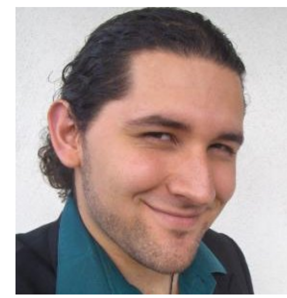
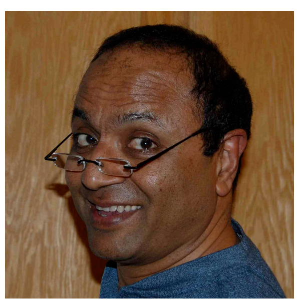
Measuring and improving quality of automated program repair

Yuriy Brun, UMass Amherst



Ted Smith Yalin Ke Manish Motwani Mauricio Soto



Earl Barr Prem Devanbu Claire Le Goues René Just Katie Stolee

Automated Program Repair

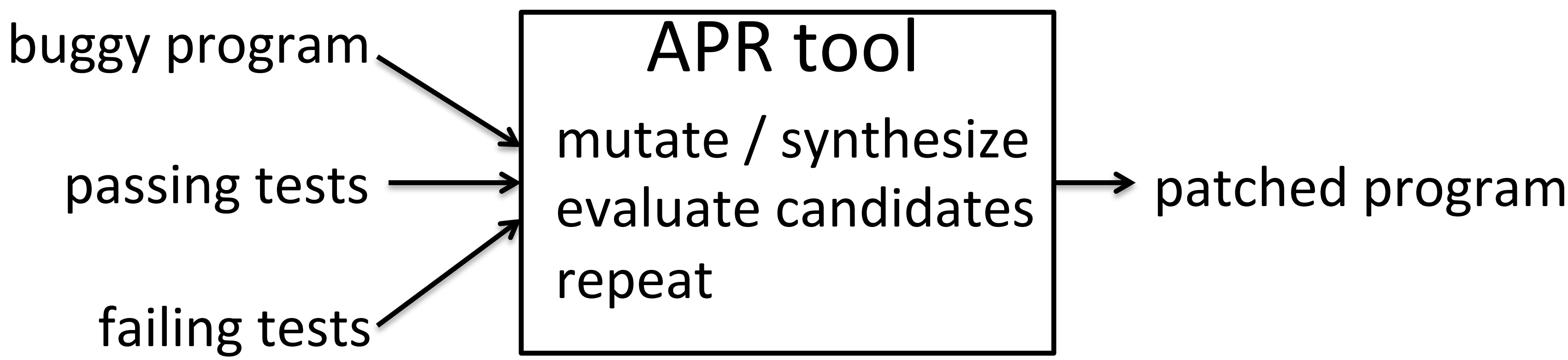
- Given a software system with a bug
  - (typically) a set of passing and a set of failing tests –

  produce a variant of that software system without the bug.

Given a system S that passes tests T_p and fails tests T_f, automatically produce S' that passes T_p and T_f

Exploration-based approaches

basic idea:



the many exploration-based repair tools

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

Prophet [Long and Rinard 2016] 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]

The automatic program repair story

- Early papers asked:
  - What fraction of bugs can APR fix?
  - How long does it take APR to fix bugs?
  - How much does it cost for APR to fix bugs?
  - Can humans maintain APR fixes?

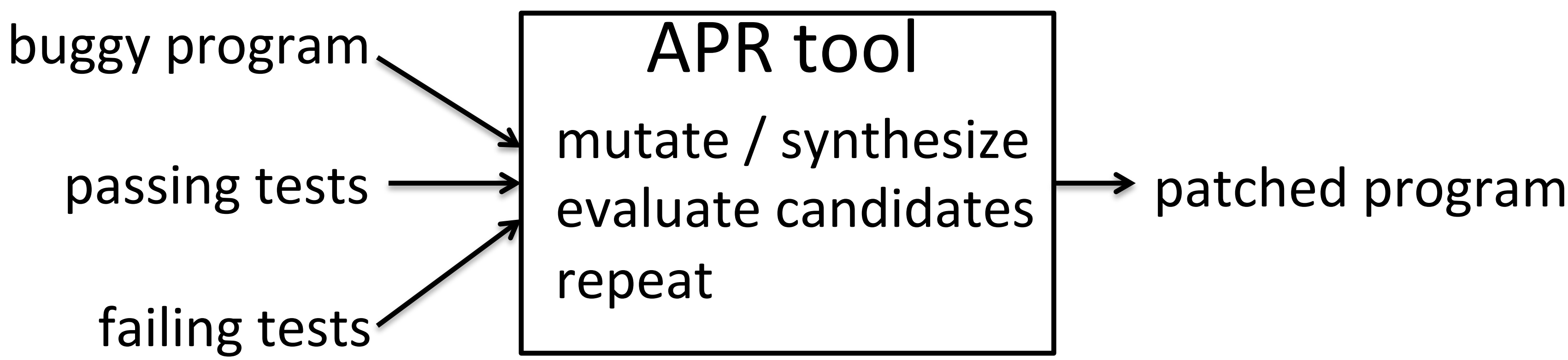
The story was, APR produces a patch that passes all tests

implies

problem solved

Cobra effect

Does exploration-based repair repair?



The patch may break untested or under-tested functionality

How can we know if APR repairs

- Look at the produced patches by hand
  [Qi, Long, Achour, Rinard, ISSTA 2015]
  [Durieux, Martinez, Monperrus, Sommerard, Xuan, 2015]
- Have others look at the produced patches by hand
  [Fry, Landau, Weimer, ISSTA 2012]
  [Kim, Nam, Song, Kim, ICSE 2013]

- Produce patches with test suite T,
  evaluate them on independent test suite T'
  [Brun, Barr, Xiao, Le Goues, Devanbu, 2013]
  [Smith, Barr, Le Goues, Brun, ESEC/FSE 2015]
  - objective
  - repeatable

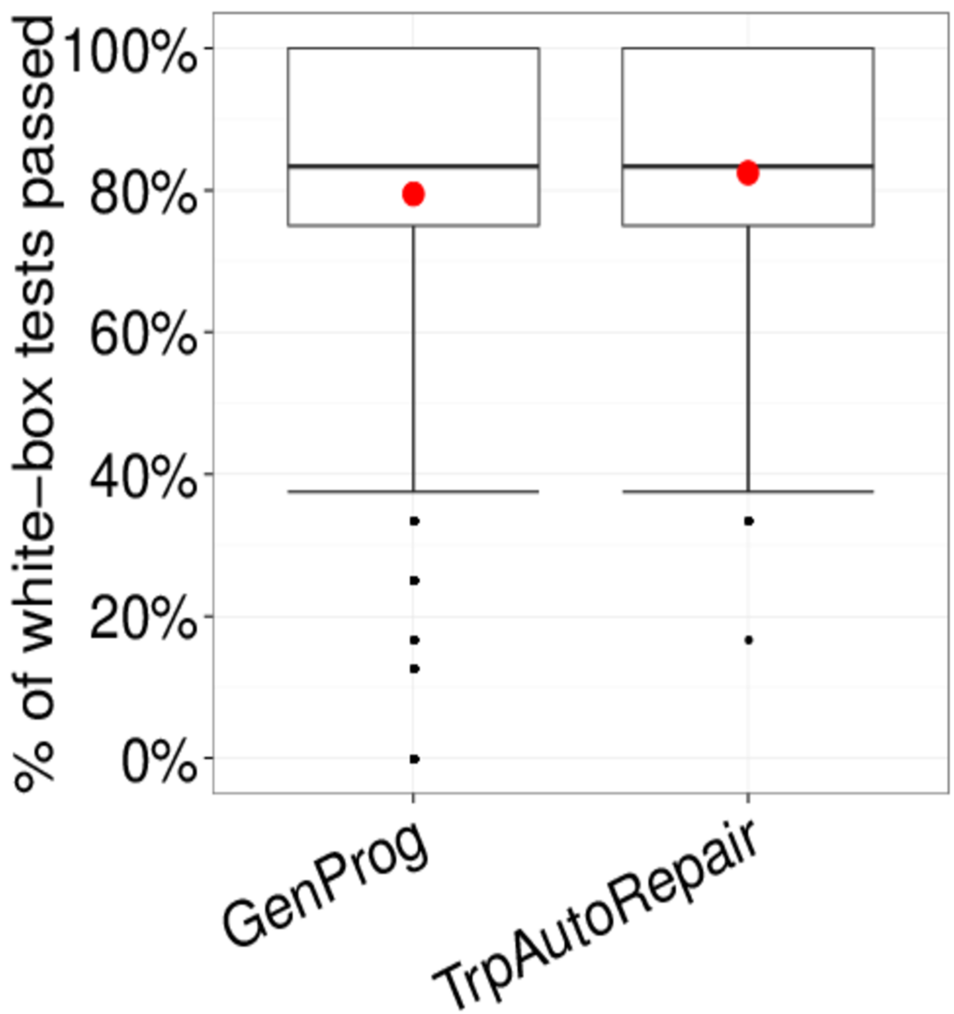
IntroClass Benchmark

Requires a large set of bugs
for programs with 2 independent test suites
and the test suites need to be good

- IntroClass:
  998 bugs in very small, student-written C programs,
  with a KLEE-generated test suite,
  and a human-written test suite.

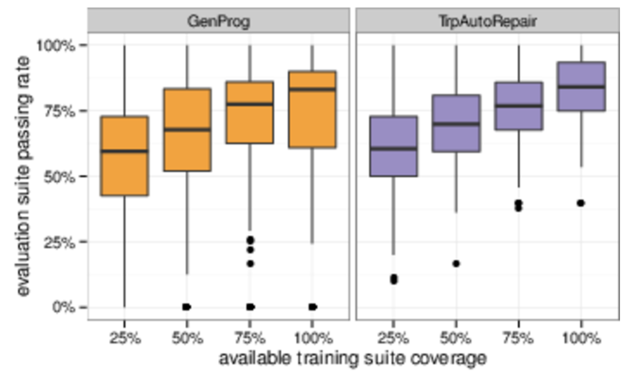
- http://repairbenchmarks.cs.umass.edu, [TSE 2015]

Do GenProg and TrpAutoRepair patches pass kept-out tests?



More GenProg and TrpAutoRepair findings

- The better the test suite coverage, the better the patch



- APR causes harm to high-quality programs, but is helpful for low-quality programs
- Human-written tests lead to better patches
- Student-written patches also break tests.

More answers and details in
"Is the Cure Worse Than the Disease? Overfitting in Automated Program Repair"
by Smith, Barr, Le Goues, Brun, ESEC/FSE 2015

Can we improve the patch quality?

- Recent work:
  - SPR [Long and Rinard, ESEC/FSE 2015]
  - Prophet [Long and Rinard, POPL 2016]
- Both SPR and Prophet produce more correct patches than GenProg, TrpAutoRepair, AE

- My vision: repair at a higher level

SearchRepair: Use existing code

Replace whole code blocks with code from other projects (e.g., GitHub)

Imagine a program with a buggy sort method:

Option 1

Mutate, synthesize, and tweak the sort method until a set of sorting tests pass

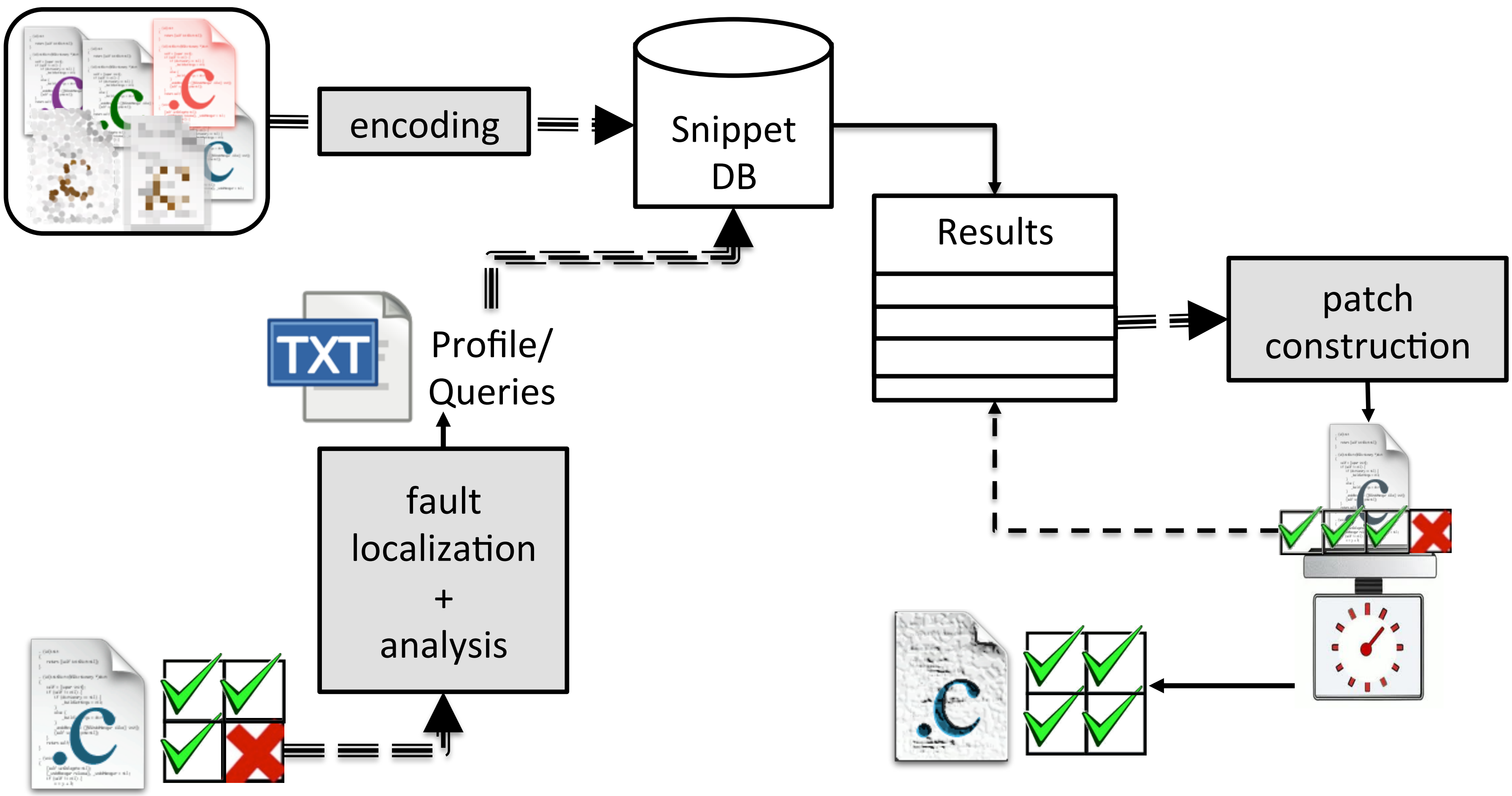
Option 2

Find a method on GitHub that passes the sorting tests

"Repairing Programs with Semantic Code Search" by Ke, Stolee, Le Goues, Brun, ASE 2015

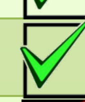
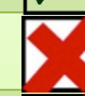
SearchRepair: Use existing code

Replace whole code blocks with code from other projects (e.g., GitHub)



Example: median

```c
int main() {
 int a, b, c, median = 0;
 printf("Please enter 3 numbers separated by spaces >");
 scanf("%d%d%d", &a, &b, &c);
 if ((a<=b && a>=c) || (a>=b && a<=c))
     median = a;
 else if ((b<=a && b>=c) || (b>=a && b<=c))
     median = b;
 else if ((c<=b && a>=c) || (c>=b && a<=c))
     median = c;
 printf("%d is the median", median);
 return 0;
}
```

test	input	test result
t_1	9 9 9	
t_2	0 2 3	
t_3	0 1 0	
t_4	2 0 1	
t_5	2 8 6	

Encoding

Given snippets of code, automatically compute the SMT constraints between snippet inputs and outputs. Store in DB.

```
if((x <= y && x >= z) || (x >= y && x <=z))
    m = x;
else if((y <=  x && y >= z) || (y >= x && y <= z))
    m = y;
else
    m = z;
```

```
vars:  LOCAL(int x, int y, int z, int m)
  p1:  ASSUME[(x <= y && x >= z)|| (x >= y && x <=z)]
       STMT[m = x]
  p2:  ASSUME[not((x <= y && x >= z)|| (x >= y && x <=z))
            && ((y <= x && y >= z)|| (y >= x && y <= z))]
       STMT[m = y]
  p3:  ASSUME[not((x <= y && x >= z)|| (x >= y && x <=z))
            && not((y <= x && y >= z)|| (y >= x && y <= z))]
       STMT[m = z]
```

Fault localization

Identify the code lines that execute more often on failing tests, the elevate these lines to block level.

```c
int main() {
 int a, b, c, median = 0;
 printf("Please enter 3 numbers separated by spaces >");
 scanf("%d%d%d", &a, &b, &c);
 if ((a<=b && a>=c) || (a>=b && a<=c))
    median = a;
 else if ((b<=a && b>=c) || (b>=a && b<=c))
    median = b;
 else if ((c<=b && a>=c) || (c>=b && a<=c))
    median = c;
 printf("%d is the median", median);
 return 0;
}
```

Semantic search and context

Identify input-output behavior on passing tests, and use SMT solver to find satisfying snippets in DB (potential patches).

```c
int main() {
 int a, b, c, median = 0;
 printf("Please enter 3 numbers separated by spaces >");
 scanf("%d%d%d", &a, &b, &c);
 if ((a<=b && a>=c) || (a>=b && a<=c))
    median = a;
 else if ((b<=a && b>=c) || (b>=a && b<=c))
    median = b;
 else if ((c<=b && a>=c) || (c>=b && a<=c))
    median = c;
 printf("%d is the median", median);
 return 0;
}
```

```c
if((x <= y && x >= z) || (x >= y && x <=z))
    m = x;
else if((y <=  x && y >= z) || (y >= x && y <= z))
    m = y;
else
    m = z;
```

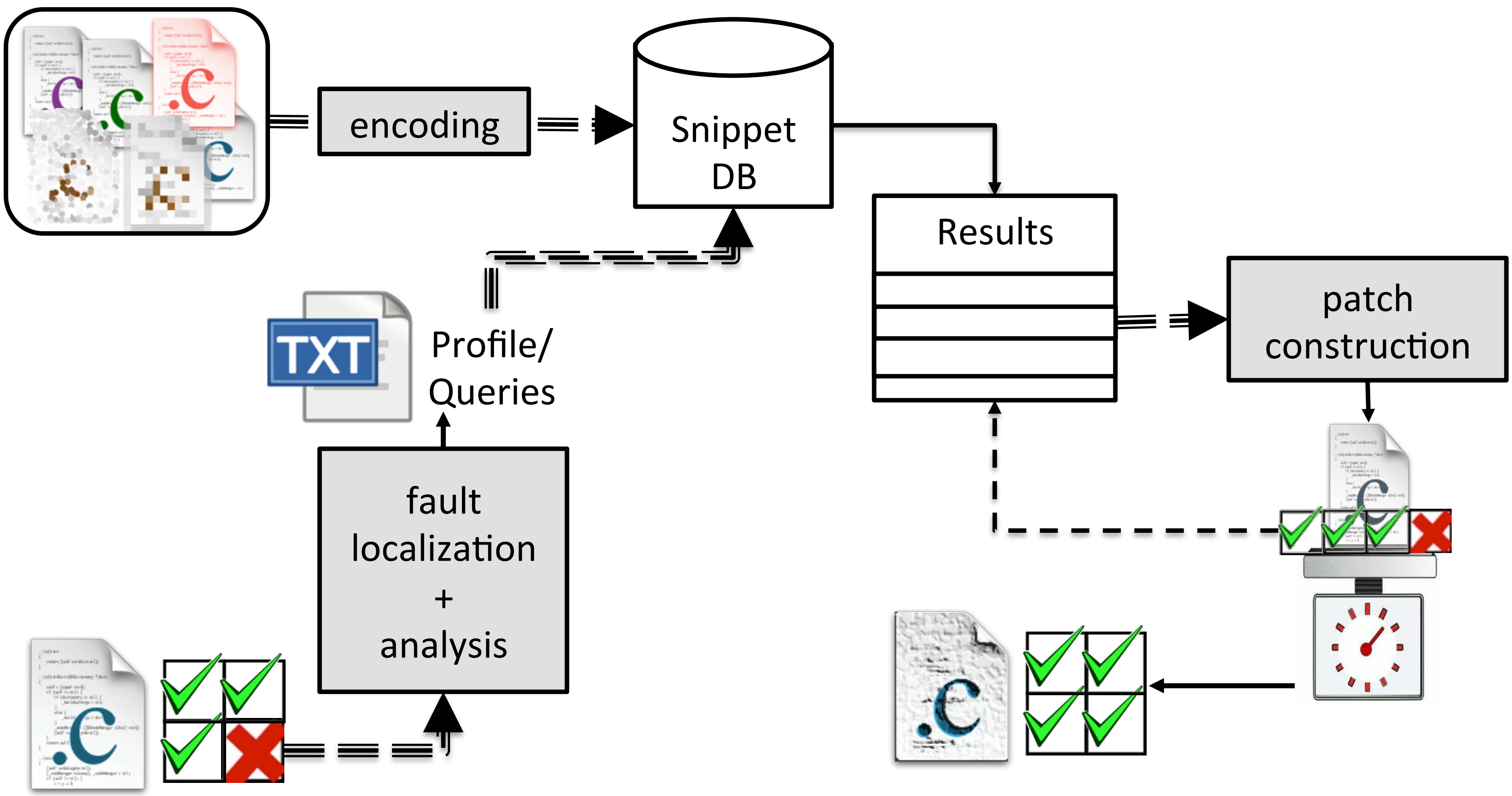
Barr, Harman, Jia, Marginean, Petke, ISSTA 2015 could enable larger-scale transplantation

Validate potential patches

Rerun tests to select patches that repair the bug.

SearchRepair: Use existing code

- Replace whole code blocks with code from other projects (e.g., GitHub)



SearchRepair vs. Exploration

% of kept-out tests patches pass

SearchRepair	GenProg	TRPAutoRepair	AE
97.2%	68.7%	72.1%	64.2%

"Repairing Programs with Semantic Code Search" by Ke, Stolee, Le Goues, Brun, ASE 2015

Contributions

- Repeatable, automated, objective methodology for evaluating automated repair quality
  - including the IntroClass dataset
- SearchRepair: semantic-search-based repair
- A small-scale prototype of SearchRepair, evaluated on IntroClass
  - greatly improves repair quality over GenProg, TrpAutoRepair, and AE