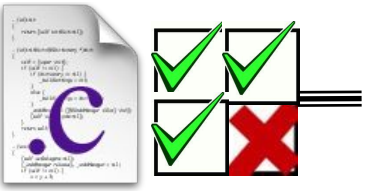
Repairing Programs with Semantic Code Search

Dr. Kathryn Stolee

North Carolina State University

Input: buggy program, tests



Repair Magic

Potential patches

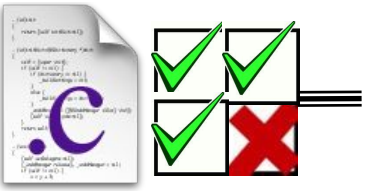
Output: fixed program

Challenge #1: How to find/create a patch?

Challenge #2: How to select a patch?

Challenge #3: How to determine the patch is correct/good?

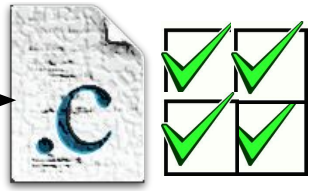
Input: buggy program, tests



SearchRepair

Potential patches

Output: fixed program



Our repair magic - uses semantic code search

Compute the median of three numbers

```
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;
}
```

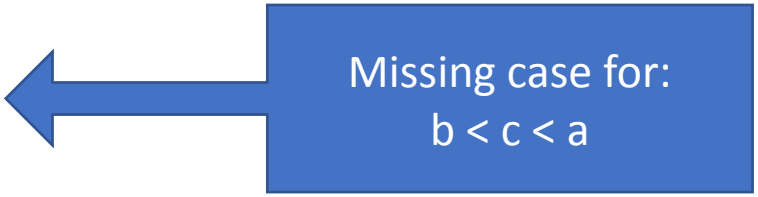
```
int median(int a, int b, int c) {
  int result = 0;
  if ((b<=a && a<=c) ||
      (c<=a && a<=b)
    result = a;
  if ((a<b && b <= c) ||
      (c<=b && b<a))
    result = b;
  if ((a<c && c<b) ||
      (b<c && c<a))
    result = c;
  return result;
}
```

```
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 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;
  else 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 < 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	✓

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.

Challenge #1: Finding Patches

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.

Semantic code search

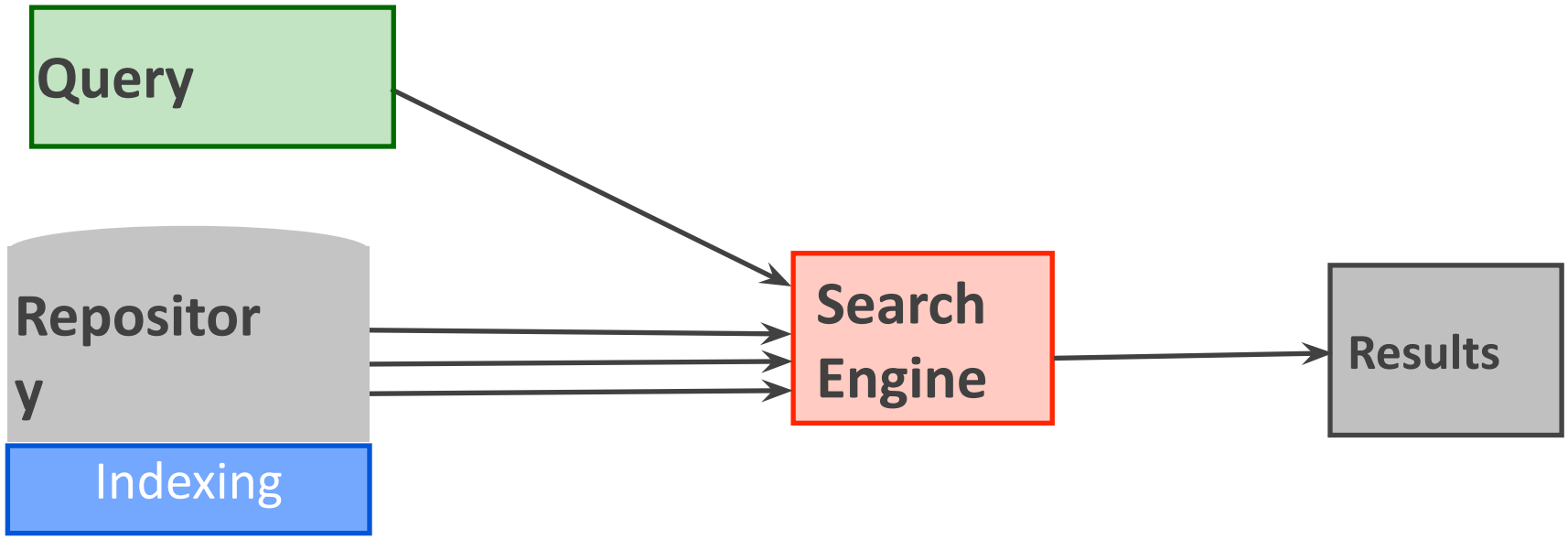
- Keyword: “C median three numbers”

- Semantic:

Input	Expected Output
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.

How Does Search Work?



About Search

Google

Semantic Search

Querying

Indexing

Matching

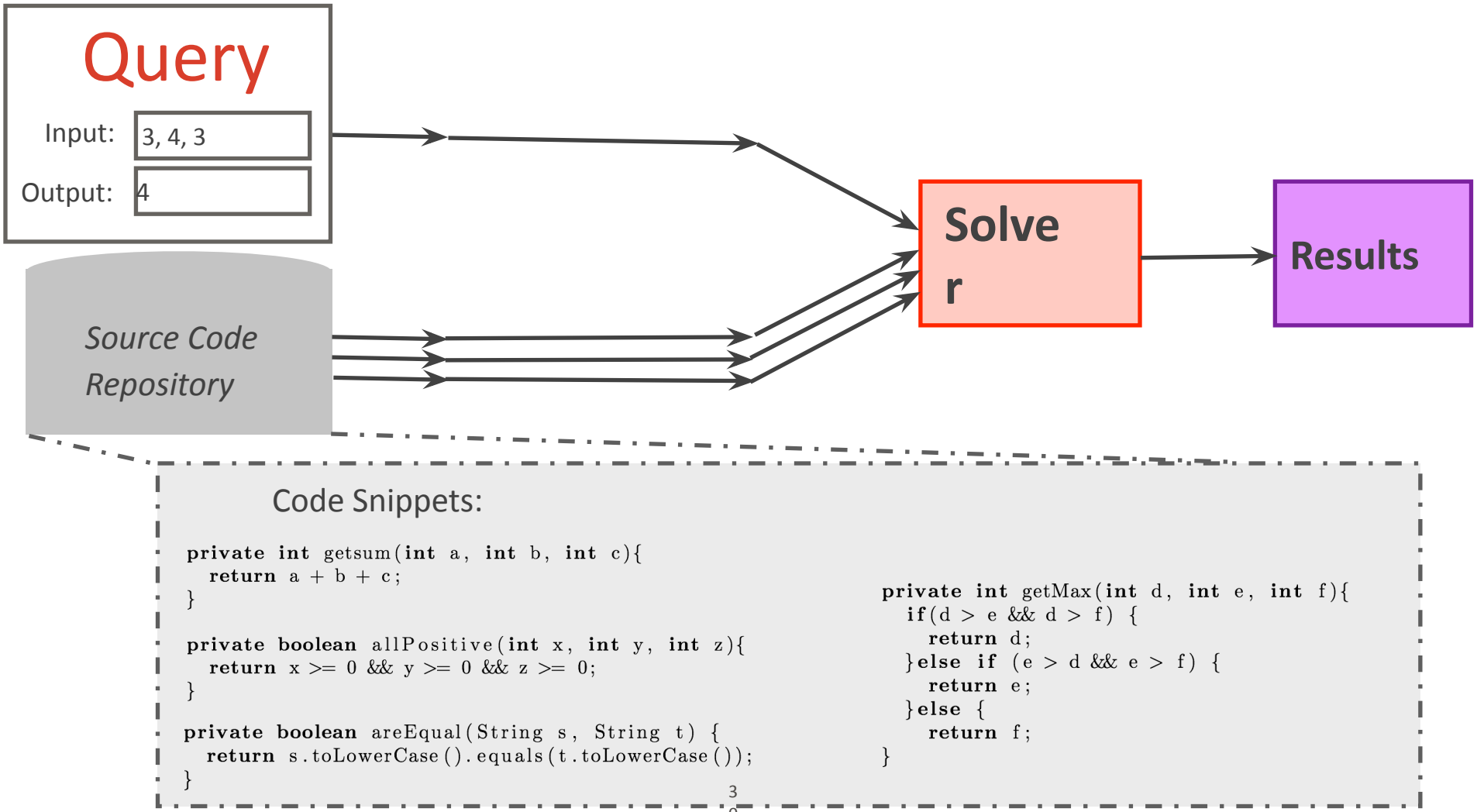
About Search

	Google	Semantic Search
Querying	keywords	input/output example
Indexing		
Matching		

About Search

	Google	Semantic Search
Querying	keywords	input/output example
Indexing		
Matching	text matching	Solver

Behind the Scenes



SMT Solvers

Satisfiability **M**odulo **T**heory solvers determine if a logical formula is satisfiable

Facts	Assertions
`a >= 0`	`(assert (>= a 0))`
`b = 2`	`(assert (= b 2))`
`c = 2`	`(assert (= c 2))`
`c = a * b`	`(assert (= ( * a b) c))`

Result: **sat** $a \mapsto 1$

SMT Solvers

Satisfiability **M**odulo **T**heory solvers determine if a logical formula is satisfiable

Facts	Assertions
`a >= 0`	`(assert (>= a 0))`
`b = ?`	**`(assert (= b ?))`**
`c = 2`	`(assert (= c 2))`
`c = a * b`	`(assert (= ( * a b) c))`

Result: **sat** $a \mapsto 2 \wedge b \mapsto 2$

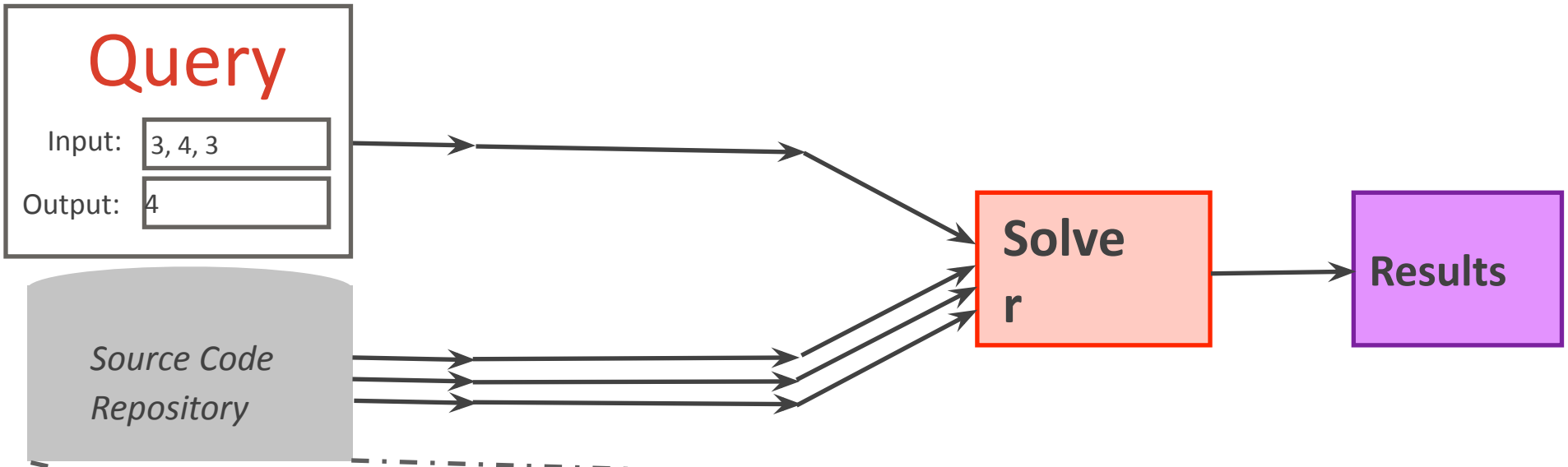
SMT Solvers

Satisfiability **M**odulo **T**heory solvers determine if a logical formula is satisfiable

Facts	Assertions
a = 0	**(assert (= a 0))**
b = ?	(assert (= b ?))
c = 2	(assert (= c 2))
c = a * b	(assert (= (* a b) c))

Result: **unsat**

Behind the Scenes



Code Snippets:

```
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;
}
```

About Search

	Google	Semantic Search
Querying	keywords	input/output example
Indexing		
Matching	text matching	Solver

About Search

	Google	Semantic Search
Querying	keywords	input/output example
Indexing	words and their locations in web pages	code behavior
Matching	text matching	Solver

Indexing

Code Snippet:

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

```
(declare-fun a () Int)
(declare-fun b () Int)
(declare-fun c () Int)
(declare-fun return () Int)
(assert (= return (+ (+ a b ) c)))
```

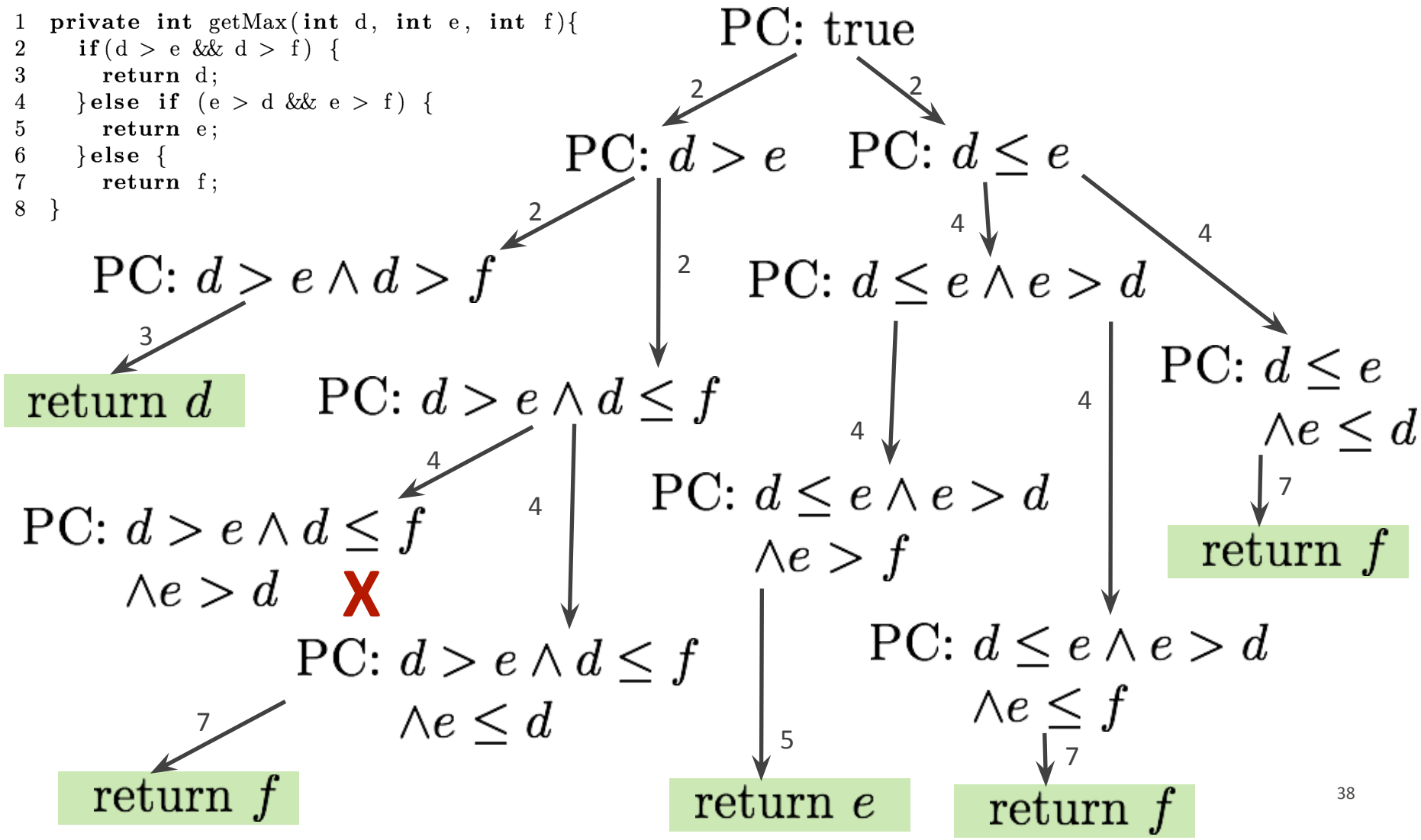
Code Snippet:

```
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;
}
```

```
(declare-fun d () Int)
(declare-fun e () Int)
(declare-fun f () Int)
(declare-fun return () Int)
(assert (or
  (and (> d e) (> d f) (= return d))
  (and (> d e) (≤ d f) (≤ e d) (= return f))
  (and (≤ d e) (> e d) (> e f) (= return e))
  (and (≤ d e) (> e d) (≤ e f) (= return f))
  (and (≤ d e) (≤ e d) (= return f))))
```

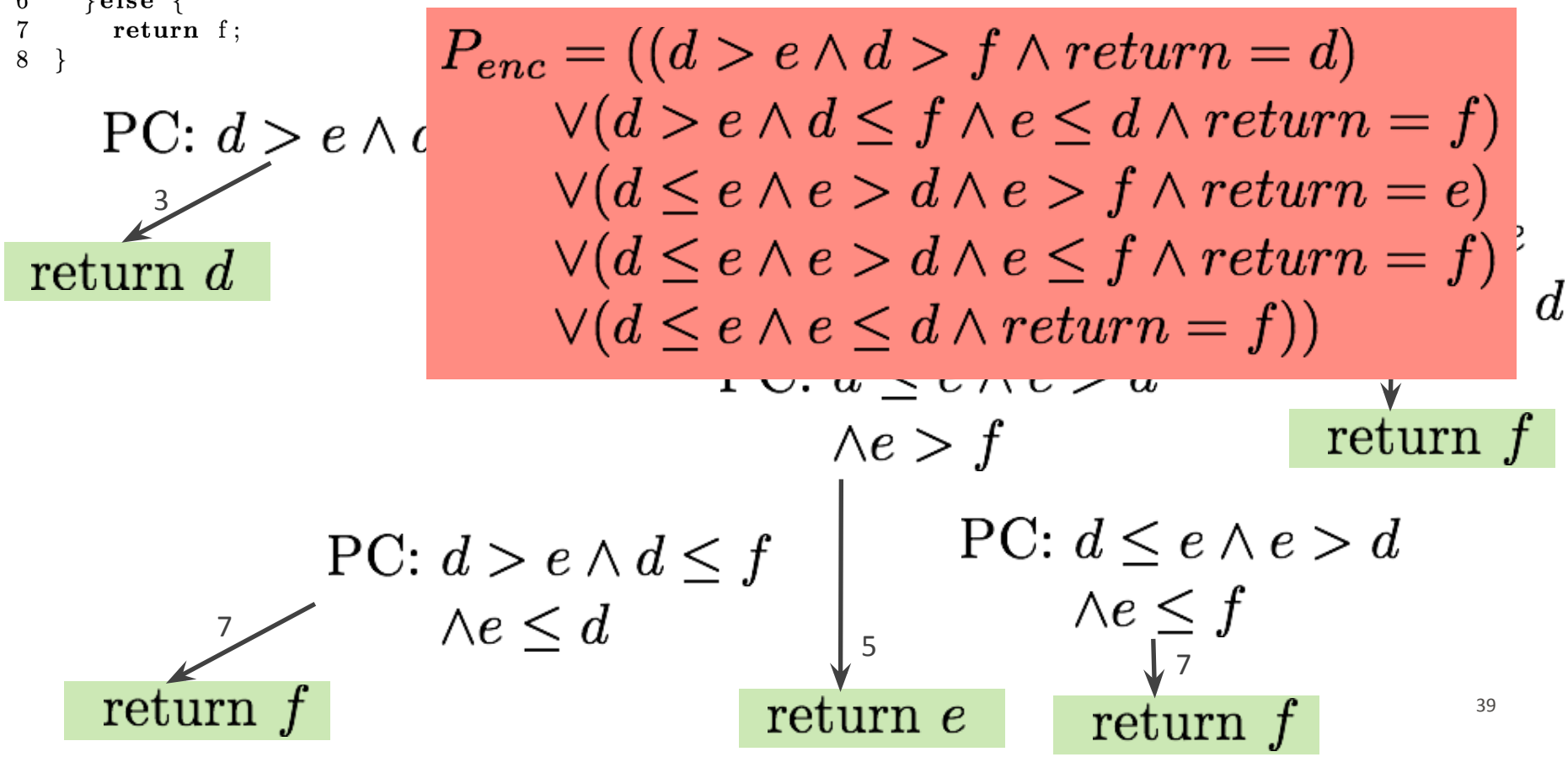
Symbolic Execution

```
1 private int getMax(int d, int e, int f){
2   if(d > e && d > f) {
3     return d;
4   }else if (e > d && e > f) {
5     return e;
6   }else {
7     return f;
8 }
```



Symbolic Execution

```
1 private int getMax(int d, int e, int f){
2   if(d > e && d > f) {
3     return d;
4   }else if (e > d && e > f) {
5     return e;
6   }else {
7     return f;
8 }
```




```
(declare-fun a () Int)
(declare-fun b () Int)
(declare-fun c () Int)
(declare-fun return () Int)
(assert (= return (+ (+ a b ) c)))
```

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

}

About Search

	Google	Semantic Search
Querying	keywords	input/output example
Indexing	words and their locations in web pages	code behavior
Matching	text matching	Solver

Matching

Encoding

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

Potential Search Result

```
(declare-fun a () Int)
(declare-fun b () Int)
(declare-fun c () Int)
(declare-fun return () Int)
(assert (= return (+ (+ a b) c)))
(assert (and (= a 3) (= b 0)
    (= c 0)))
(assert (= return 3))
```

Query

Input	Output	Result
3, 4, 3	4	**unsat**
3, 0, 0	3	**sat**

Not a Result!

Matching

```
1 private int getMax(int d, int e, int f){
2   if(d > e && d > f) {
3     return d;
4   }else if (e > d && e > f) {
5     return e;
6   }else {
7     return f;
8 }
```

PC1: $d > e \wedge d > f \wedge return = d$
PC2: $d > e \wedge d \leq f \wedge e \leq d \wedge return = f$
PC3: $d \leq e \wedge e > d \wedge e > f \wedge return = e$
PC4: $d \leq e \wedge e > d \wedge e \leq f \wedge return = f$
PC5: $d \leq e \wedge e \leq d \wedge return = f$

Query

Input	Output	Result
3, 4, 3	4	sat
3, 0, 0	3	sat

This is a result!

Bringing it all together: Semantic Search for Program Repair

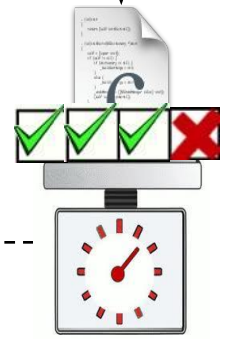
encoding

Snippet DB

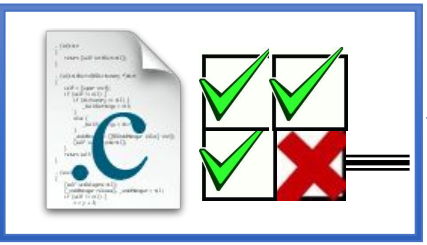
Results

patch construction

Profile/ Queries



fault localization + analysis



This is the buggy program and test cases

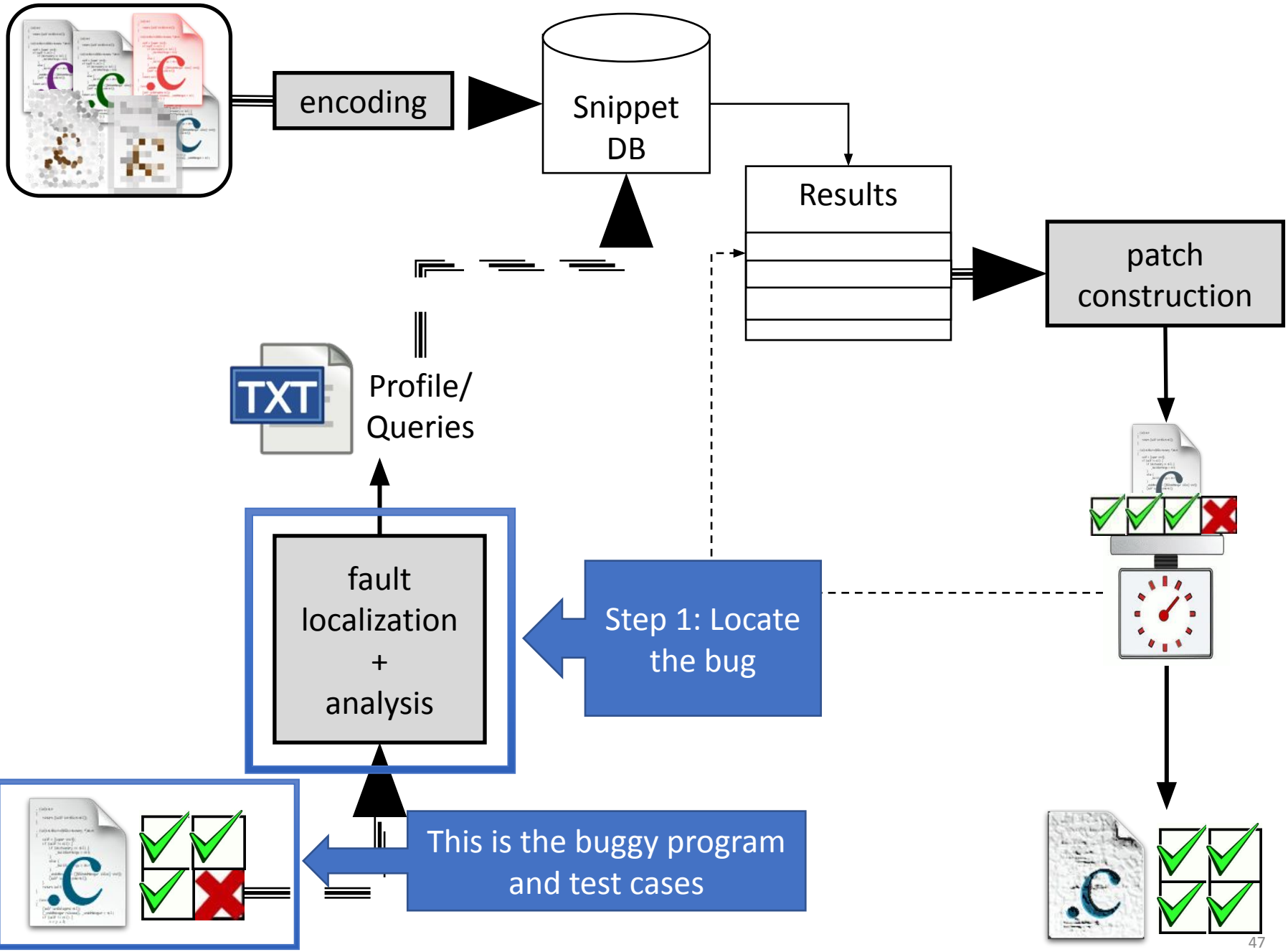
SearchRepair: The Plan

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.

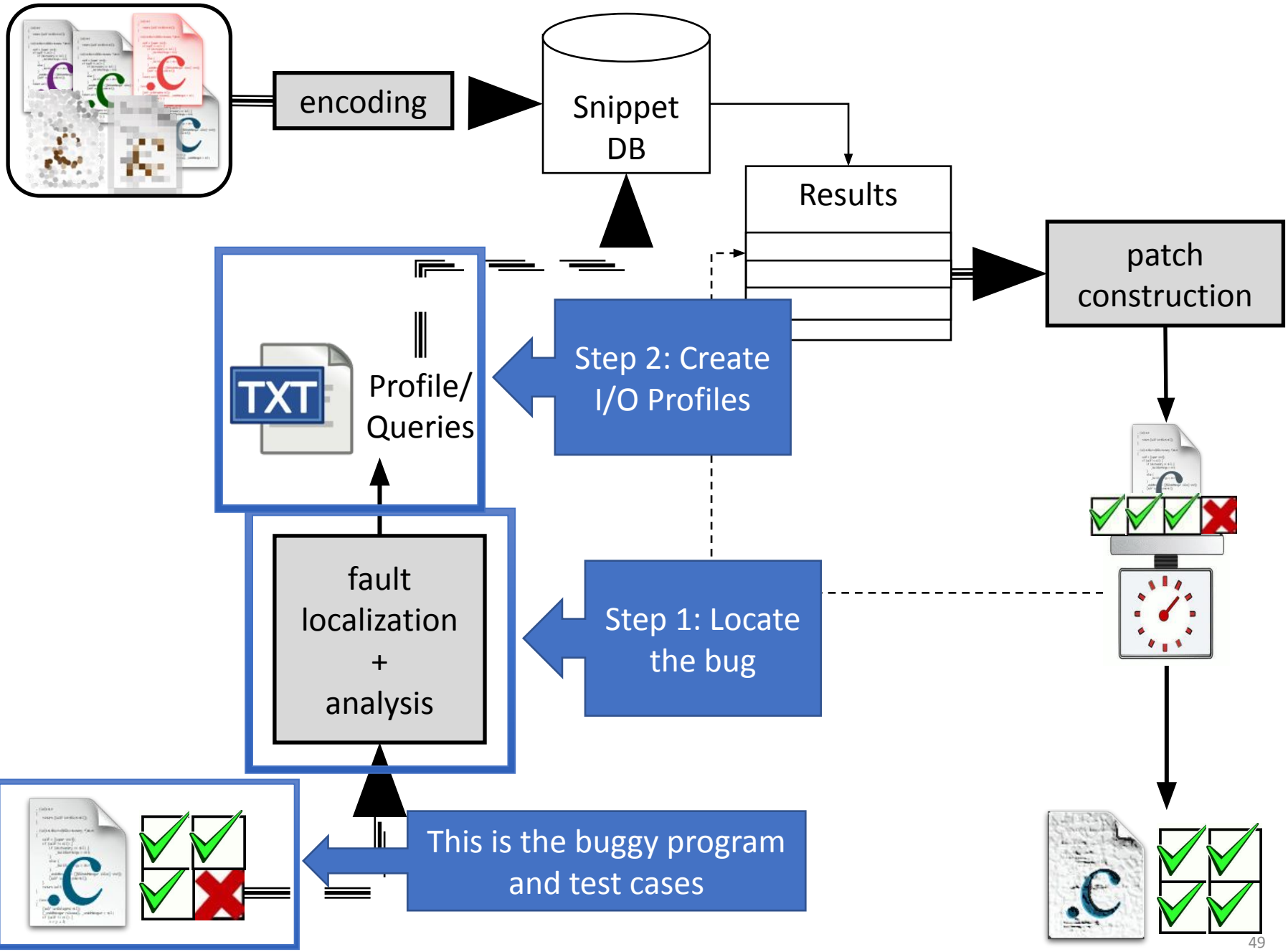

encoding
Snippet DB
Results
patch construction
TXT
Profile/ Queries
fault localization + analysis
Step 1: Locate the bug
This is the buggy program and test cases

Modified SB-Fault localization

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

```
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;
}
```

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.

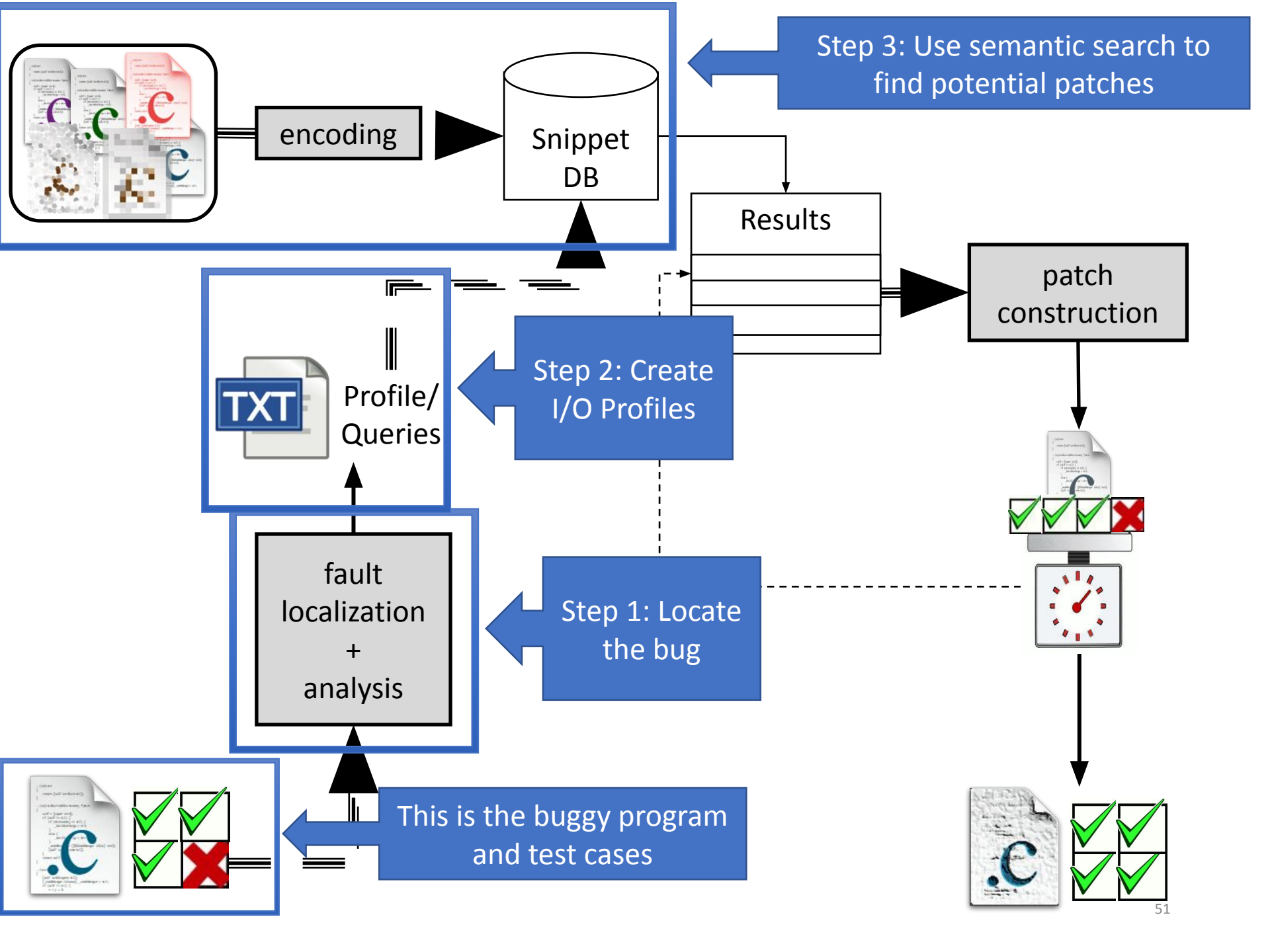

encoding
Snippet DB
Results
patch construction
TXT
Profile/ Queries
Step 2: Create I/O Profiles
fault localization + analysis
Step 1: Locate the bug
This is the buggy program and test cases

```
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;
}
```

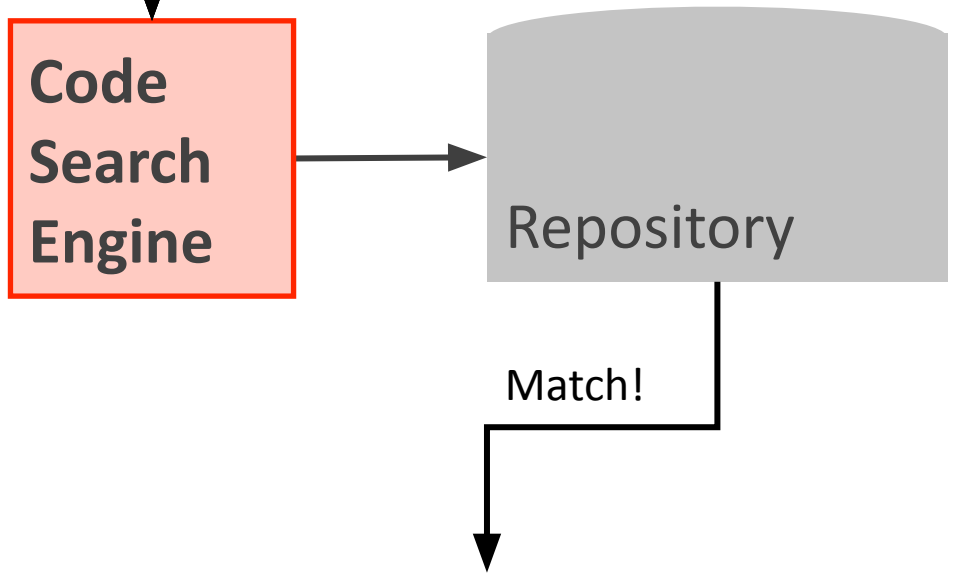
Input:
a=6, b=2, c=8,
result=*

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

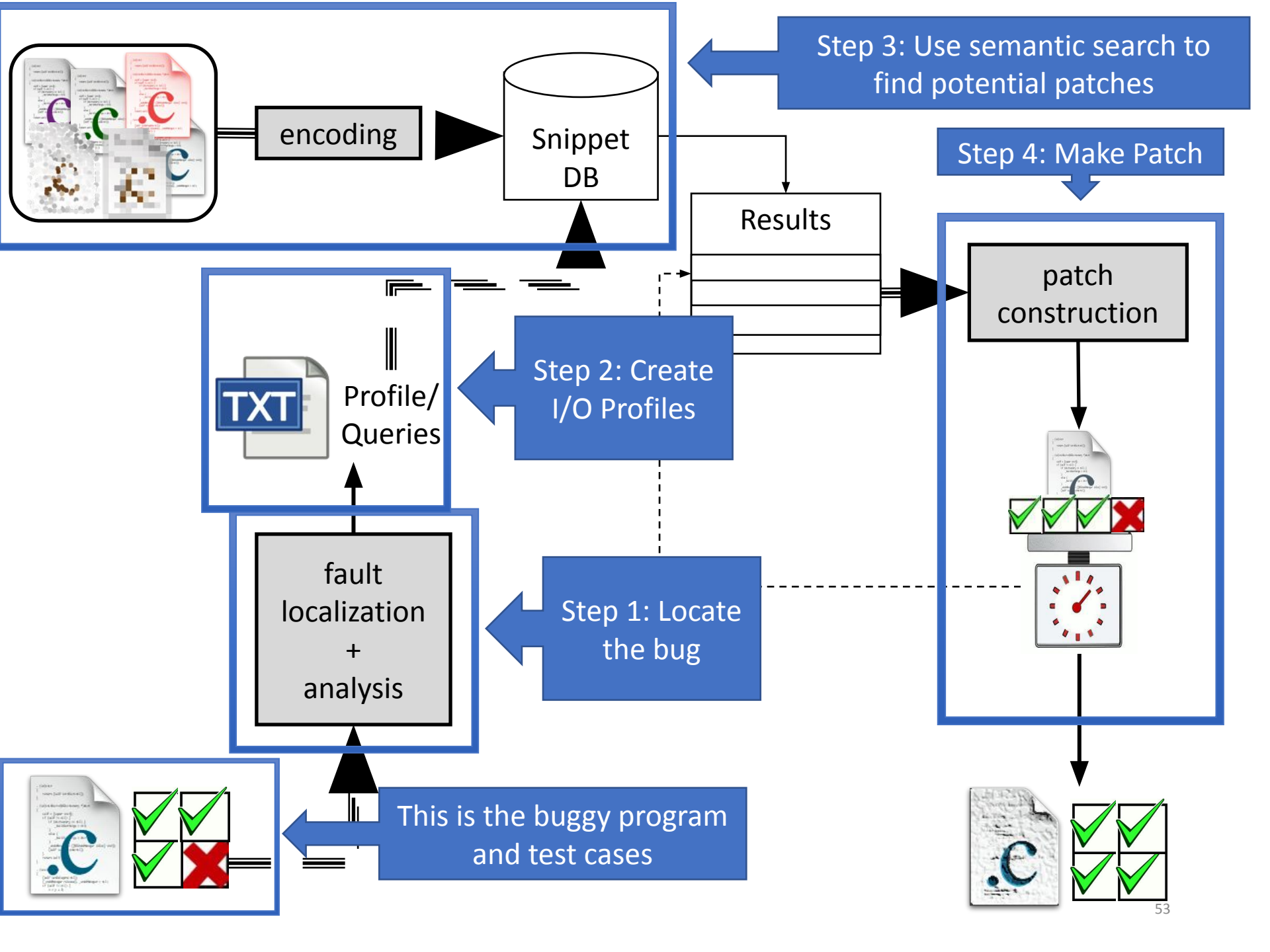
Input	Expected	Pass?
6,2,8	6	✓
6,8,2	6	✓
8,2,6	6	X
8,6,2	6	✓


Step 3: Use semantic search to find potential patches
encoding
Snippet DB
Results
patch construction
TXT
Profile/ Queries
Step 2: Create I/O Profiles
fault localization + analysis
Step 1: Locate the bug
This is the buggy program and test cases

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




```
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;
```


Step 3: Use semantic search to find potential patches
encoding
Snippet DB
Step 4: Make Patch
Results
patch construction
TXT
Profile/ Queries
Step 2: Create I/O Profiles
fault localization + analysis
Step 1: Locate the bug
This is the buggy program and test cases

```
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;
```

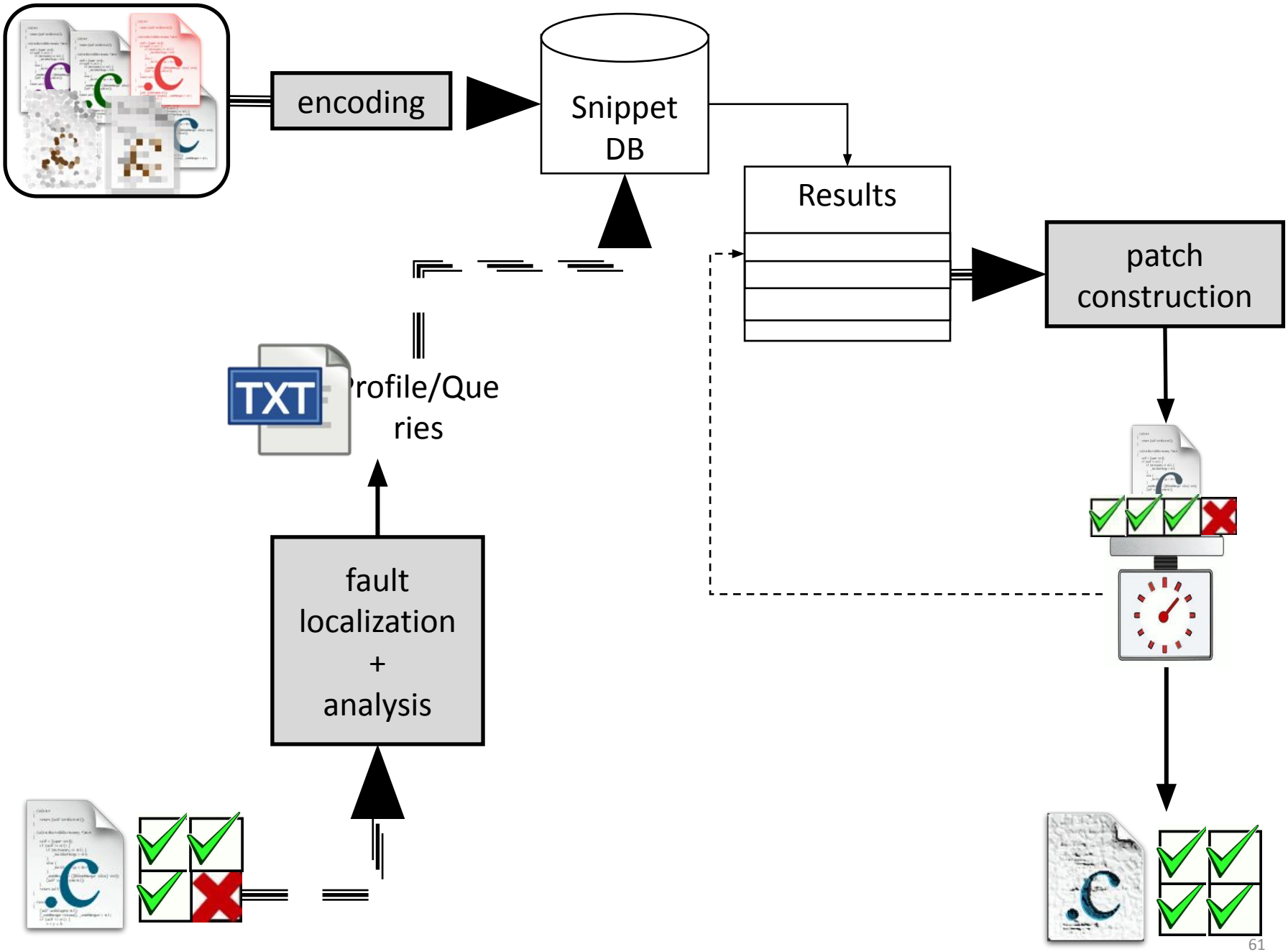
```
int med_broken(int a, int b, int c) {
  int result;
  if((a<=b && a>=c)||
     (a>=b && a<=c))
     result = a;
  else if((b<=a && b>=c)||
          (b>=a && b<=c))
     result = b;
  else
     result = c;

 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	X
8,6,2	6	✓

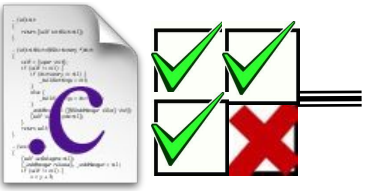
```
int med_broken(int a, int b, int c) {
  int result;
  if((a<=b && a>=c)||
     (a>=b && a<=c))
     result = a;
  else if((b<=a && b>=c)||
          (b>=a && b<=c))
     result = b;
  else
     result = c;


 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
rofile/Que ries
fault localization + analysis

Challenge #3: Patch quality

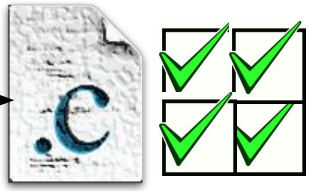
Input: buggy program, tests



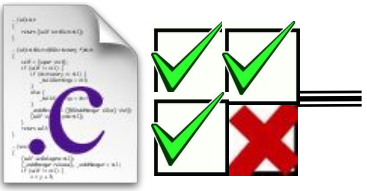
Automatic repair magic

Potential patches

Output: fixed program



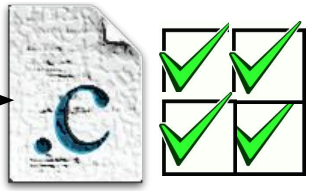
Input: buggy program, tests



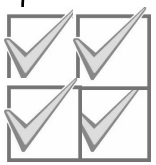
Automatic repair magic

Potential patches

Output: fixed program



Performance on withheld tests?

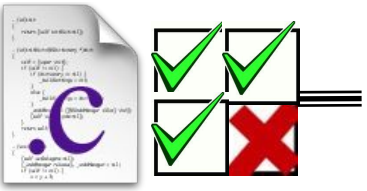


Overfitting

- 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!

Recall Goal: fixing bugs in a way that results in higher-quality patches.

Evaluation

Introclass

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**	

- 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.

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]

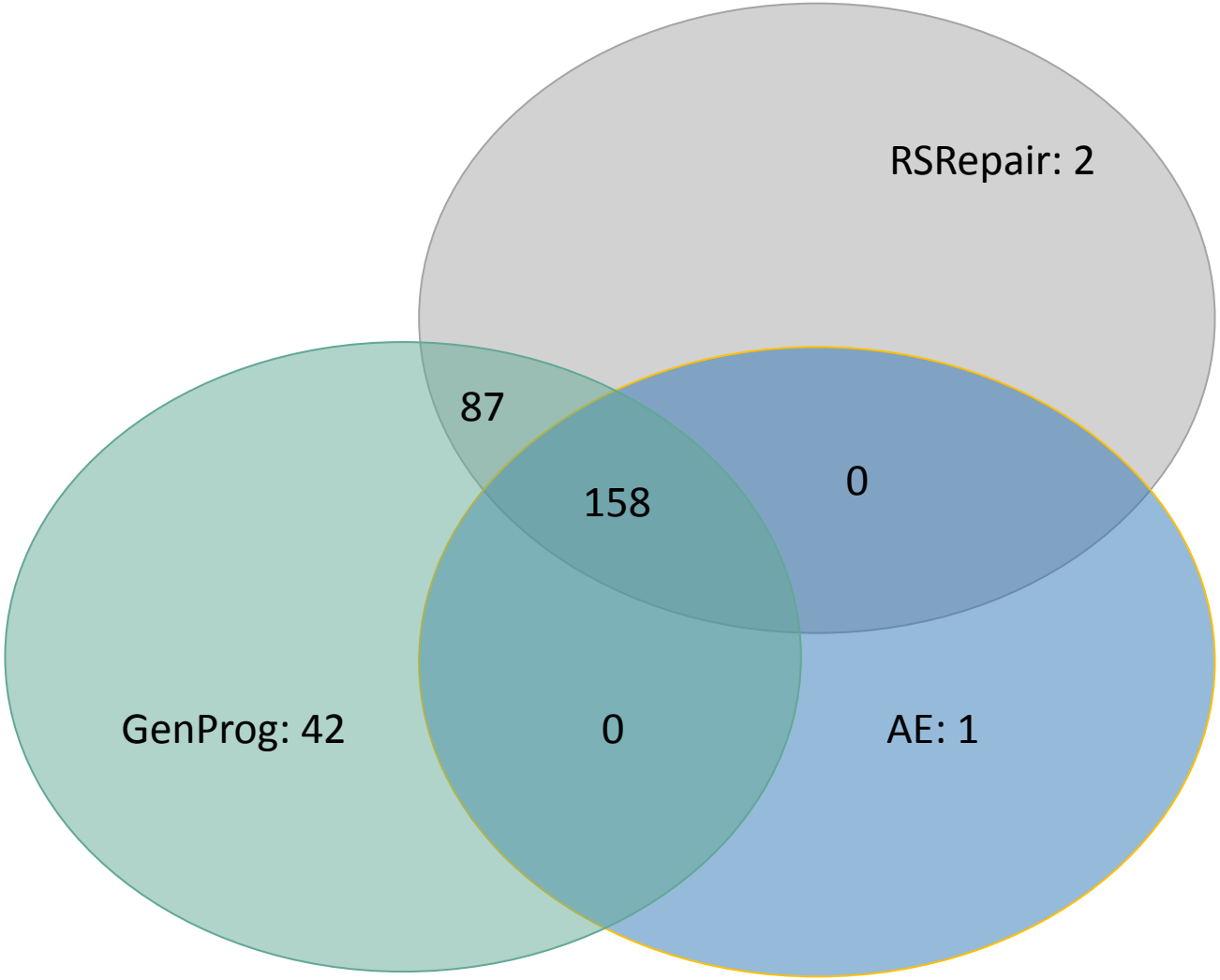
[1] Claire Le Goues, ThanhVu Nguyen, Stephanie Forrest and Westley Weimer. GenProg: A Generic Method for Automated Software Repair. TSE 2012.
[2] Westley Weimer, Zachary P. Fry, Stephanie Forrest: Leveraging Program Equivalence for Adaptive Program Repair: Models and First Results. ASE 2013.
[3] Y. Qi, 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					29
digits					91
grade					227
median					168
smallest					155
syllables					109
total					**778**

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

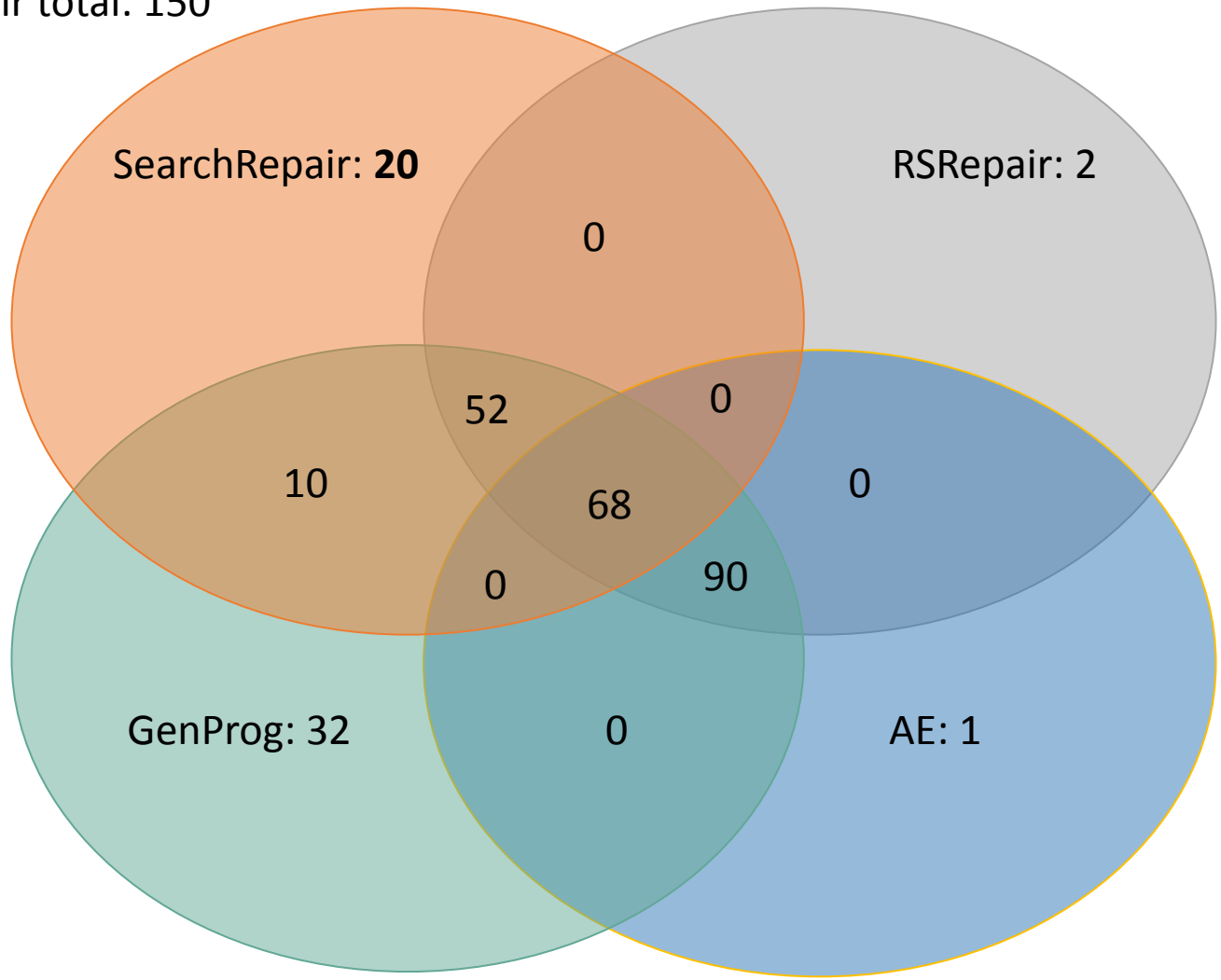


GenProg total: 287

AE total: 159

SearchRepair total: 150

RSRepair total: 247



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

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
	68.7%		

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
	68.7%	72.1%	

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
	68.7%	72.1%	64.2%

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%

Cool, but does it scale?

ManyBugs

Program	kLOC	Tests	Versions	Patched
gzip	8.4	12	4	**0**
libtiff	77	78	9	**9**
php	1099	8471	39	**8**
python	407	355	4	**2**
wireshark	2814	63	2	**2**
gmp	145	146	2	**1**
lighttpd	62	295	5	**1**
				23

ManyBugs

Program	kLOC	Tests	Versions	Patched	Held-out Tests?
gzip	8.4	12	4	**0**	0
libtiff	77	78	9	**9**	2
php	1099	8471	39	**8**	3
python	407	355	4	**2**	0
wireshark	2814	63	2	**2**	2
gmp	145	146	2	**1**	0
lighttpd	62	295	5	**1**	1
				23	8

Python bug #69223

Developer Patch

```
   }
+  if (timeout < 0) {
+    PyErr_SetString(PyExc_ValueError ,
+        "timeout must be non-negative");
+    return NULL;
+  }
   seconds = (long)timeout;
```

SearchRepair Patch

```
if (n < 0) {
    PyErr_SetString(PyExc_ValueError ,
       "read length must be positive");
    return NULL;
}
```

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.
  - Can patch real bugs

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

Challenge #2: Selecting a Patch

Possible solutions can be numerous!

This brings us to semi-automated repair – a human in the loop *at some point during the repair process*.

	A	B
1	**Input**	**Output**
2	3	21
3	5	
4	6	
5	7	

=sum(A2:A5)

=sum(A2:A12)

=A2*A5

=3*A3+A4

=2*(A4+A5)-A3

=3*A4+2*A3-A5

=(A2*A3*A4)-(A3*13)-4

=if(A2>2,sum(A2:A5),0)

=A4*A5/2

=(A3*A4)-(A2+A4)

=(A2*A2*A2)-A4

This is the *solution space*

How do we choose?

	A	B
1	**Input**	**Output**
2	3	21
3	5	
4	6	
5	7	

Example 1 : 11 programs

```
=sum(A2:A5)
=sum(A2:A12)
=A2*A5
=3*A3+A4
=2*(A4+A5)-A3
=3*A4+2*A3-A5
=(A2*A3*A4)-(A3*13)-4
=if(A2>2,sum(A2:A5),0)
=A4*A5/2
=(A3*A4)-(A2+A4)
=(A2*A2*A2)-A4
```

	A	B
1	**Input**	**Output**
2	3	24
3	6	
4	7	
5	8	

Example 1 : 11 programs

Example 2

=sum(A2:A5)

=sum(A2:A12)

=A2*A5

=3*A3+A4

=2*(A4+A5)-A3

=3*A4+2*A3-A5

=(A2*A3*A4)-(A3*13)-4

=if(A2>2,sum(A2:A5),0)

=A4*A5/2

=(A3*A4)-(A2+A4)

=(A2*A2*A2)-A4

	A	B
1	**Input**	**Output**
2	3	24
3	6	
4	7	
5	8	

Example 1 : 11 programs

Example 2 : 5 programs

=sum(A2:A5)

=sum(A2:A12)

=A2*A5

=3*A3+A4

=2*(A4+A5)-A3

=3*A4+2*A3-A5

=(A2*A3*A4)-(A3*13)-4

=if(A2>2,sum(A2:A5),0)

=A4*A5/2

=(A3*A4)-(A2+A4)

=(A2*A2*A2)-A4

	A	B
1	**Input**	**Output**
2	3	27
3	7	
4	8	
5	9	

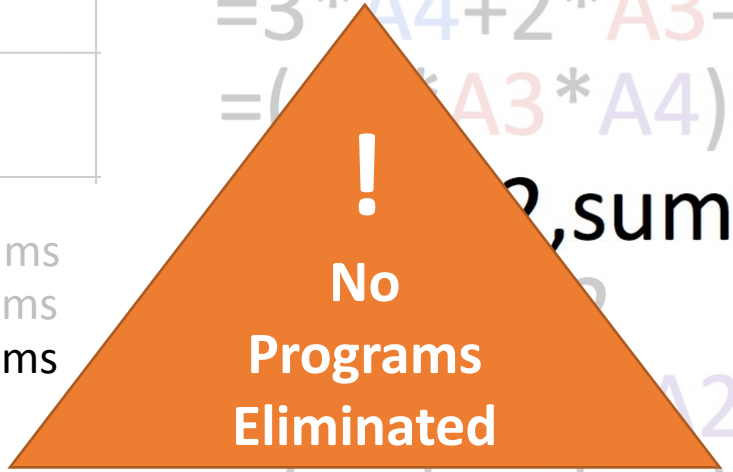
Example 1 : 11 programs
Example 2 : 5 programs
Example 3

=sum(A2:A5)
=sum(A2:A12)
=A2*A5
=3*A3+A4
=2*(A4+A5)-A3
=3*A4+2*A3-A5
=(A2*A3*A4)-(A3*13)-4
=if(A2>2,sum(A2:A5),0)
=A4*A5/2
=(A3*A4)-(A2+A4)
=(A2*A2*A2)-A4

	A	B
1	**Input**	**Output**
2	3	27
3	7	
4	8	
5	9	

Example 1 : 11 programs
Example 2 : 5 programs
Example 3 : 5 programs

=sum(A2:A5)
=sum(A2:A12)
=A2*A5
=3*A3+A4
=2*(A4+A5)-A3
=3*A4+2*A3-A5
=(A3*A4)-(A3*13)-4
=2,sum(A2:A5),0)
=(A2+A4)
=(A2*A2*A2)-A4



	A	B
1	**Input**	**Output**
2	3	15
3	4	
4	3	
5	5	

Example 1 : 11 programs
Example 2 : 5 programs
Example 3 : 5 programs
Example 4

=sum(A2:A5)
=sum(A2:A12)
=A2*A5
=3*A3+A4
=2*(A4+A5)-A3
=3*A4+2*A3-A5
=(A2*A3*A4)-(A3*13)-4
=if(A2>2,sum(A2:A5),0)
=A4*A5/2
=(A3*A4)-(A2+A4)
=(A2*A2*A2)-A4

	A	B
1	**Input**	**Output**
2	3	15
3	4	
4	3	
5	5	

Example 1 : 11 programs
Example 2 : 5 programs
Example 3 : 5 programs
Example 4 : 4 programs

=sum(A2:A5)
=sum(A2:A12)
=A2*A5
=3*A3+A4
=2*(A4+A5)-A3
=3*A4+2*A3-A5
=(A2*A3*A4)-(A3*13)-4
=if(A2>2,sum(A2:A5),0)
=A4*A5/2
=(A3*A4)-(A2+A4)
=(A2*A2*A2)-A4

How can we make it easier *for the user*?

	A	B
1	Generated Input	Output
2		
3		
4		
5		

Example 1 : 11 programs

=sum(A2:A5)

=sum(A2:A12)

=A2*A5

=3*A3+A4

=2*(A4+A5)-A3

=3*A4+2*A3-A5

=(A2*A3*A4)-(A3*13)-4

=if(A2>2,sum(A2:A5),0)

=A4*A5/2

=(A3*A4)-(A2+A4)

=(A2*A2*A2)-A4

	A	B
1	Generated Input	Output
2	3	15
3	6	
4	7	
5	5	

Example 1 : 11 programs

Example 2

=sum(A2:A5)

=sum(A2:A12)

=A2*A5

=3*A3+A4

=2*(A4+A5)-A3

=3*A4+2*A3-A5

=(A2*A3*A4)-(A3*13)-4

=if(A2>2,sum(A2:A5),0)

=A4*A5/2

=(A3*A4)-(A2+A4)

=(A2*A2*A2)-A4

	A	B
1	Generated Input	Output
2	3	15
3	6	
4	7	
5	5	

Example 1 : 11 programs

Example 2 : 1 program

=sum(A2:A5)

=sum(A2:A12)

=A2*A5

=3*A3+A4

=2*(A4+A5)-A3

=3*A4+2*A3-A5

=(A2*A3*A4)-(A3*13)-4

=if(A2>2,sum(A2:A5),0)

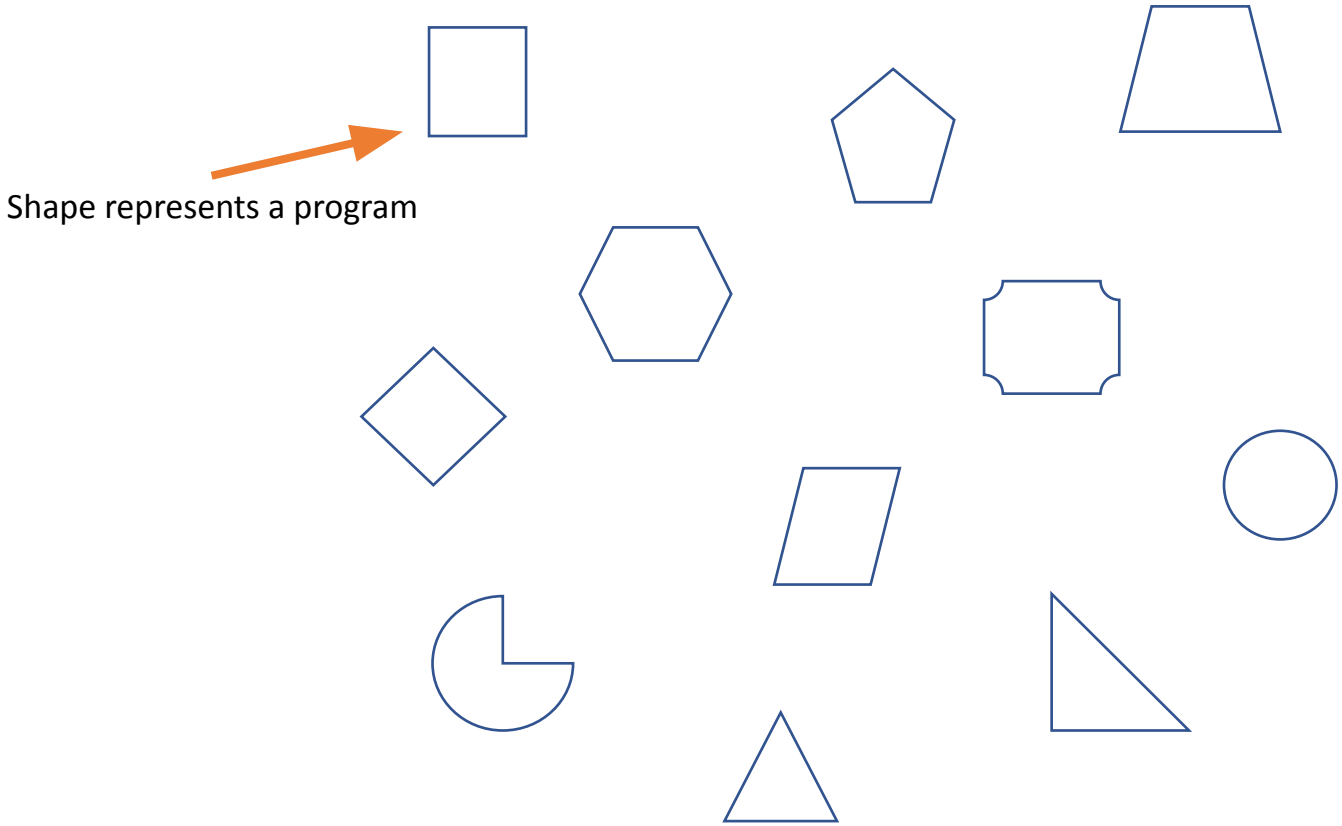
=A4*A5/2

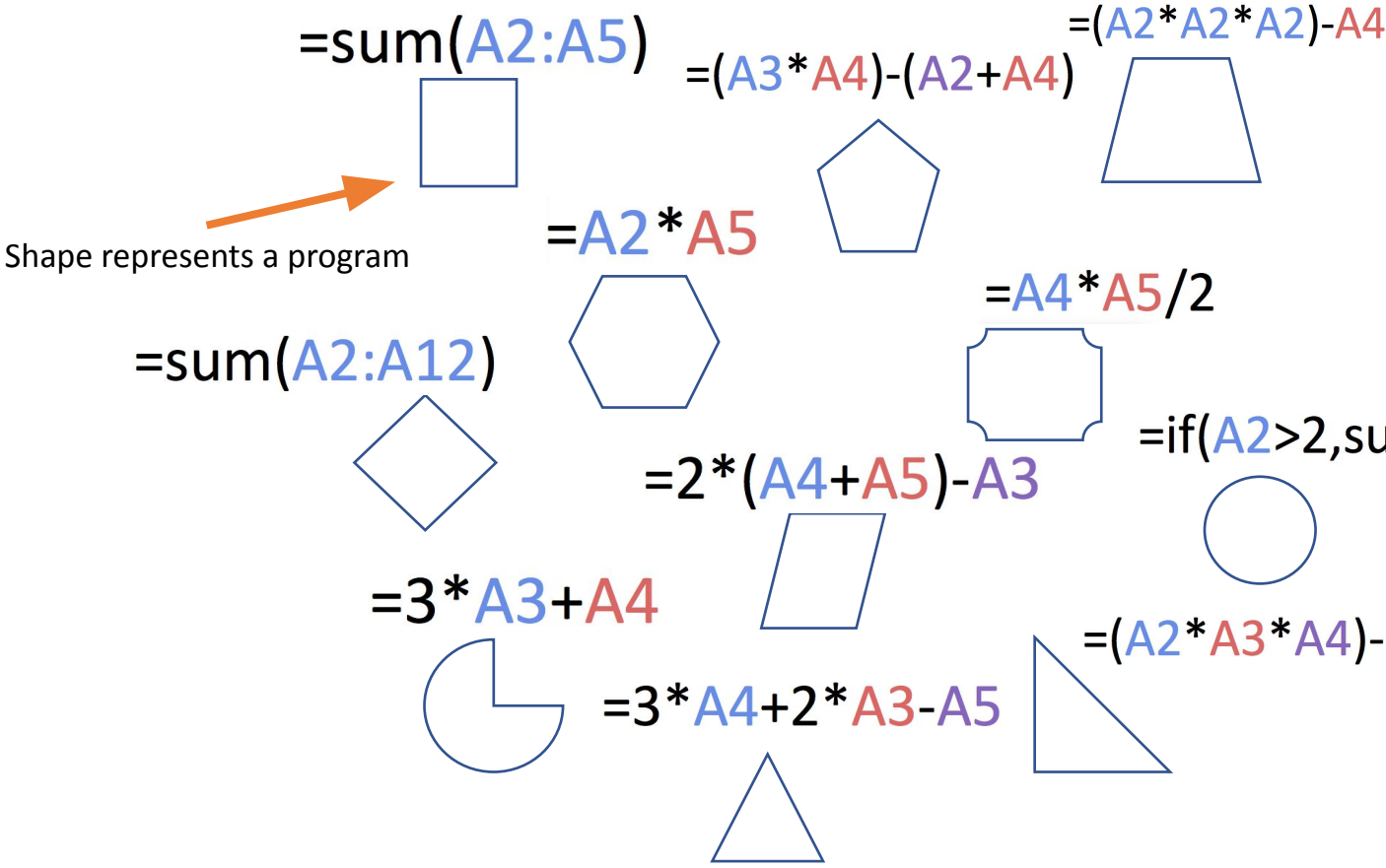
=(A3*A4)-(A2+A4)

=(A2*A2*A2)-A4

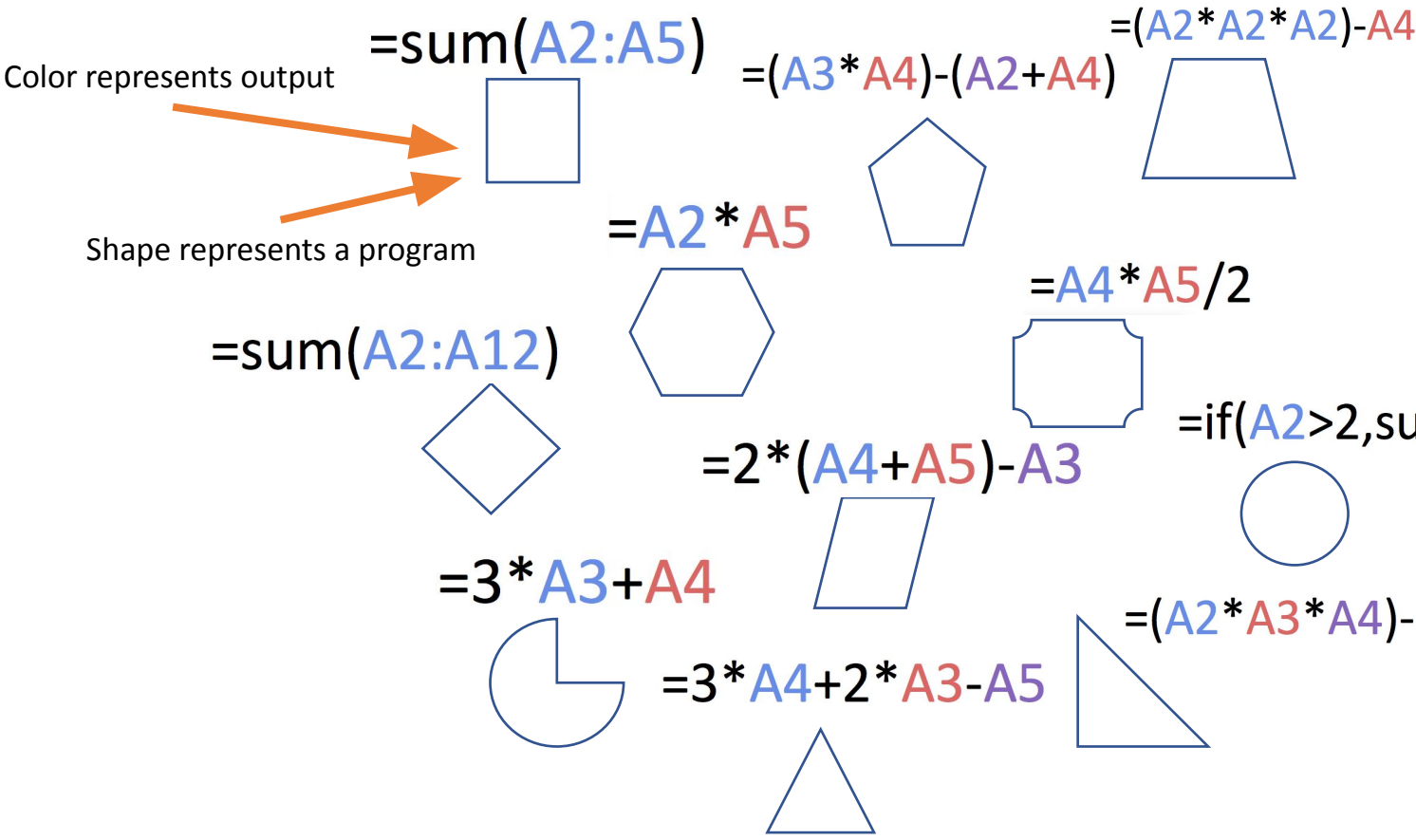
Goal: reduce user decisions in pruning the solution space

Approach: find an input that maximizes the diversity of output values


Shape represents a program

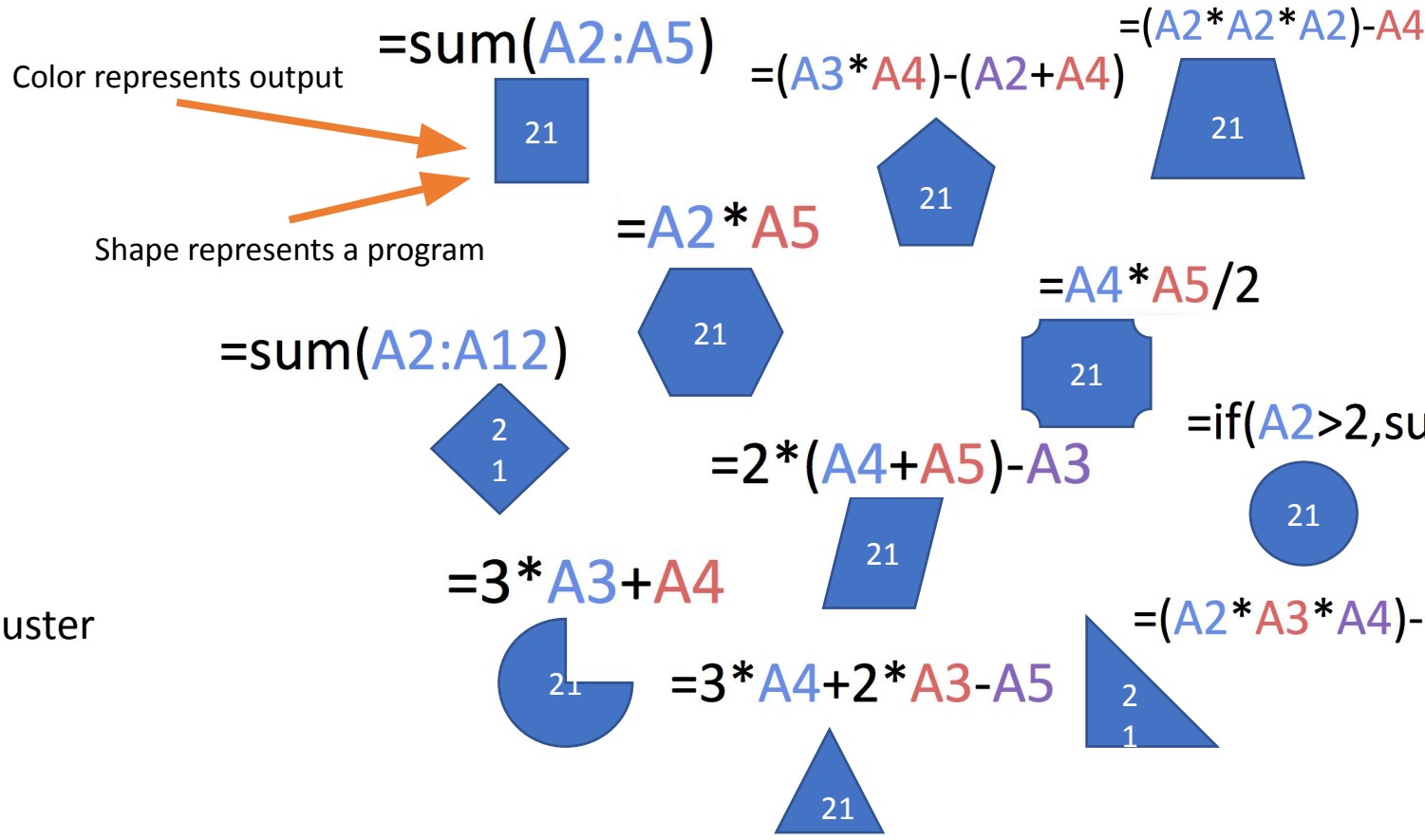

=sum(A2:A5)
=(A3*A4)-(A2+A4)
=(A2*A2*A2)-A4
Shape represents a program
=A2*A5
=A4*A5/2
=sum(A2:A12)
=if(A2>2,su
=2*(A4+A5)-A3
=3*A3+A4
=(A2*A3*A4)-
=3*A4+2*A3-A5

	A
1	Input
2	3
3	5
4	6
5	7



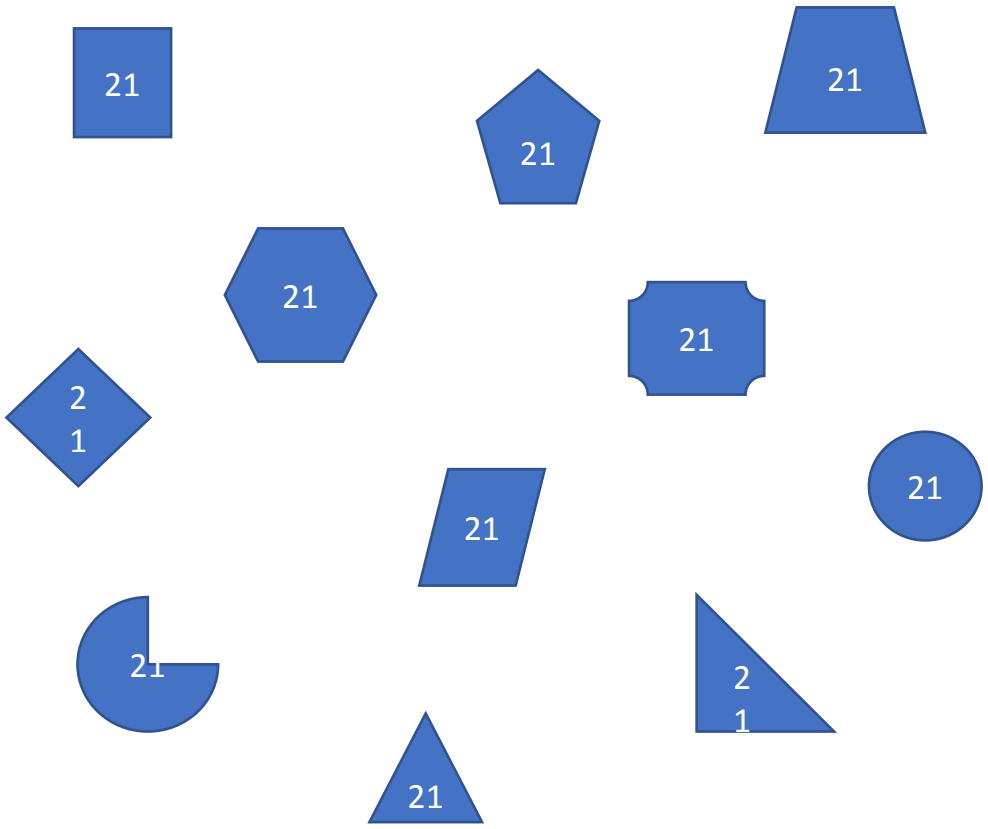
	A
1	Input
2	3
3	5
4	6
5	7

Input = 3 5 6 7 : 1 cluster



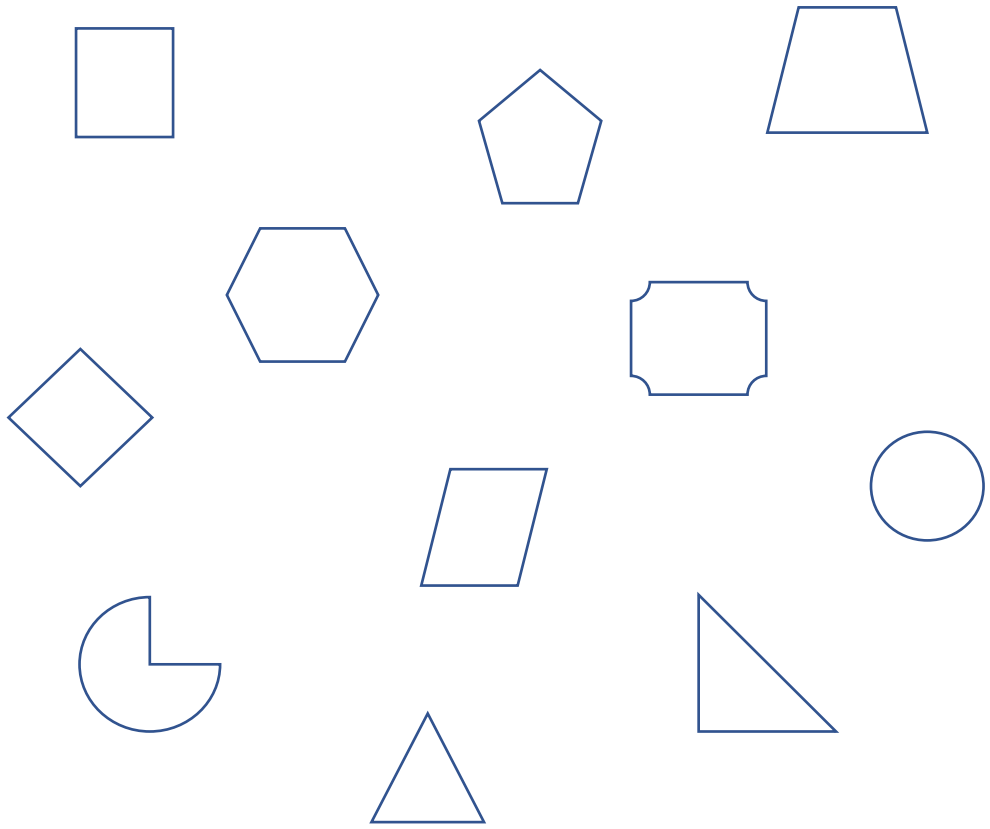
	A
1	**Input**
2	3
3	5
4	6
5	7

Input = 3 5 6 7 : 1 cluster



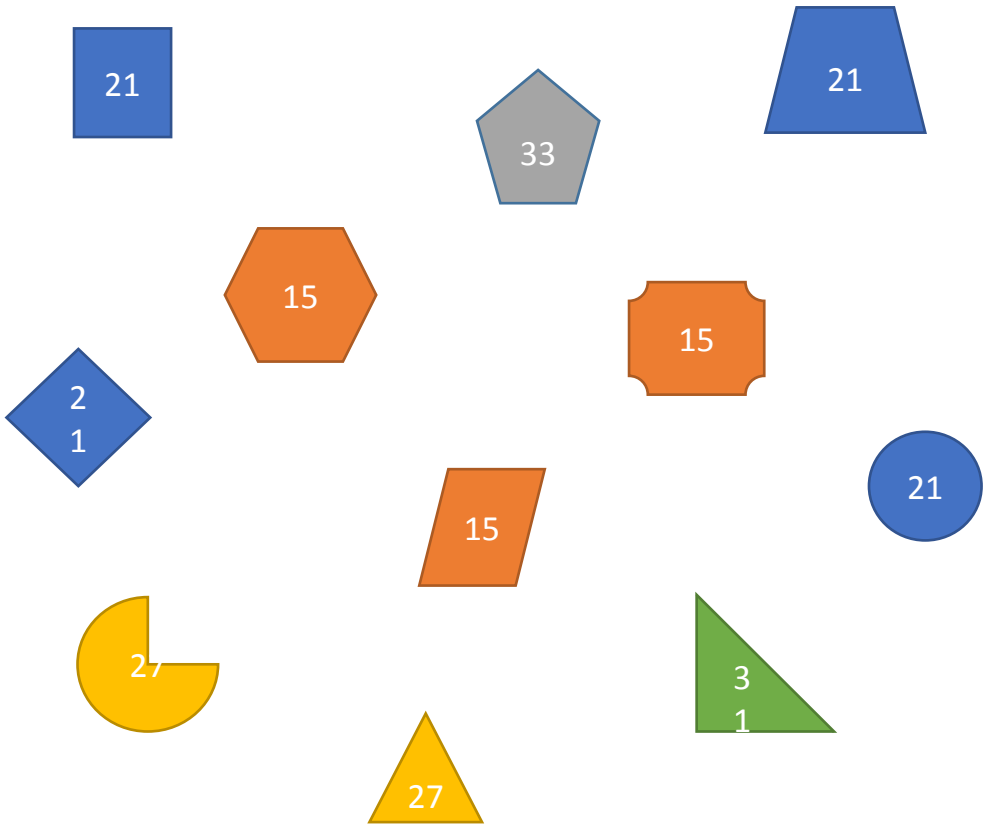
	A
1	**Permuted Input**
2	3
3	7
4	6
5	5

Input = 3 5 6 7 : 1 cluster
Input = 3 7 6 5



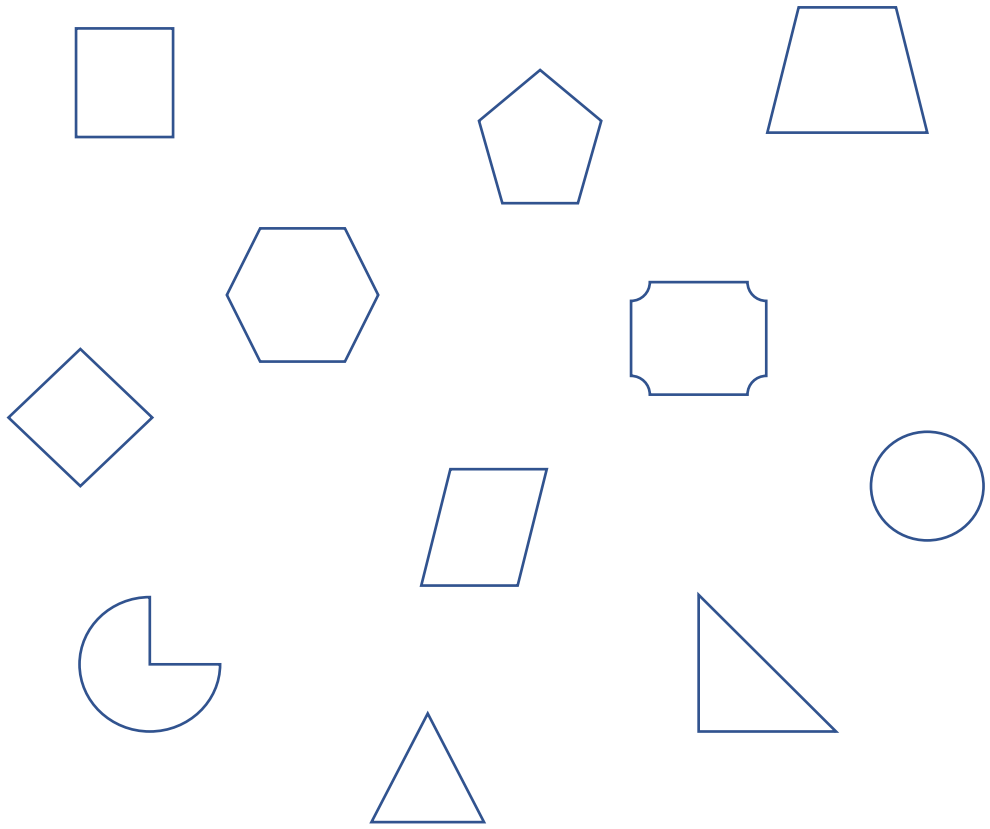
	A
1	**Permuted Input**
2	3
3	7
4	6
5	5

Input = 3 5 6 7 : 1 cluster
Input = 3 7 6 5 : 5 clusters



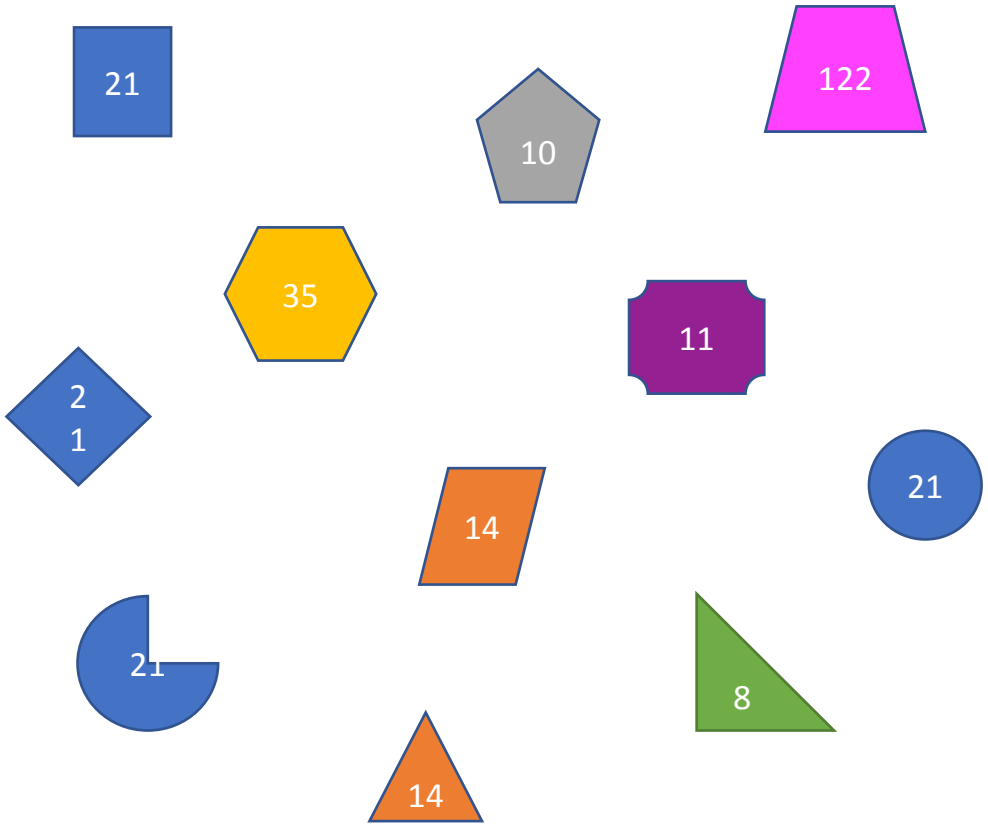
	A
1	**Permuted Input**
2	5
3	6
4	3
5	7

Input = 3 5 6 7 : 1 cluster
Input = 3 7 6 5 : 5 clusters
Input = 5 6 3 7



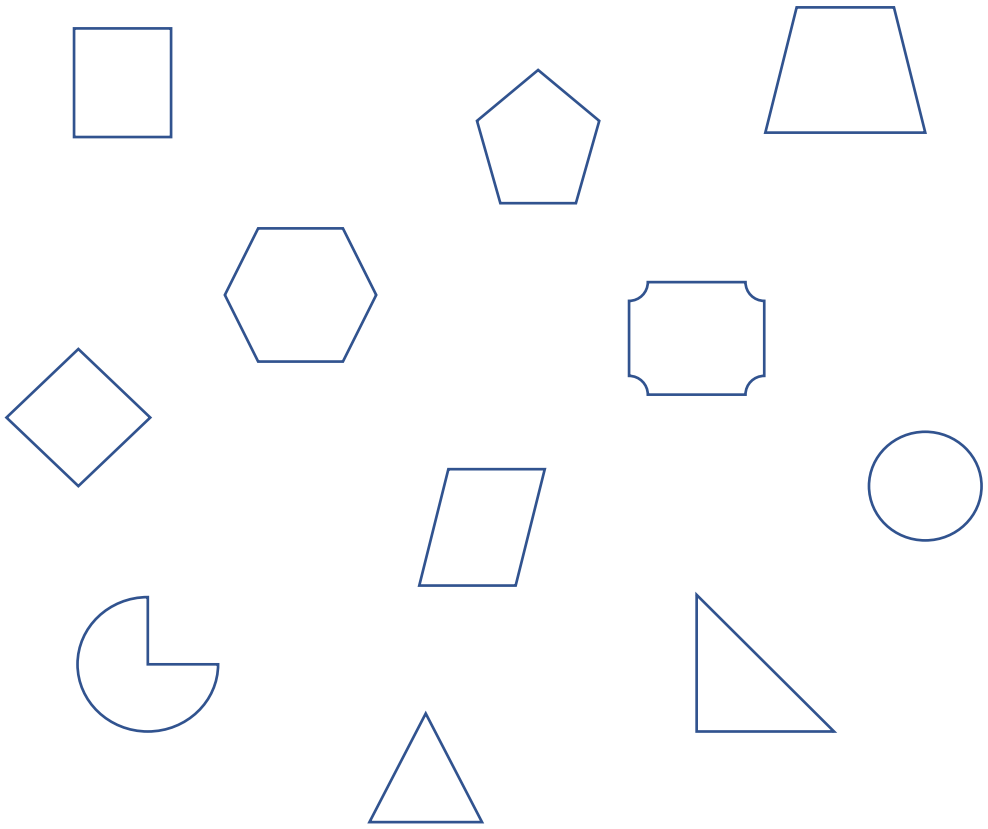
	A
1	**Permuted Input**
2	5
3	6
4	3
5	7

Input = 3 5 6 7 : 1 cluster
Input = 3 7 6 5 : 5 clusters
Input = 5 6 3 7 : 7 clusters



	A
1	**Permuted Input**
2	3
3	6
4	7
5	5

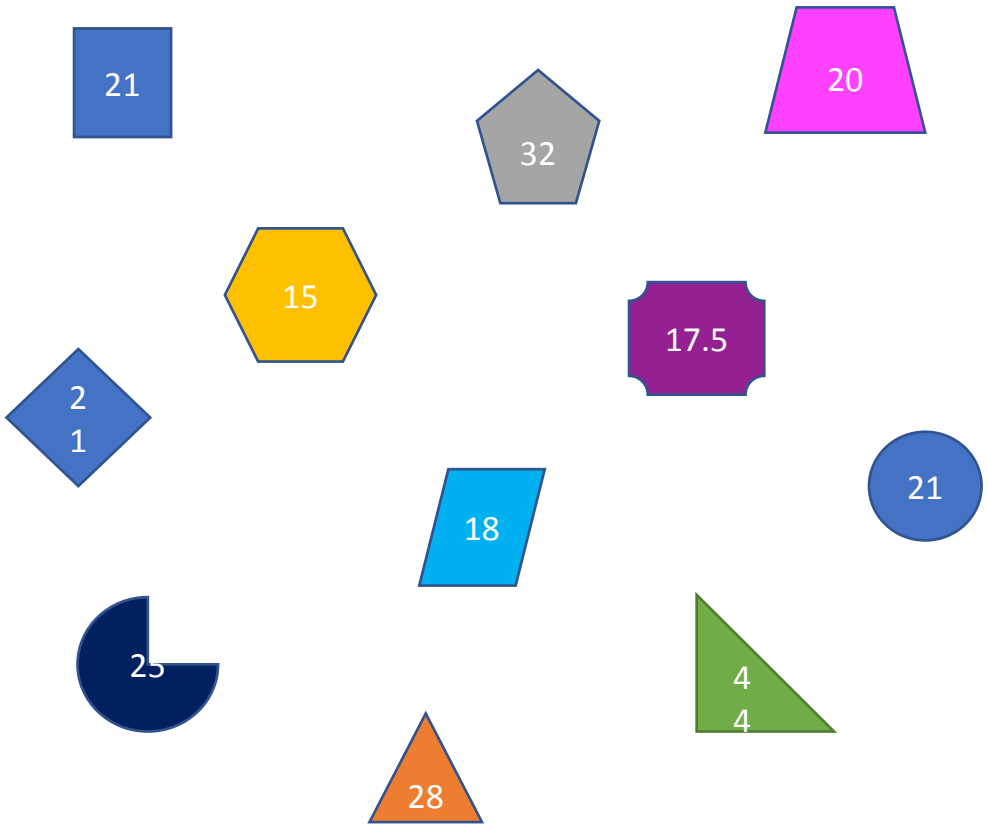
Input = 3 5 6 7 : 1 cluster
Input = 3 7 6 5 : 5 clusters
Input = 5 6 3 7 : 7 clusters
Input = 3 6 7 5



	A
1	**Permuted Input**
2	3
3	6
4	7
5	5

Input = 3 5 6 7 : 1 cluster
Input = 3 7 6 5 : 5 clusters
Input = 5 6 3 7 : 7 clusters

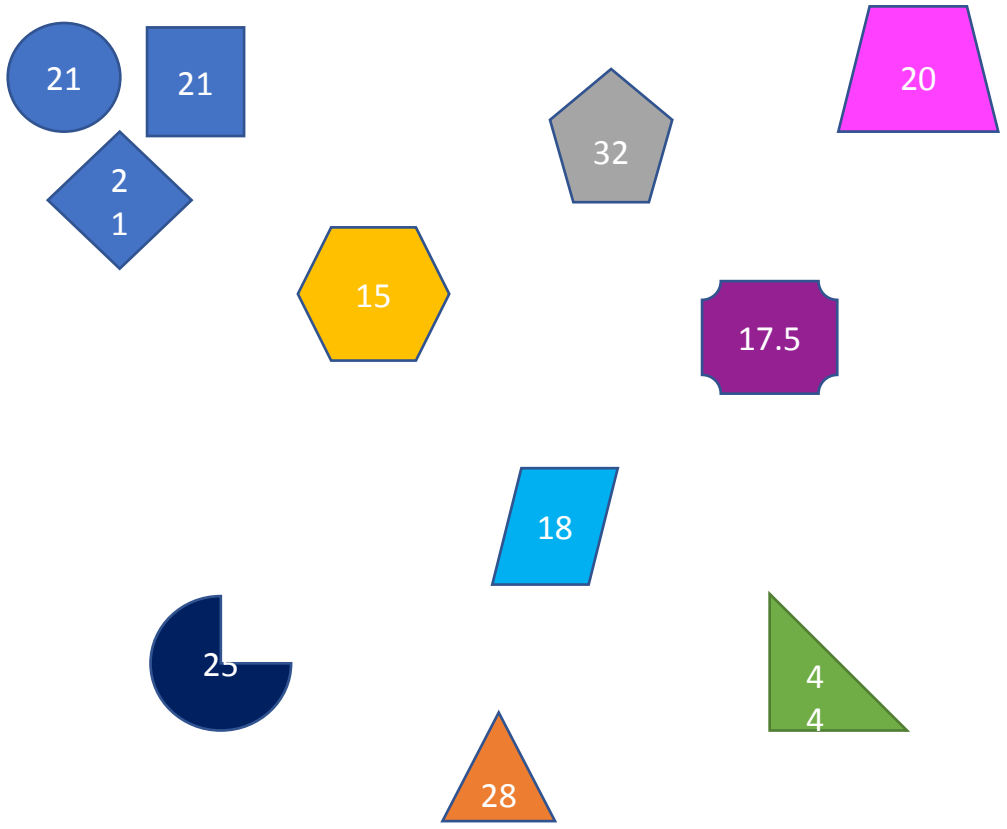
Input = 3 6 7 5 : 9 clusters



Ask user to choose the output

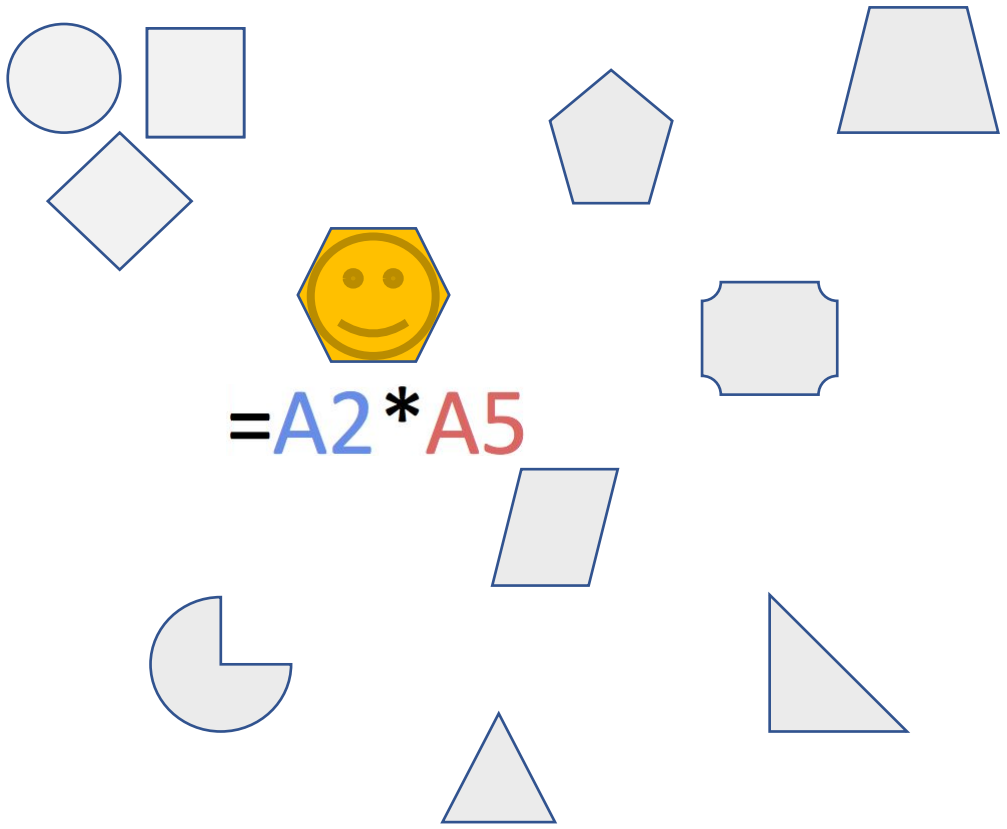
	A
1	**“Best” Input**
2	3
3	6
4	7
5	5

Input = 3 5 6 7 : 1 cluster
Input = 3 7 6 5 : 5 clusters
Input = 5 6 3 7 : 7 clusters
Input = 3 6 7 5 : 9 clusters



	A
1	**“Best” Input**
2	3
3	6
4	7
5	5

Input = 3 5 6 7 : 1 cluster
Input = 3 7 6 5 : 5 clusters
Input = 5 6 3 7 : 7 clusters
Input = 3 6 7 5 : 9 clusters



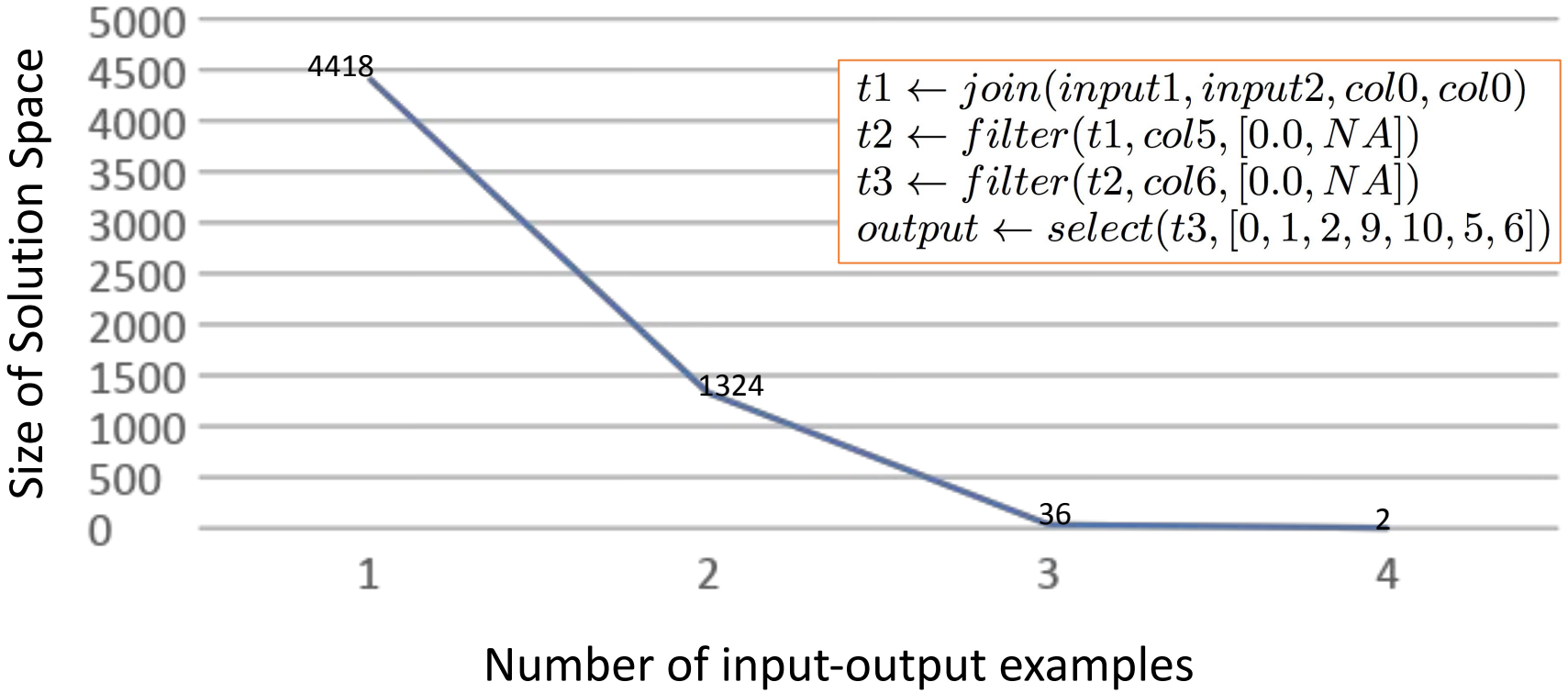
Even with a more complex example, the solution space reduces quickly

bus_id	name	address	city	state	latitude	longitude	phone
16441	"HAWAIIAN DRIVE"	"2600 SAN BRUNO AVE"	"SFO"	"CA"	NA	-122.404101	NA
61073	"FAT ANGEL"	"1740 O' FARRELL ST "	"SFO"	"CA"	0.0	-122.433243	NA
66793	"CP - ROOM D14"	" CANDLESTICK PARK "	"SFO"	"CA"	37.712613	-122.387477	NA
1747	"NARA SUSHI"	"1515 POLK ST "	"SFO"	"CA"	37.790716	NA	NA
509	"CAFE BAKERY"	"1365 NORIEGA ST "	"SFO"	"CA"	37.754090	0.0	NA

bus_id	Score	date
509	85	20130506
1747	93	20121204
16441	94	20130424
61073	98	20130422
66793	100	20130112

bus_id	name	address	Score	date	latitude	longitude
66793	"CP - ROOM D14"	" CANDLESTICK PARK "	100	20130112	37.712613	-122.387477

Solution Space Reduction on Example



Next Step: integrate these ideas into program repair for semi-automated patch selection

Limitations / Next Steps

- Patches are only as sophisticated as semantic search can encode
- We need smart organization of the DB for efficient search
- Other quality metrics that don’t depend on a held-out test suite?

Acknowledgements

- [2018-2023] NSF CAREER #1749936: On the Foundations of Semantic Code Search
- [2016-2020] NSF SHF Medium #1645136: Collaborative Research: Semi and Fully Automated Program Repair and Synthesis via Semantic Code Search
- [2014-2016] NSF SHF EAGER #1446932: Collaborative Research: Demonstrating the Feasibility of Automatic Program Repair Guided by Semantic Code Search.







Bonus

Why Not Execute the Programs?

```
1  private String getPart(String s, char c) {
2    int index = s.lastIndexOf(c);
3    return s.substring(0, index);
4  }
```

Input	Output	Result	
“log.txt”	“log”	sat	, $c \mapsto$ ‘.’

ManyBugs

			test suite				
program	**kLOC**	defects	**count**	format	median LOC	stmt coverage	description
fbc	**97**	3	**483**	BASIC	35	80%	legacy language compiler
gmp	**145**	2	**146**	C	117	64%	multi-precision math library
gzip	**491**	5	**12**	Bash	25	34%	data compression utility
libtiff	**77**	24	**78**	Bash/C	7	25%	image processing library
lighttpd	**62**	9	**295**	Perl	106	62%	web server
php	**1099**	104	**8471**	phpt	35	80%	web programming language
python	**407**	15	**355**	Python	203	67%	general-purpose language
walgrind	**793**	15	**565**	vgtest/C	40	62%	dynamic debugging tool
wireshark	**2814**	8	**63**	Bash	---	44%	network packet analyzer
Total	**5985**	**185**	**10468**				