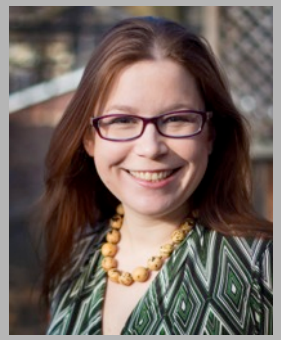
# REPAIRING PROGRAMS WITH SEMANTIC CODE SEARCH


Yalin Ke

Iowa State

Kathryn T. Stolee

Iowa State

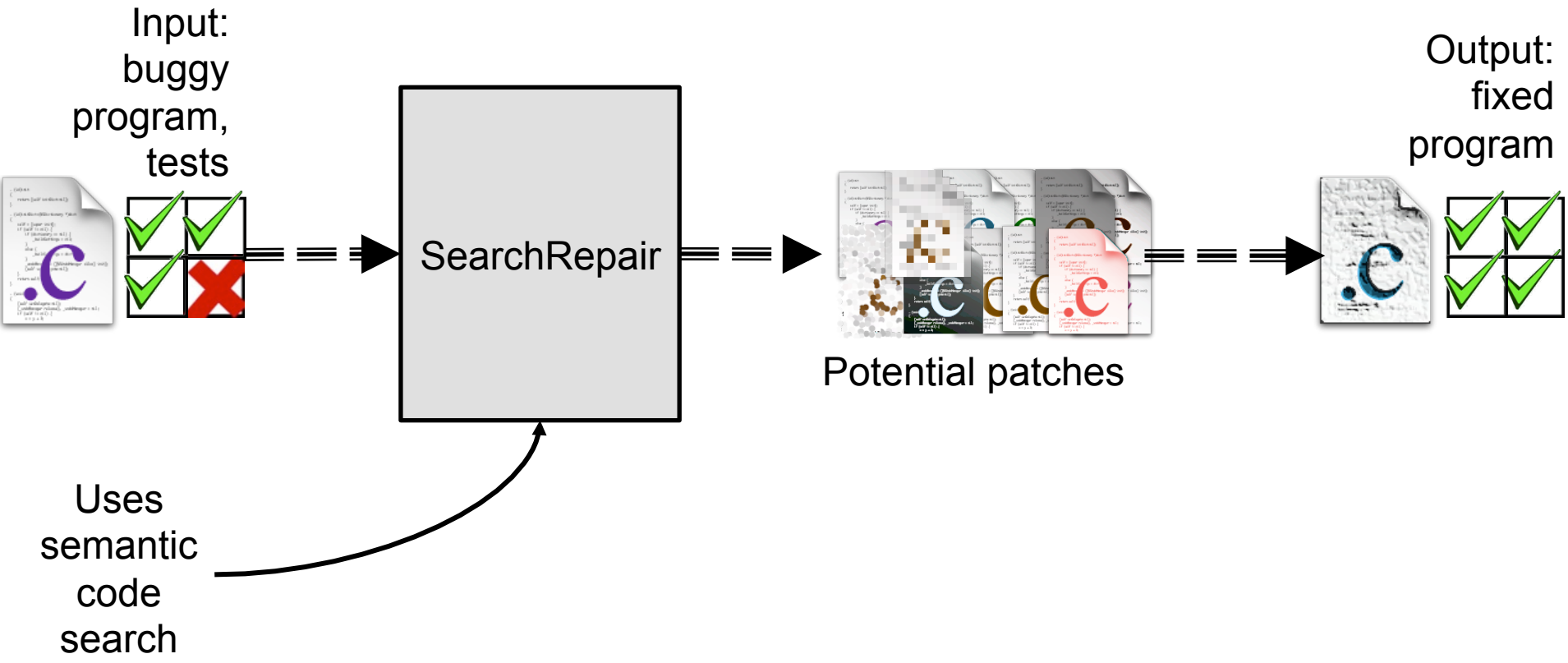


Claire Le Goues

Carnegie Mellon

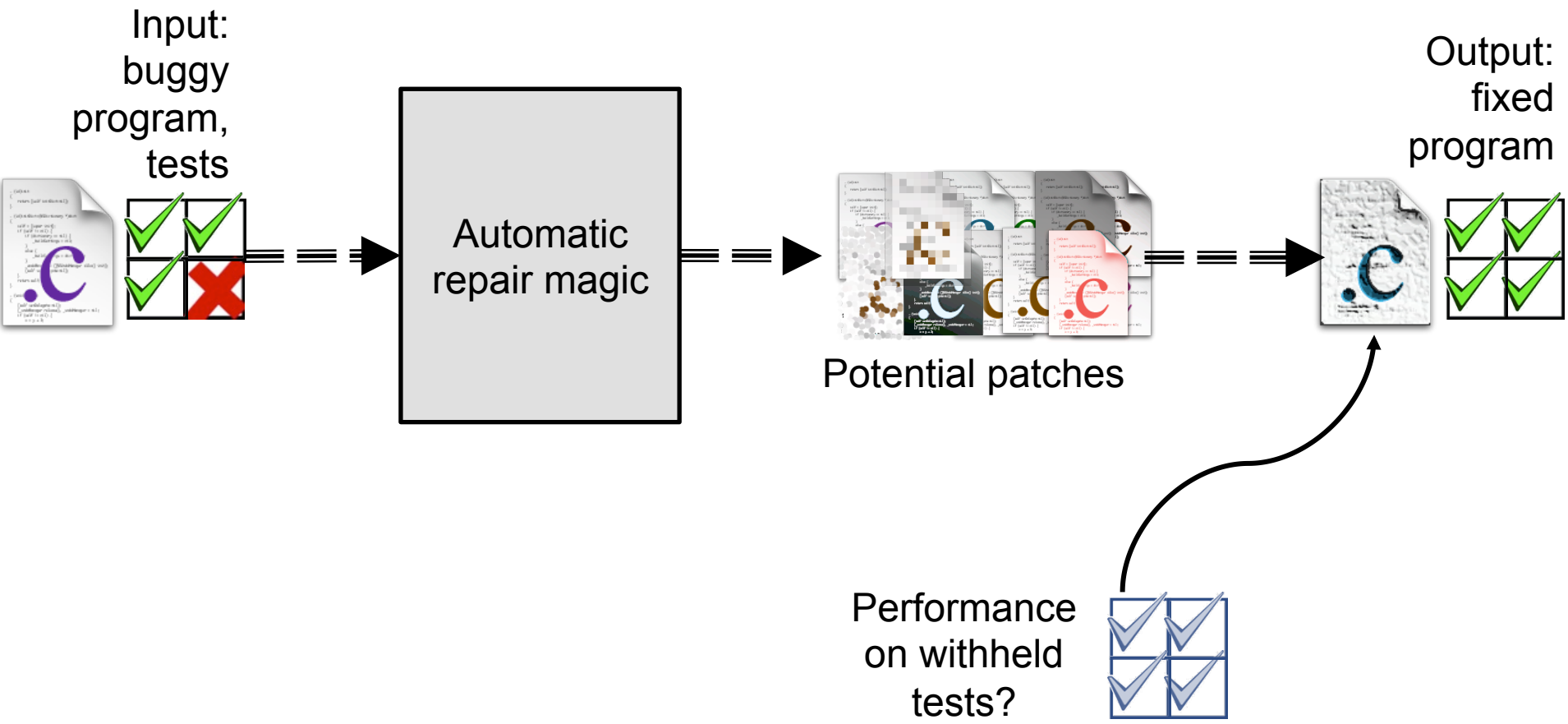
Yuriy Brun

UMass Amherst


Input:
buggy
program,
tests
SearchRepair
Potential patches
Output:
fixed
program
Uses
semantic
code
search

PROBLEM

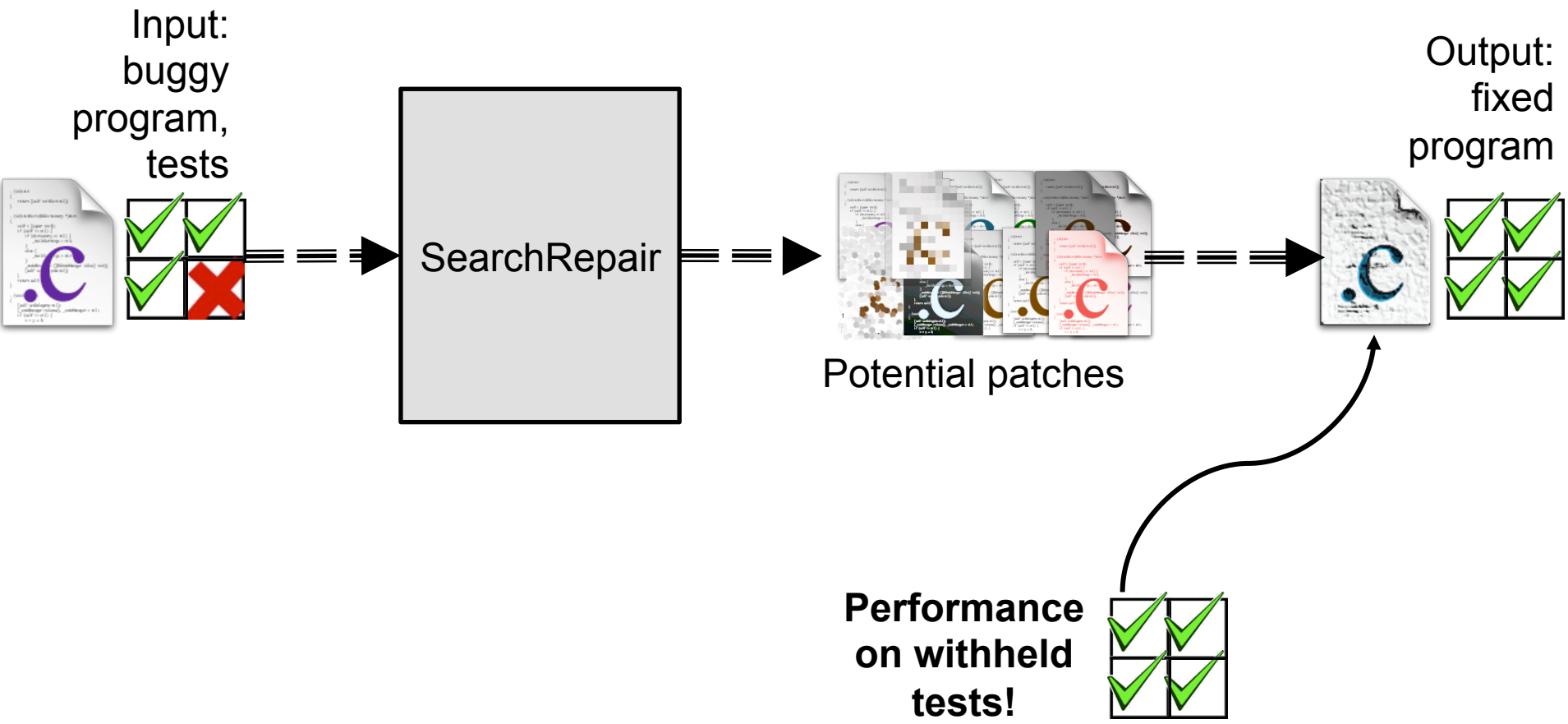
# PATCH QUALITY


Input:
buggy
program,
tests
Automatic
repair magic
Potential patches
Output:
fixed
program
Performance
on withheld
tests?

# OVERFITTING

**Does the patch *generalize* beyond the test cases used to create it?**

Edward K. Smith, Earl Barr, Claire Le Goues, and Yuriy Brun, Is the Cure Worse than the Disease? Overfitting in Automated Program Repair, ESEC/FSE 2015.


Input:
buggy
program,
tests
SearchRepair
Potential patches
Output:
fixed
program
Performance
on withheld
tests!

# 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 | ✓ |

Search

median three numbers

Search

Repositories

Code 25,815

Issues 10

Users

## Languages

C

| | |
|---|---|
| Text | 19,500 |
| HTML | 17,252 |
| PHP | 9,448 |
| XML | 8,554 |
| JavaScript | 8,416 |
| C++ | 4,583 |
| Python | 4,508 |
| TeX | 3,871 |
| Gettext Catalog | 3,207 |

Advanced search Cheat sheet

## We've found 25,815 code results

Sort: **Best match**

**canadaduane/winter09 – median.h** C

Showing the top eight matches. Last indexed on Sep 26.

```
1  #ifndef MEDIAN_H
2  #define MEDIAN_H
3
4  typedef struct ARRAY {
5      int* ptr;
6      int size;
7  } Array;
8
9  int median( Array numbers );
10 int median_of_first( Array numbers );
11 int median_of_three( Array numbers );
12 int median_random( Array numbers );
13
14 #endif
```

**dalkire/CModernApproach – 09e15.c** C

Showing the top four matches. Last indexed 27 days ago.

```
4  * The following (rather confusing) function finds the median of three numbers.
5  * Rewrite the function so that it has just one return statement.
6  * double median(double x, double y, double z)
7  * {
8  *     if (x <= y)
9  *         if (y <= z) return y;
```

**luctheduke/LC-CS171 – median.c** C

Showing the top four matches. Last indexed 27 days ago

# WHAT IF...

**Instead of trying to make small changes, we replaced buggy regions with code that correctly captures the overall desired logic?**

**Principle: using human-written code to fix code at a higher granularity level leads to better quality repairs.**

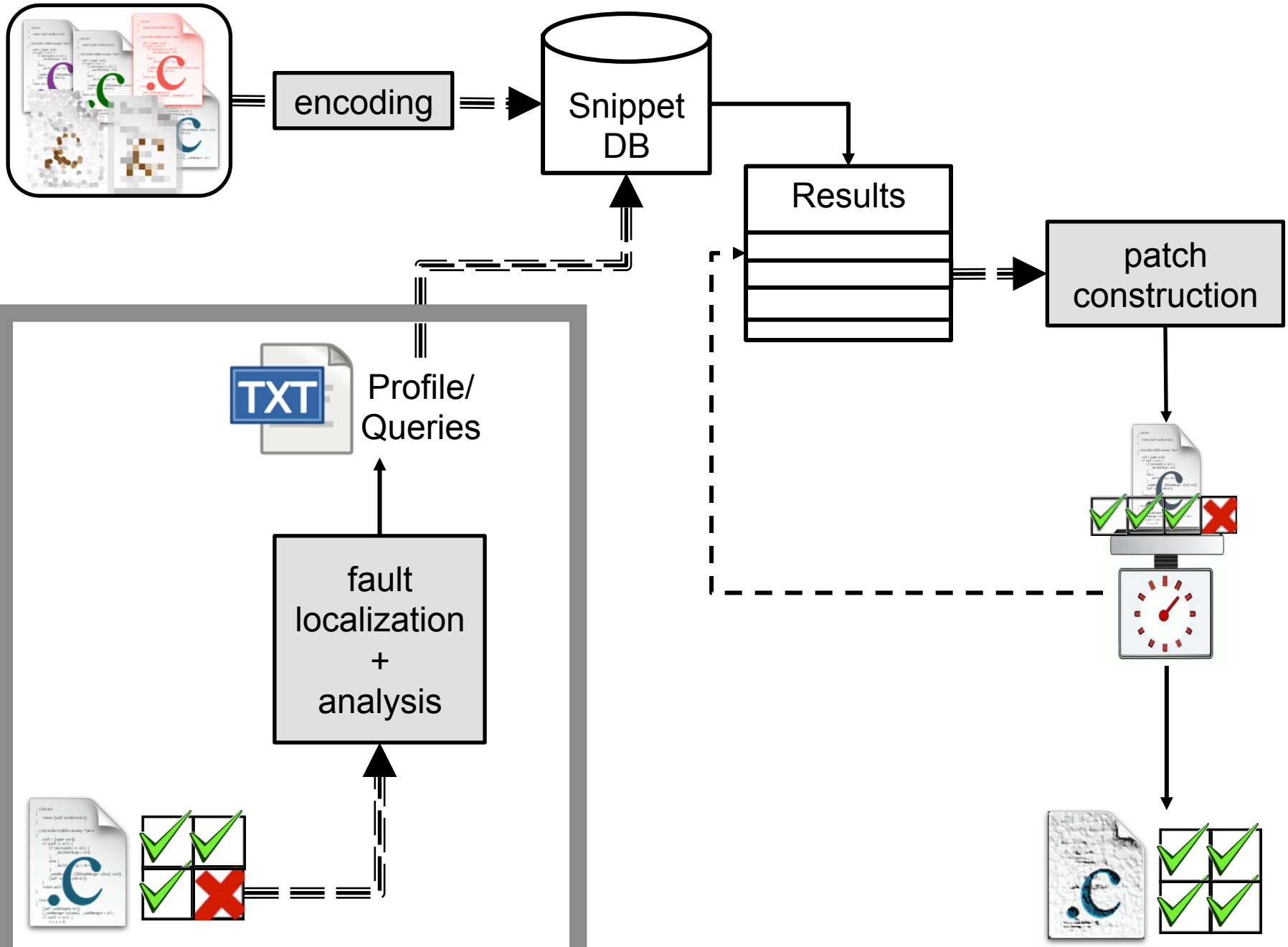
# SearchRepair: THE PLAN

1. Localize bug to a *region.*

2. Create input/output examples that show what the code should do.

3. Use *semantic code search* to find snippets that do the right thing.

4. Construct and test candidate patches for each result from the search.

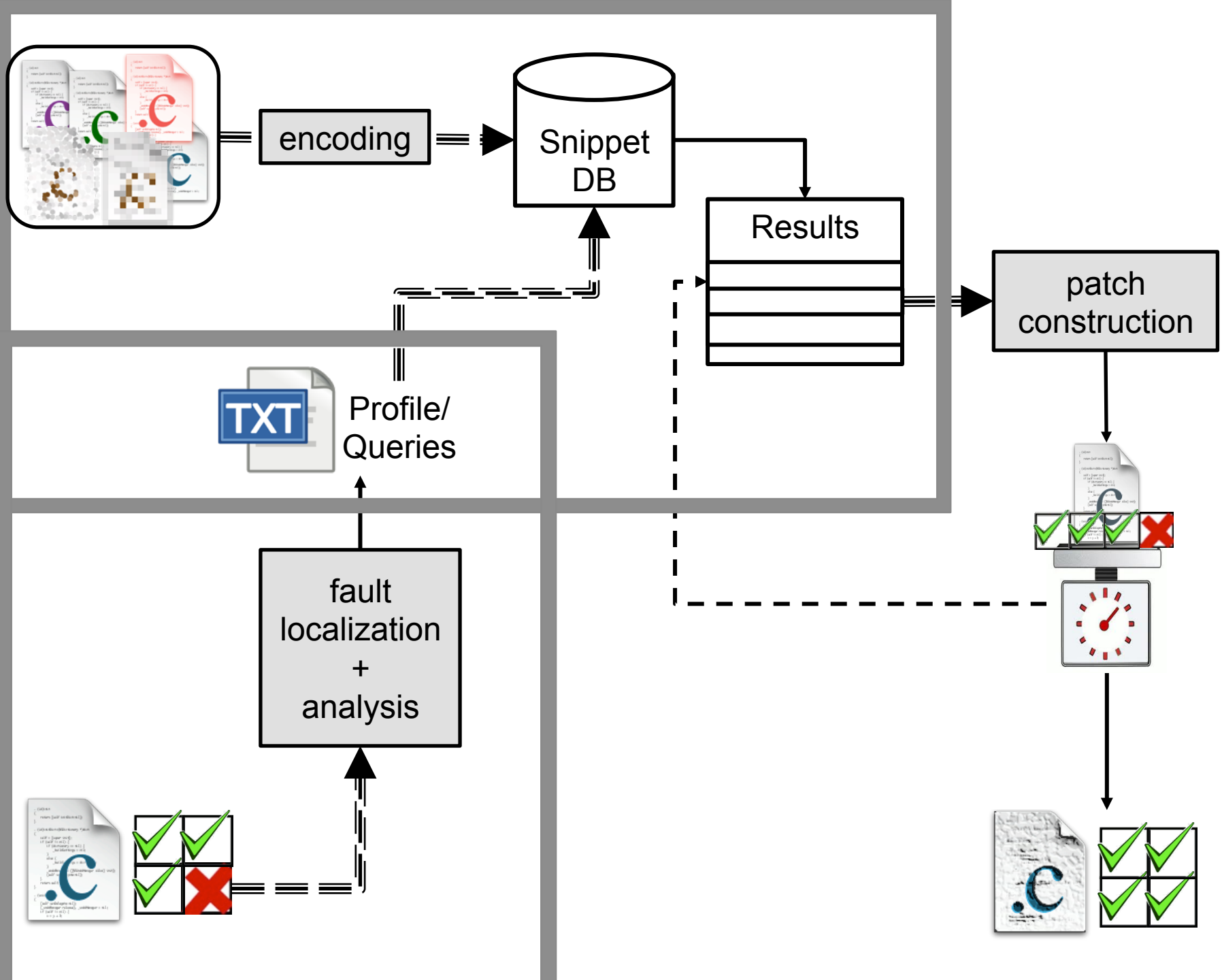

encoding
Snippet DB
Results
patch construction
TXT
Profile/ Queries
fault localization + analysis

# MODIFIED SB-FAULT LOCALIZATION

```
int med_broken(int a, int b, int
  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? |
|---|---|---|
| 6,2,8 | 6 | ✓ |
| 6,8,2 | 6 | ✓ |
| 8,2,6 | 6 | X |
| 8,6,2 | 6 | ✓ |

James A. Jones, Mary Jean Harrold, and John Stasko. Visualization of test information to assist fault localization. ICSE 2002.
M. Gabel and Z. Su. A study of the uniqueness of source code. FSE, 2010.


encoding
Snippet DB
Results
patch construction
TXT
Profile/ Queries
fault localization + analysis

# SEARCHREPAIR: HIGH-QUALITY AUTOMATED BUG REPAIR USING SEMANTIC SEARCH

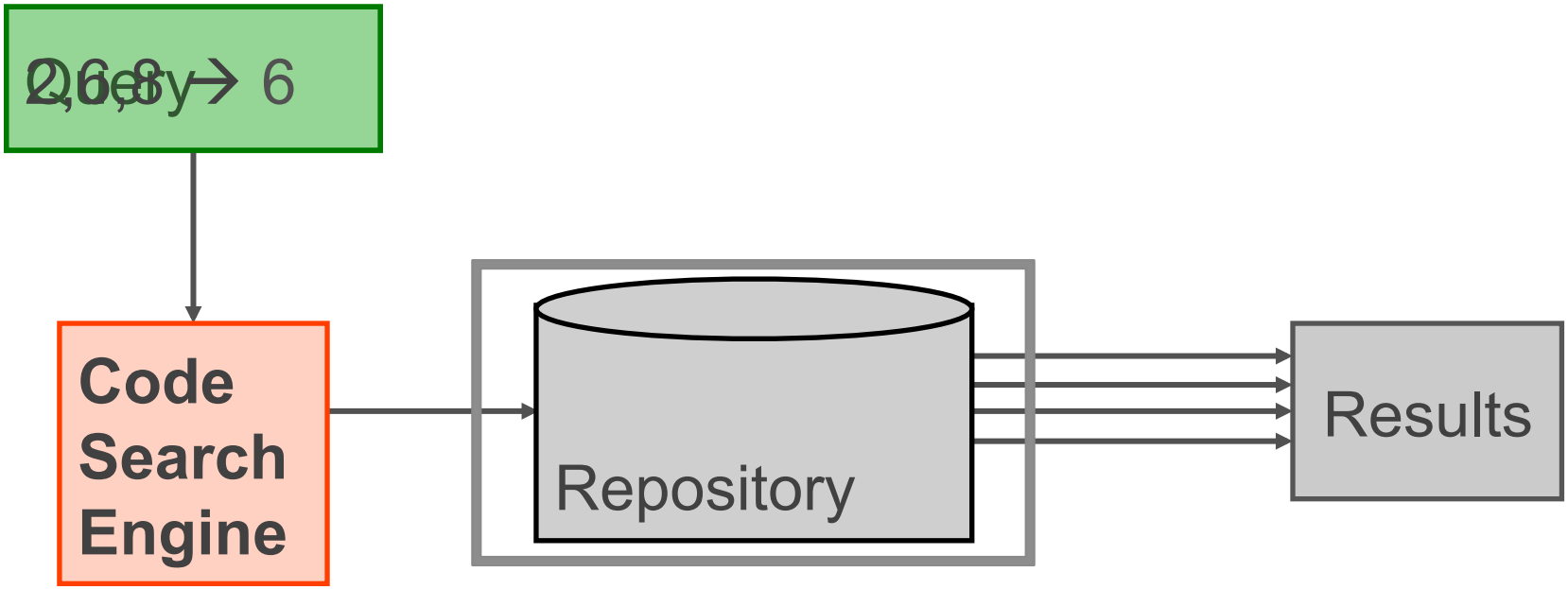
# SEMANTIC CODE SEARCH

**Keyword: “C median three numbers”**

**Semantic:**

| Input | Expected |
|---|---|
| 2,6,8 | 6 |
| 2,8,6 | 6 |
| 6,2,8 | 6 |
| 6,8,2 | 6 |
| 8,6,2 | 6 |
| 9,9,9 | 9 |

*K. T. Stolee, S. Elbaum, M. B. Dwyer, "Code search with input/output queries: Generalizing, ranking, and assessment”, JSS 2015.*
*K. T. Stolee, S. Elbaum, and D. Dobos. 2014. "Solving the Search for Source Code". TOSEM 2014.*
Steven P. Reiss. Semantics-based code search. ICSE, 2009.


Query
Code Search Engine
Repository
Results

$$P_{enc} = ((d > e \wedge d > f \wedge return = d) \vee (d > e \wedge d \leq f \wedge e \leq d \wedge return = f) \vee (d \leq e \wedge e > d \wedge e > f \wedge return = e) \vee (d \leq e \wedge e > d \wedge e \leq f \wedge return = f) \vee (d \leq e \wedge e \leq d \wedge return = f))$$

Repository

```
private int getsum(int a, int b, int c){
  return a + b + c;
}

private boolean allPositive(int x, int y, int z){
  return x >= 0 && y >= 0 && z >= 0;
}

private boolean areEqual(String s, String t) {
  return s.toLowerCase().equals(t.toLowerCase());
}
```

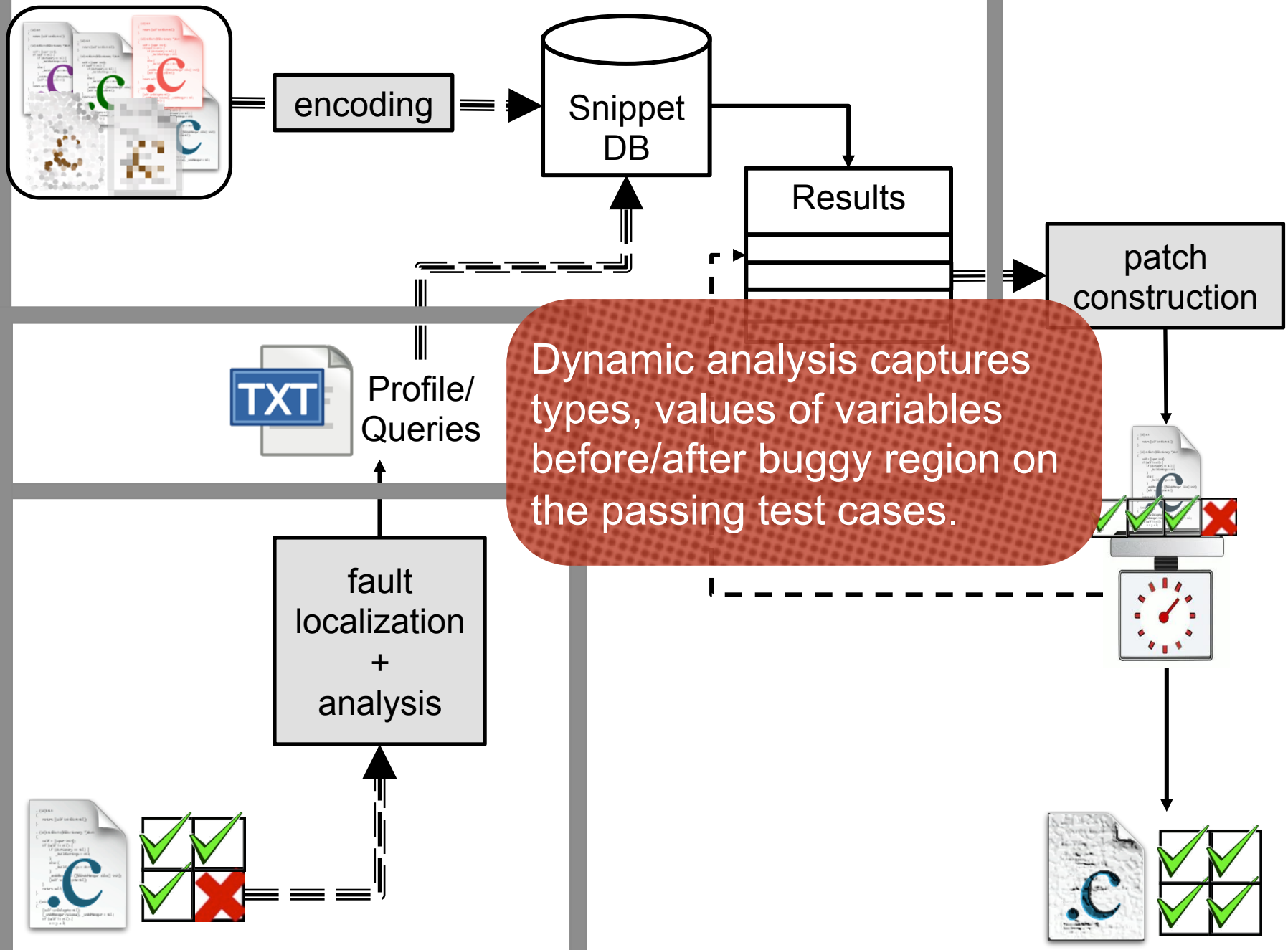
```
private int getMax(int d, int e, int f){
  if(d > e && d > f) {
    return d;
  }else if (e > d && e > f) {
    return e;
  }else {
    return f;
}
```

# SEMANTIC CODE SEARCH

1. **Store candidate snippets as symbolic constraints.**

2. **Search using input/output examples that show what the desired code should do.**

3. **See which symbolic constraints are co-satisfiable with the input/output ~~examples~~ *constraints* (Z3).**

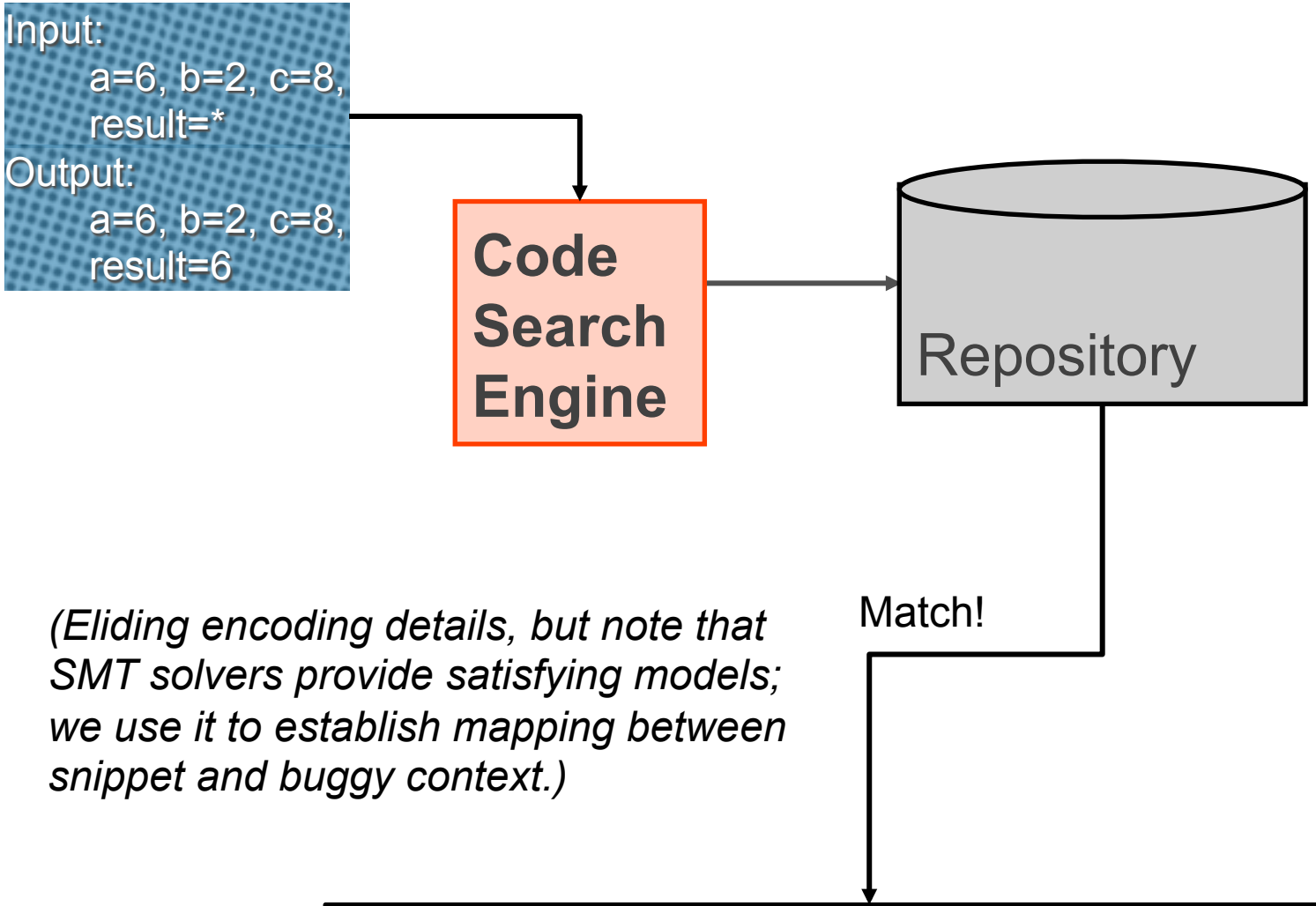

encoding
Snippet DB
Results
patch construction
TXT
Profile/ Queries
Dynamic analysis captures types, values of variables before/after buggy region on the passing test cases.
fault localization + analysis

```
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:
a=6, b=2, c=8,
result=*

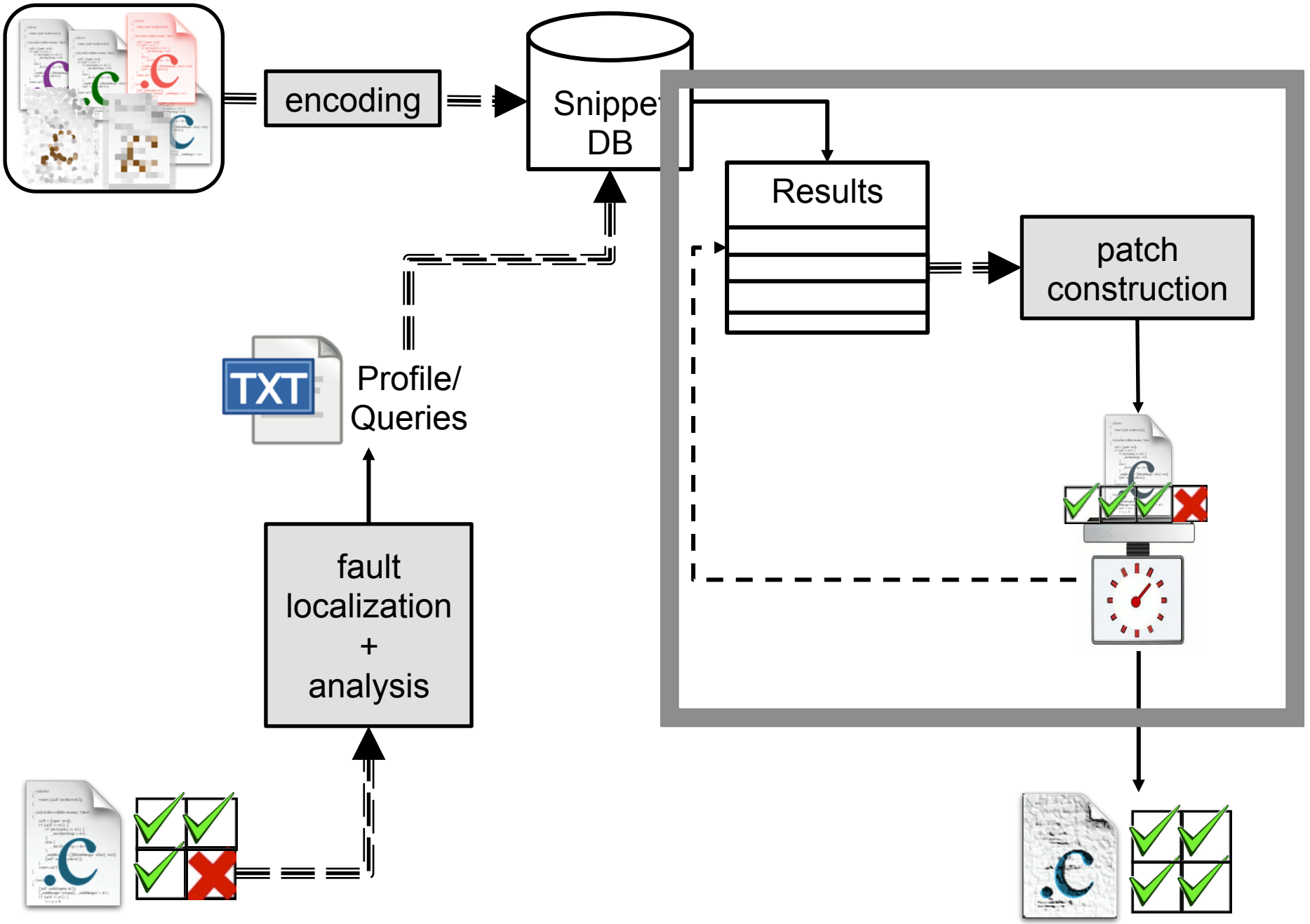
Output:
a=6, b=2, c=8,
result=6

| Input | Expected | Pass? |
|---|---|---|
| 6,2,8 | 6 | ✓ |
| 6,8,2 | 6 | ✓ |
| 8,2,6 | 6 | X |
| 8,6,2 | 6 | ✓ |




*(Eliding encoding details, but note that SMT solvers provide satisfying models; we use it to establish mapping between snippet and buggy context.)*

```
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;
```


encoding
Snippet DB
Results
patch construction
TXT
Profile/ Queries
fault localization + analysis

```
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;
}
```

```
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;
```

```
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;
```

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

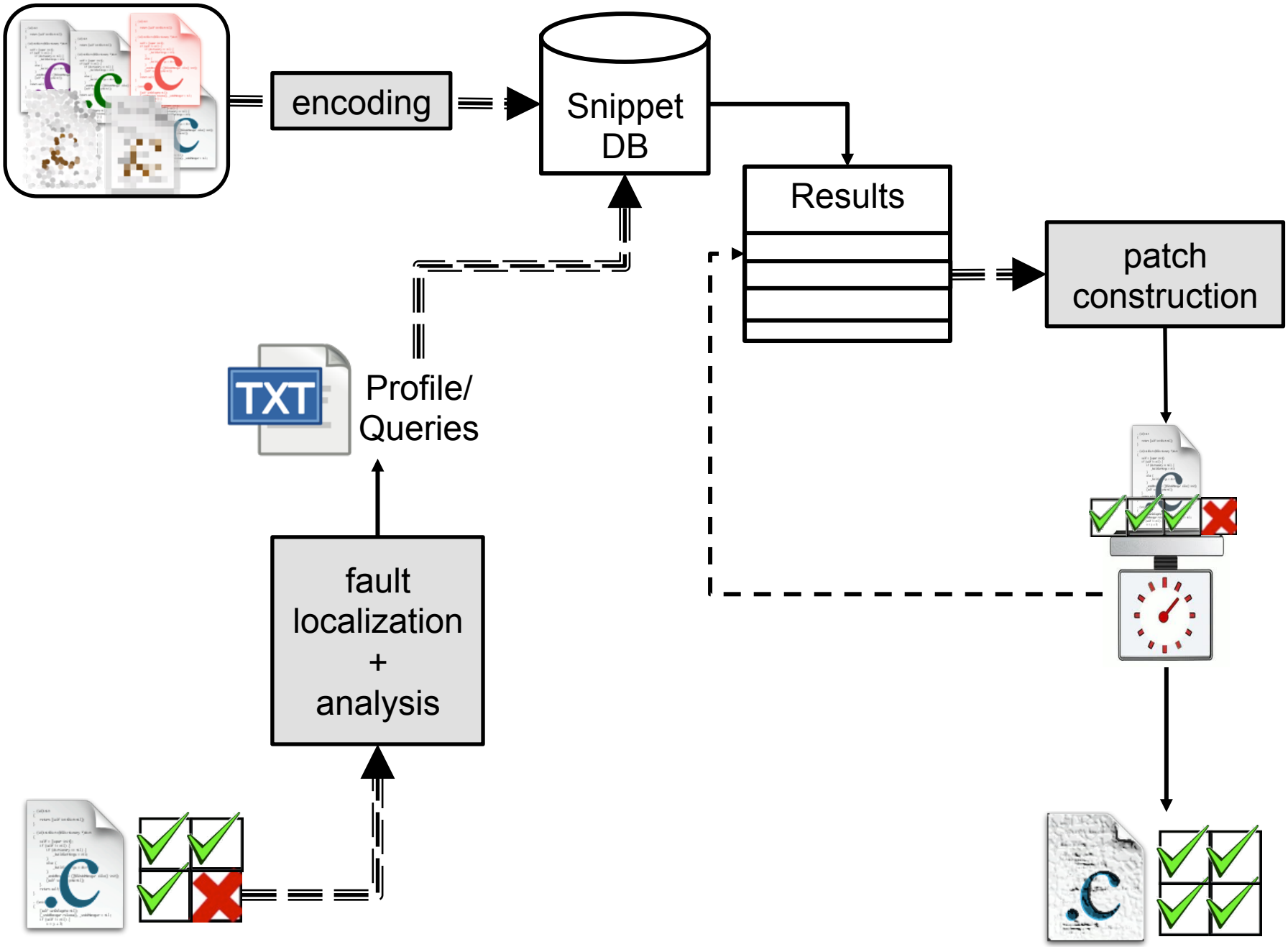
```
int med_broken(int a, int b, int c) {
  int result;
  if((a<=b && a>=c)||
     (a>=b && a<=c))
     result = a;
  else if((b<=a && b>=c)||
          (b>=a && b<=c))
     result = b;
  else
     result = c;

 return result;
}
```

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

 return result;
}
```

| Input | Expected | Pass? |
|---|---|---|
| 6,2,8 | 6 | ✓ |
| 6,8,2 | 6 | ✓ |
| 8,2,6 | 6 | ✓ |
| 8,6,2 | 6 | ✓ |


encoding
Snippet DB
Results
patch construction
TXT
Profile/ Queries
fault localization + analysis

RECALL GOAL: FIXING BUGS THIS WAY RESULTS IN HIGHER-QUALITY PATCHES.

# EVALUATION

# INTROCLASS

**Dataset: benchmark of student-written C programs**

**Key: *two independent test suites.* Use one for repair, one for validation of quality claims!**

- Code DB constructed of *other students'* answers.

| Program | Versions | Description |
|---|---|---|
| checksum | 29 | check sum of a string |
| digits | 91 | digits of a number |
| grade | 226 | grade from score |
| median | 168 | median of three numbers |
| smallest | 155 | smallest of four numbers |
| syllables | 109 | count vowels in string |
| **Total** | **778** | |

Le Goues et al., The ManyBugs and IntroClass Benchmarks for Automated Repair of C Programs, TSE 2015

# SUCCESS CRITERIA

## METRICS

**Defects repaired.**

**Patch quality: percentage of held-out test cases that a patched program passes.**

## COMPARISON

**Previous work:**

- GenProg [1]
- AE [2]
- TrpAutoRepair/RSRepair [3, 4]

[1] Claire Le Goues, ThanhVu Nguyen, Stephanie Forrest and Westley Weimer. GenProg: A Generic Method for Automated Software Repair. TSE 2012.
[2] Westley Weimer, Zachary P. Fry, Stephanie Forrest: Leveraging Program Equivalence for Adaptive Program Repair: Models and First Results. ASE 2013.
[3] Y. Qi, X. Mao, and Y. Lei. Efficient automated program repair through fault-recorded testing prioritization. ICSM 2013.
[4] Yuhua Qi, Xiaoguang Mao, Yan Lei, Ziying Dai, and Chengsong Wang. The strength of random search on automated program repair. ICSE 2014.

| program | SearchRepair | AE | GenProg | TrpAuto/ RSRepair | Total |
|---|---|---|---|---|---|
| checksum | 0 | 0 | 8 | 0 | 29 |
| digits | 0 | 17 | 30 | 19 | 91 |
| grade | 5 | 2 | 2 | 2 | 227 |
| median | 68 | 58 | 108 | 93 | 168 |
| smallest | 73 | 71 | 120 | 119 | 155 |
| syllables | 4 | 11 | 19 | 14 | 109 |
| **total** | **150** | **159** | **287** | **247** | **778** |

# CURRENT LIMITATIONS

**Snippet encoding: need support for more datatypes, library calls, console output, etc.**

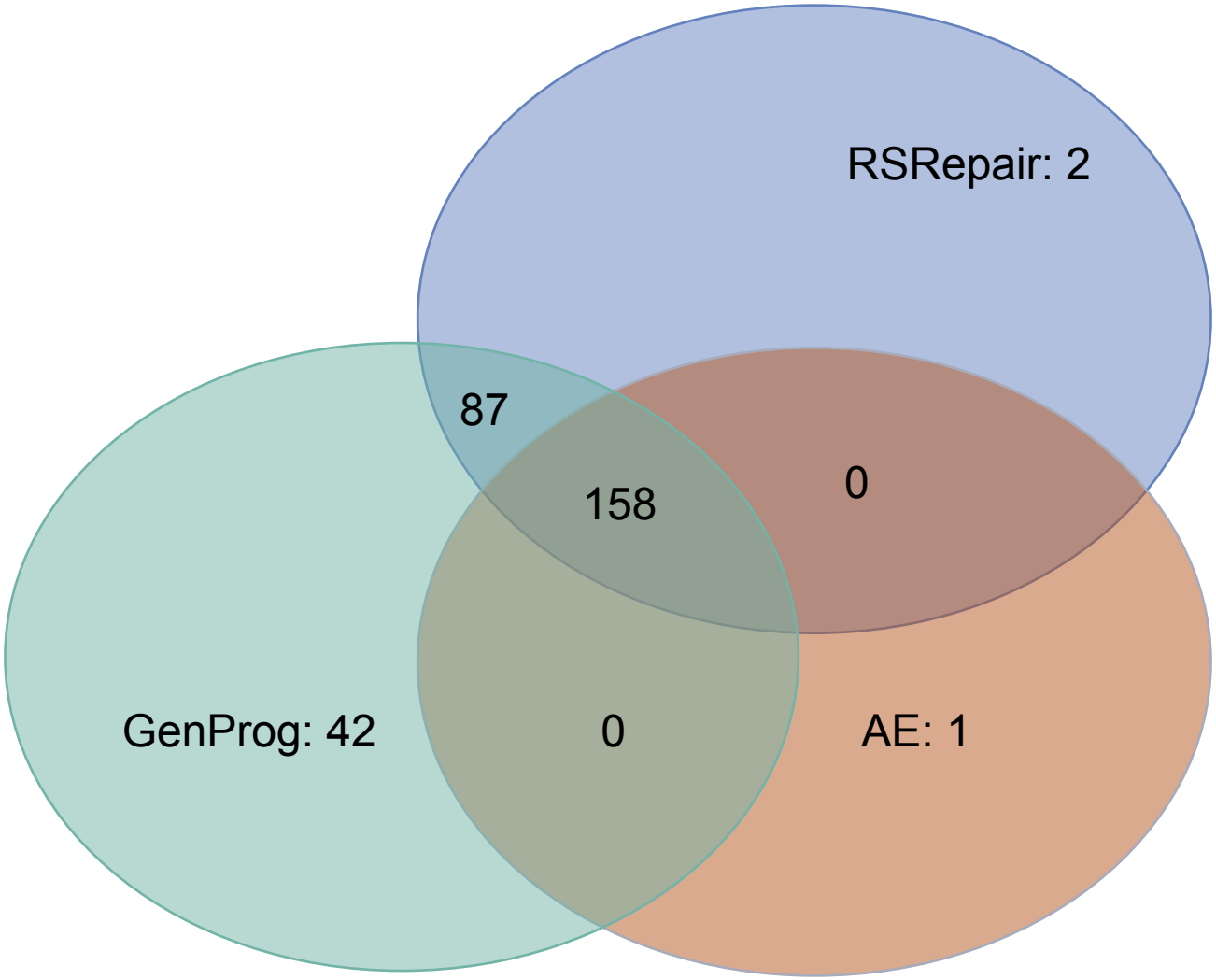
- Hand-rolled symbolic execution.

**Match queries: various inefficiencies, especially in mapping variables to context.**

| program | SearchRepair | AE | GenProg | TrpAuto/ RSRepair | Total |
|---|---|---|---|---|---|
| checksum | 0 | 0 | 8 | 0 | 29 |
| digits | 0 | 17 | 30 | 19 | 91 |
| grade | 5 | 2 | 2 | 2 | 227 |
| median | 68 | 58 | 108 | 93 | 168 |
| smallest | 73 | 71 | 120 | 119 | 155 |
| syllables | 4 | 11 | 19 | 14 | 109 |
| **total** | **150** | **159** | **287** | **247** | **778** |

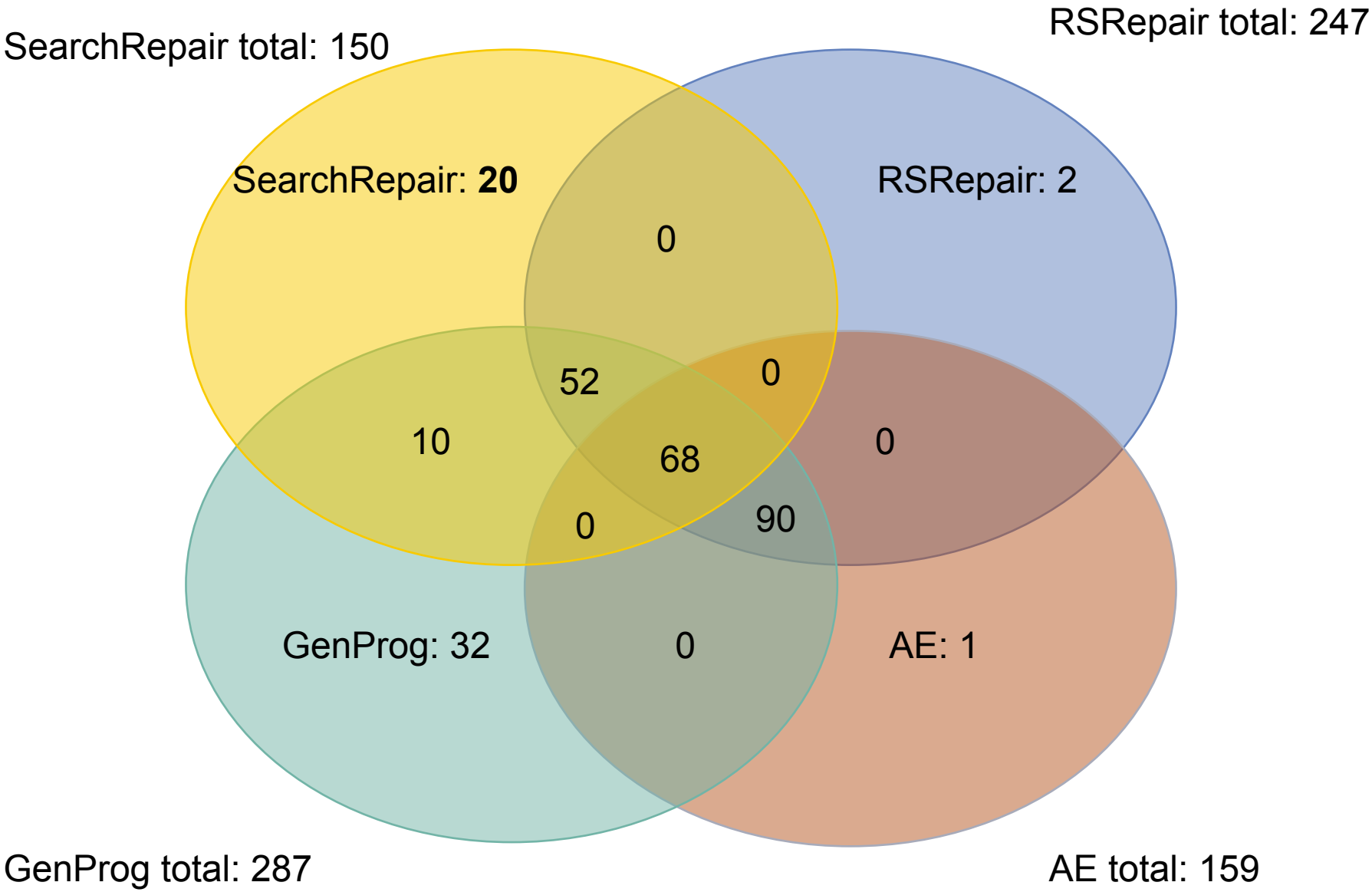
310 unique program/bugs repaired total

RSRepair total: 247




GenProg total: 287

AE total: 159


SearchRepair total: 150
RSRepair total: 247
SearchRepair: **20**
RSRepair: 2
0
52
0
10
68
0
0
90
GenProg: 32
0
AE: 1
GenProg total: 287
AE total: 159

# QUALITY

**Use the second test suite (from KLEE) to assess degree to which the patches generalize beyond the tests used to create them.**

- Recall: Patched programs pass all tests used to create them by definition.

| SearchRepair | GenProg | RSRepair/ TRPAutoRepair | AE |
|---|---|---|---|
| **97.2%** | 68.7% | 72.1% | 64.2% |

# TAKEAWAY

**SearchRepair uses semantic search to fix bugs by looking for code that *does* the right thing.**

**Compared to previous work, SearchRepair:**

- Repairs *different* faults
- Produces patches of *measurably* higher quality.

**Code at:** https://github.com/ProgramRepair/SearchRepair