Fizzer with Local Space Fuzzing

Martin Jonáš, Jan Strejček, Marek Trtík
Masaryk University, Brno, Czechia

May 8, 2025

Main Approach of Fizzer

Main approach

- fuzzer focusing on branch coverage
- instrument all atomic Boolean expressions with tracking distance to the oposite value (e.g., `x - y` for `x <= y`)
- when trying to cover the opposite value, minimize the distance by gradient descent
- also try to cover values by random local search

Main idea of Fizzer

```
char x = read_char();
char y = read_char();
char z = read_char();
char v = read_char();

if (x == 'F') {
  if (y == 'A') {
    if (z == 'S') {
      if (v == 'E') {
          publish_paper();
      }
    }
  }
}
```

Main idea of Fizzer

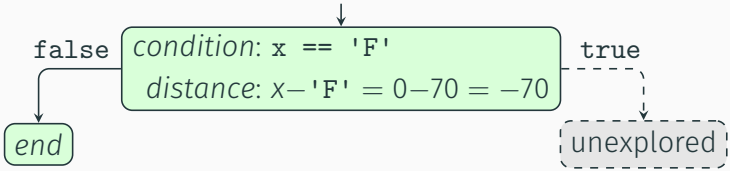
```
1  char x = read_char();
2  char y = read_char();
3  char z = read_char();
4  char v = read_char();
5
6  if (x == 'F') {
7    if (y == 'A') {
8      if (z == 'S') {
9        if (v == 'E') {
10           publish_paper();
11       }
12     }
13   }
14 }
```

- consider input $i_1 = 0, i_2 = 0, i_3 = 0, i_4 = 0$

Main idea of FIZZER

```
1  char x = read_char();
2  char y = read_char();
3  char z = read_char();
4  char v = read_char();
5
6  if (x == 'F') {
7    if (y == 'A') {
8      if (z == 'S') {
9        if (v == 'E') {
10           publish_paper();
11       }
12     }
13   }
14 }
```

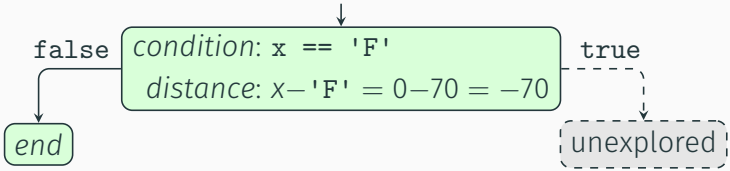
- consider input $i_1 = 0, i_2 = 0, i_3 = 0, i_4 = 0$



Main idea of FIZZER

```
1  char x = read_char();
2  char y = read_char();
3  char z = read_char();
4  char v = read_char();
5
6  if (x == 'F') {
7    if (y == 'A') {
8      if (z == 'S') {
9        if (v == 'E') {
10           publish_paper();
11       }
12     }
13   }
14 }
```

- consider input $i_1 = 0, i_2 = 0, i_3 = 0, i_4 = 0$



1. minimize $|distance| = |x - \texttt{'F'}|$ using gradient descent
2. get input $i_1 = 70, i_2 = 0, i_3 = 0, i_4 = 0$
3. run it, extend the tree, and repeat

Atomic Boolean expressions

```
1       bool b = x > 3;
2       // some code
3       if (b) {
4           // do something
5       }
```

What is the distance for `b`?

Atomic Boolean expressions

```
1    bool b = x > 3;
2    // some code
3    if (b) {
4        // do something
5    }
```

What is the distance for `b`?

Solution

- do not focus on covering branching instructions, but focus on covering both values of all atomic Boolean expressions (ABEs)
- ABEs create new Boolean values from non-Boolean arguments, potentially used later for branching
- in the above program: `x > 3`

Sensitivity analysis

- for a program path and ABE, compute the sensitive bytes of the input
- these are input bytes that can affect the distance and are changed during gradient descent / random local search

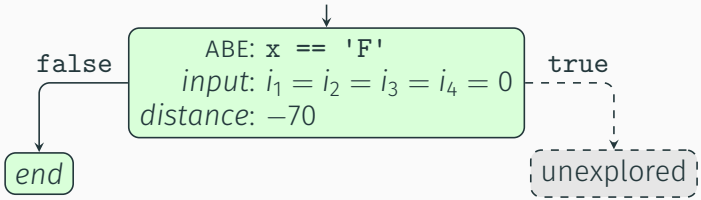
Original sensitivity analysis

```
char x = read_char();
char y = read_char();
char z = read_char();
char v = read_char();

if (x == 'F') {
  if (y == 'A') {
    if (z == 'S') {
      if (v == 'E') {
        publish_paper();
      }
    }
  }
}
```

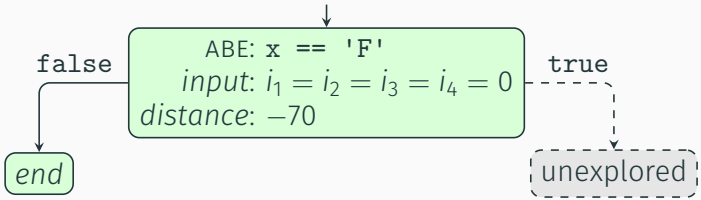
Original sensitivity analysis

```
1  char x = read_char();
2  char y = read_char();
3  char z = read_char();
4  char v = read_char();
5
6  if (x == 'F') {
7    if (y == 'A') {
8      if (z == 'S') {
9        if (v == 'E') {
10         publish_paper();
11       }
12     }
13   }
14 }
```



Original sensitivity analysis

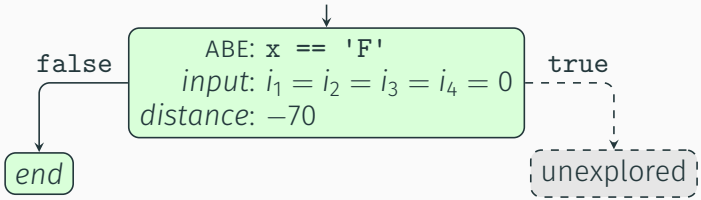
```
1  char x = read_char();
2  char y = read_char();
3  char z = read_char();
4  char v = read_char();
5
6  if (x == 'F') {
7    if (y == 'A') {
8      if (z == 'S') {
9        if (v == 'E') {
10         publish_paper();
11       }
12     }
13   }
14 }
```



1. execute $i_1 = 128$, $i_2 = 0$, $i_3 = 0$, $i_4 = 0$
   distance $128 - 70 = 58 \rightarrow i_1$ sensitive

Original sensitivity analysis

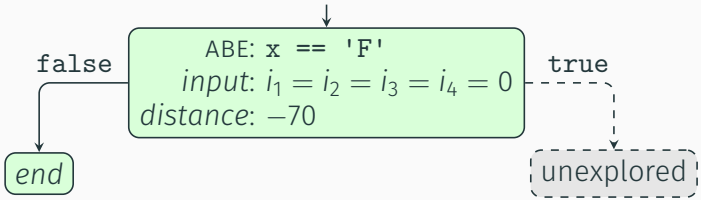
```
1  char x = read_char();
2  char y = read_char();
3  char z = read_char();
4  char v = read_char();
5
6  if (x == 'F') {
7    if (y == 'A') {
8      if (z == 'S') {
9        if (v == 'E') {
10         publish_paper();
11       }
12     }
13   }
14 }
```



1. execute $i_1 = 128$, $i_2 = 0$, $i_3 = 0$, $i_4 = 0$
   distance $128 - 70 = 58 \rightarrow i_1$ sensitive
2. execute $i_1 = 0$, $i_2 = 128$, $i_3 = 0$, $i_4 = 0$
   distance $0 - 70 = -70 \rightarrow$ continue

Original sensitivity analysis

```
1  char x = read_char();
2  char y = read_char();
3  char z = read_char();
4  char v = read_char();
5
6  if (x == 'F') {
7    if (y == 'A') {
8      if (z == 'S') {
9        if (v == 'E') {
10         publish_paper();
11       }
12     }
13   }
14 }
```



1. execute $i_1 = 128$, $i_2 = 0$, $i_3 = 0$, $i_4 = 0$
   distance $128 - 70 = 58 \rightarrow i_1$ sensitive
2. execute $i_1 = 0$, $i_2 = 128$, $i_3 = 0$, $i_4 = 0$
   distance $0 - 70 = -70 \rightarrow$ continue
3. execute $i_1 = 0$, $i_2 = 64$, $i_3 = 0$, $i_4 = 0$
   distance $0 - 70 = -70 \rightarrow$ continue
4. ...

What Is New

New sensitivity analysis

Problems of the original sensitivity analysis

- requires too many executions (potentially one for each input bit)

Solution

- replaced by dynamic taint analysis
- each input is assigned a taint
- taints are propagated through instructions
- based on taints in the ABE, we know the sensitive inputs

Problem

- gradient descent and local search can diverge from the current path

```
1  char x = read_char();
2  char y = read_char();
3
4  if (x == y) {
5      if (x == 42) {
6          doSomething()
7      }
8  }
```

- input $\texttt{x} = 0$, $\texttt{y} = 0$
- covers `true` value of `x == y`
- covers `false` value of `x == 42`
- any attempt to cover `true` value of `x == 42` by changing its sensitive bytes (i.e., `x`) does not reach call `doSomething()`

Search in Local Spaces

Solution

- search in the local space of the last ABE
- try to modify the values of the other variables to preserve previous ABEs (based on the gradients in the ABE)

Example

```
1  char x = read_char();
2  char y = read_char();
3
4  if (x == y) {
5      if (x == 42) {
6          doSomething()
7      }
8  }
```

- input $\mathtt{x} = 0$, $\mathtt{y} = 0$
- local space is given by the constraint $\mathtt{x} - \mathtt{y} = 0$
- when $\mathtt{x}$ is changed to 42 by gradient descent step, change $\mathtt{y}$ to 42
- reaches `doSomething()` call

Test-Comp Results

Fizzer in Test-Comp 2025

- 4th place Cover-Error
- 4th place Cover-Branches
- 3th place Overall

Implementation

Fizzer

- works LLVM-IR representation of the program
- open source
- in C++
- no external dependencies (except LLVM)
- available from

`https://github.com/staticafi/sbt-fizzer/`

Implementation

Fizzer

- works LLVM-IR representation of the program
- open source
- in C++
- no external dependencies (except LLVM)
- available from

`https://github.com/staticafi/sbt-fizzer/`

Thank you for your attention