair programs
- If a repair is found, measure correctness of repair
  - this time with *black-box* tests

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:

http://people.cs.umass.edu/~brun/pubs/pubs/Ke15ase.pdf