Specification-less Semantic Bug Detection

Zhendong Su

ETH Zurich
What is the key mission of Computer Science?
To help people turn creative ideas into working systems
Software research is central to this mission
A lot of progress to celebrate for!
To help people turn creative ideas into working systems
P \models \varphi?
$P \models \varphi ?$

but ... where is $\varphi$ ?
THE SOFTWARE CRISIS
THE SOFTWARE
CRISIS

SPECIFICATION
A dilemma

Need $\varphi$ to show $P \models \varphi$
A dilemma

Need $\varphi$ to show $P \models \varphi$

... but $\varphi$ is not available
A dilemma

Need $\varphi$ to show $P \models \varphi$

... but $\varphi$ is not available

One of the greatest challenges!
A dilemma

Need $\varphi$ to show $P \models \varphi$

... but $\varphi$ is not available

practical & technical

One of the greatest challenges!
3 mitigation examples

- Validating compilers
- Validating database engines
- Validating object detection systems
3 mitigation examples

- Validating compilers
- Validating database engines
- Validating object detection systems

All critical infrastructures & specification-less validation
Validate Production Compilers
Compiler complexity

LoC (million)

- LLVM: 4
- GCC: 15
- Linux: 19
LLVM bug 14972

```c
struct tiny { char c; char d; char e; }; 

void foo(struct tiny x) {
  if (x.c != 1) abort();
  if (x.e != 1) abort();
}

int main() {
  struct tiny s;
  s.c = 1; s.d = 1; s.e = 1;
  foo(s);
  return 0;
}
```

$ clang -m32 -O0 test.c ; ./a.out
$ clang -m32 -O1 test.c ; ./a.out
Aborted (core dumped)
Developer comment

“... very, very concerning when I got to the root cause, and very annoying to fix ...”

http://llvm.org/bugs/show_bug.cgi?id=14972
Vision

\[ P \equiv P_1 P_2 P_3 \ldots P_n \]
Key challenges

- **Generation**
  - How to generate *different*, yet *equivalent* tests?

- **Validation**
  - How to check that tests are *indeed equivalent*?

- Both are long-standing hard issues
Equiv. modulo inputs

- Relax equiv. wrt a given input \( i \)
  - Must: \( P(i) = P_k(i) \) on input \( i \)
  - Okay: \( P(j) \neq P_k(j) \) on all input \( j \neq i \)

- Exploit close interplay between
  - Dynamic program execution on some input
  - Static compilation for all input
Profile

input \( I \)

program \( P \)

output \( O \)
Mutate
equivalent modulo $I$
Find bugs
Revisit challenges

- **Generation (easy)**
  - How to generate *different*, yet *equivalent* tests?

- **Validation (easy)**
  - How to check that tests are *indeed equivalent*?
LLVM bug 14972

```c
struct tiny { char c; char d; char e; }

void foo(struct tiny x) {
  if (x.c != 1) abort();
  if (x.e != 1) abort();
}

int main() {
  struct tiny s;
  s.c = 1; s.d = 1; s.e = 1;
  foo(s);
  return 0;
}
```

$ clang -m32 -O0 test.c ; ./a.out
$ clang -m32 -O1 test.c ; ./a.out
Aborted (core dumped)
struct tiny { char c; char d; char e; }; 

f(int n, struct tiny x, struct tiny y, 
    struct tiny z, long l) { 
    if (x.c != 10) abort();
    if (x.d != 20) abort();
    if (x.e != 30) abort();
    if (y.c != 11) abort();
    if (y.d != 21) abort();
    if (y.e != 31) abort();
    if (z.c != 12) abort();
    if (z.d != 22) abort();
    if (z.e != 32) abort();
    if (l != 123) abort();
} 

main() { 
    struct tiny x[3];
    x[0].c = 10;
    x[1].c = 11;
    x[2].c = 12;
    x[0].d = 20;
    x[1].d = 21;
    x[2].d = 22;
    x[0].e = 30;
    x[1].e = 31;
    x[2].e = 32;
    f(3, x[0], x[1], x[2], (long)123);
    exit(0);
} 

$ clang -m32 -O0 test.c ; ./a.out
$ clang -m32 -O1 test.c ; ./a.out
Seed file

```c
struct tiny { char c; char d; char e; };  
f(int n, struct tiny x, struct tiny y, struct tiny z, long l) {
    if (x.c != 10) abort();
    if (x.d != 20) abort();
    if (x.e != 30) abort();
    if (y.c != 11) abort();
    if (y.d != 21) abort();
    if (y.e != 31) abort();
    if (z.c != 12) abort();
    if (z.d != 22) abort();
    if (z.e != 32) abort();
    if (l != 123) abort();
}
main() {
    struct tiny x[3];
    x[0].c = 10;
    x[1].c = 11;
    x[2].c = 12;
    x[0].d = 20;
    x[1].d = 21;
    x[2].d = 22;
    x[0].e = 30;
    x[1].e = 31;
    x[2].e = 32;
    f(3, x[0], x[1], x[2], (long)123);
    exit(0);
}
```

```
$ clang -m32 -O0 test.c ; ./a.out
$ clang -m32 -O1 test.c ; ./a.out
```
### Transformed file

```c
struct tiny { char c; char d; char e; }

f(int n, struct tiny x, struct tiny y, struct tiny z, long l) {
    if (x.c != 10) /* deleted */;
    if (x.d != 20) abort();
    if (x.e != 30) /* deleted */;
    if (y.c != 11) abort();
    if (y.d != 21) abort();
    if (y.e != 31) /* deleted */;
    if (z.c != 12) abort();
    if (z.d != 22) /* deleted */;
    if (z.e != 32) abort();
    if (l != 123) /* deleted */;
}

main() {
    struct tiny x[3];
    x[0].c = 10;
    x[1].c = 11;
    x[2].c = 12;
    x[0].d = 20;
    x[1].d = 21;
    x[2].d = 22;
    x[0].e = 30;
    x[1].e = 31;
    x[2].e = 32;
    f(3, x[0], x[1], x[2], (long)123);
    exit(0);
}
```

```bash
$ clang -m32 -O0 test.c ; ./a.out
$ clang -m32 -O1 test.c ; ./a.out
Aborted (core dumped)
```
Reduced file

```c
struct tiny { char c; char d; char e; };

void foo(struct tiny x) {
    if (x.c != 1) abort();
    if (x.e != 1) abort();
}

int main() {
    struct tiny s;
    s.c = 1; s.d = 1; s.e = 1;
    foo(s);
    return 0;
}
```

$ clang -m32 -O0 test.c ; ./a.out
$ clang -m32 -O1 test.c ; ./a.out
Aborted (core dumped)
LLVM bug autopsy

```c
struct tiny { char c; char d; char e; }; 

void foo(struct tiny x) {
    if (x.c != 1) abort();
    if (x.e != 1) abort();
}

int main() {
    struct tiny s;
    s.c = 1; s.d = 1; s.e = 1;
    foo(s);
    return 0;
}
```

$ clang -m32 -O0 test.c ; ./a.out
$ clang -m32 -O1 test.c ; ./a.out
Aborted (core dumped)
LLVM bug autopsy

```c
struct tiny { char c; char d; char e; };

void foo(struct tiny x) {
    if (x.c != 1) abort();
    if (x.e != 1) abort();
}

int main() {
    struct tiny s;
    s.c = 1; s.d = 1; s.e = 1;
    foo(s);
    return 0;
}
```

$ clang -m32 -O0 test.c ; ./a.out
$ clang -m32 -O1 test.c ; ./a.out
Aborted (core dumped)

GVN: load struct using 32-bit load
SRoA: read past the struct’s end
undefined behavior
LLVM bug autopsy

```c
struct tiny { char c; char d; char e; };

void foo(struct tiny x) {
    if (x.c != 1) abort();
    if (x.e != 1) abort();
}

int main() {
    struct tiny s;
    s.c = 1; s.d = 1; s.e = 1;
    foo(s);
    return 0;
}
```

$ clang -m32 -O0 test.c ; ./a.out
$ clang -m32 -O1 test.c ; ./a.out
Aborted (core dumped)

GVN: load struct using 32-bit load
SRoA: read past the struct’s end
undefined behavior
GCC bug 58731

```c
int a, b, c, d, e;
int main() {
    for (b = 4; b > -30; b--)
        for (; c;)
            for (; ;) {
                e = a > 2147483647 - b;
                if (d) break;
            }
    return 0;
}
```

```bash
$ gcc -O0 test.c ; ./a.out
$ gcc -O3 test.c ; ./a.out
^C
```
GCC bug autopsy

```c
int a, b, c, d, e;
int main() {
    for (b = 4; b > -30; b--) {
        for (; c;)
            for (; ;) {
                e = a > 2147483647 - b;
                if (d) break;
            }
    return 0;
}
```

$ gcc -O0 test.c ; ./a.out
$ gcc -O3 test.c ; ./a.out
^C
GCC bug autopsy

```c
int a, b, c, d, e;
int main() {
    for (b = 4; b > -30; b--)
        int f = 2147483647 - b;
    for (; c;)
        for (; ;) {
            e = a > f;
            if (d) break;
        }
    return 0;
}
```

$ gcc -O0 test.c ; ./a.out
$ gcc -O3 test.c ; ./a.out
^C
GCC bug autopsy

```c
int a, b, c, d, e;
int main() {
    for (b = 4; b > -30; b--)
        int f = 2147483647 - b;
    for (; c;)
        for (; ;) {
            e = a > f;
            if (d) break;
        }
    return 0;
}
```

`$ gcc -O0 test.c ; ./a.out`

`$ gcc -O3 test.c ; ./a.out`

^C
Seed program

```c
int a, b, c, d, e;
int main() {
    for (b = 4; b > -30; b--)
        for (; c;)
            for (; ;)
                b++;
                e = a > 2147483647 - b;
                if (d) break;
    return 0;
}
```

$ gcc -O0 test.c ; ./a.out
$ gcc -O3 test.c ; ./a.out

no longer a loop invariant
Athena (OOPSLA’15)
Prune & inject dead code

Orion (PLDI’14)
Prune dead code

Hermes (OOPSLA’16)
Mutate live code
## Bug counts

<table>
<thead>
<tr>
<th></th>
<th>GCC</th>
<th>LLVM</th>
<th>TOTAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reported</td>
<td>841</td>
<td>781</td>
<td>1,622</td>
</tr>
<tr>
<td>Fixed</td>
<td>612</td>
<td>419</td>
<td>1,031</td>
</tr>
</tbody>
</table>

- **ISSTA’15**: Stress-testing link-time optimization
- **ICSE’16**: Analyzing compilers’ diagnostic support
- **PLDI’17**: Skeletal program enumeration (SPE)
LLVM 3.9 & 4.0 Release Notes

“... thanks to Zhendong Su and his team whose fuzz testing prevented many bugs going into the release ...”
GCC’s list of contributors

https://gcc.gnu.org/onlinedocs/gcc/Contributors.html

“Zhendong Su ... for reporting numerous bugs”
“Chengnian Sun ... for reporting numerous bugs”
“Qirun Zhang ... for reporting numerous bugs”
Validate Database Engines
Database engines (DBMS)

Who has heard about/used these Database Management Systems?
DBMS widely used

“SQLite is the most used database engine in the world. SQLite is built into all mobile phones and most computers and comes bundled inside countless other applications that people use every day.”

https://www.sqlite.org
# Relational Data Model

## animal_pictures

<table>
<thead>
<tr>
<th>animal</th>
<th>description</th>
<th>picture</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cat</td>
<td>A cute toast cat</td>
<td><img src="image1.png" alt="Cat picture" /></td>
</tr>
<tr>
<td>Dog</td>
<td>Cute dog pic</td>
<td><img src="image2.png" alt="Dog picture" /></td>
</tr>
<tr>
<td>Cat</td>
<td>Cat plants (cute!)</td>
<td><img src="image3.png" alt="Cat picture" /></td>
</tr>
</tbody>
</table>
Relational Data Model

A database schema describes the **tables (relations)** in the database.

<table>
<thead>
<tr>
<th>animal</th>
<th>description</th>
<th>picture</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cat</td>
<td>A cute toast cat</td>
<td><img src="image1" alt="Cat" /></td>
</tr>
<tr>
<td>Dog</td>
<td>Cute dog pic</td>
<td><img src="image2" alt="Dog" /></td>
</tr>
<tr>
<td>Cat</td>
<td>Cat plants (cute!)</td>
<td><img src="image3" alt="Cat" /></td>
</tr>
</tbody>
</table>
Structured Query Language (SQL) is a **declarative** DSL to **query and manipulate data**.

```
SELECT picture, description
FROM animal_pictures
WHERE animal = 'Cat'
    AND description LIKE '%cute%'
```
SELECT * FROM <table>
WHERE <cond>
Goal

Aim: Detect **logic bugs** in DBMS
SELECT * FROM <table> 
WHERE <cond>

Client Application  ➔ Database Management System (DBMS) ➔ Database

- row_1 <cond>
- row_2 <cond>
- row_3 ¬<cond>

- row_1 <cond> ✔️
SELECT * FROM <table>
WHERE <cond>

Client Application ↔ Database Management System (DBMS) ↔ Database

row1 <cond> row2 <cond> row3 ¬<cond>
Example SQLite bug

CREATE TABLE t1(c1, c2, c3, c4, PRIMARY KEY (c4, c3));
INSERT INTO t1(c3) VALUES (0), (0), (0), (0), (0), (0), (0), (0), (0), (0), (NULL), (1), (0);
UPDATE t1 SET c2 = 0;
INSERT INTO t1(c1) VALUES (0), (0), (NULL), (0), (0);
ANALYZE t1;
UPDATE t1 SET c3 = 1;
SELECT DISTINCT * FROM t1 WHERE t1.c3 = 1;

ANALYZE gathers statistics about tables, which are then used for query planning.
Example SQLite bug

CREATE TABLE t1(c1, c2, c3, c4, PRIMARY KEY (c4, c3));
INSERT INTO t1(c3) VALUES (0), (0), (0), (0), (0), (0), (0), (0), (0), (0), (NULL), (1), (0);
UPDATE t1 SET c2 = 0;
INSERT INTO t1(c1) VALUES (0), (0), (NULL), (0), (0);
ANALYZE t1;
UPDATE t1 SET c3 = 1;
SELECT DISTINCT * FROM t1 WHERE t1.c3 = 1;

A bug in the skip-scan optimization caused this logic bug.
DBMS (very) well-tested

SQLite (~150,000 LOC) has 662 times as much test code as source code.

SQLite is extensively fuzzed (e.g., by Google’s OS-Fuzz Project).

SQLite’s test cases achieve 100% branch test coverage.

SQLite’s performs anomaly testing (out-of-memory, I/O error, power failures).


https://www.sqlite.org/testing.html
Pivoted Query Synthesis (PQS)

>100 bugs in widely used DBMS
PQS idea

<table>
<thead>
<tr>
<th>$\text{Column}_0$</th>
<th>$\text{Column}_1$</th>
<th>$\text{Column}_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\ldots$</td>
<td>$\ldots$</td>
<td>$\ldots$</td>
</tr>
<tr>
<td>$\text{Value}_{i,0}$</td>
<td>$\text{Value}_{i,1}$</td>
<td>$\text{Value}_{i,2}$</td>
</tr>
<tr>
<td>$\ldots$</td>
<td>$\ldots$</td>
<td>$\ldots$</td>
</tr>
<tr>
<td><strong>Pivot Row</strong></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Database generation

To explore “all possible database states” we randomly create databases

<table>
<thead>
<tr>
<th>animal</th>
<th>description</th>
<th>picture</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cat</td>
<td>A cute toast cat</td>
<td><img src="image1.png" alt="Cat Toast Picture" /></td>
</tr>
<tr>
<td>Dog</td>
<td>Cute dog pic</td>
<td><img src="image2.png" alt="Dog Pic" /></td>
</tr>
<tr>
<td>Cat</td>
<td>Cat plants (cute!)</td>
<td><img src="image3.png" alt="Cat with Plant Picture" /></td>
</tr>
</tbody>
</table>
Pivot row selection

Randomly Generate Database

Select Pivot Row

<table>
<thead>
<tr>
<th>animal</th>
<th>description</th>
<th>picture</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cat</td>
<td>A cute toast cat</td>
<td><img src="image1" alt="Cat" /></td>
</tr>
<tr>
<td>Dog</td>
<td>Cute dog pic</td>
<td><img src="image2" alt="Dog" /></td>
</tr>
<tr>
<td>Cat</td>
<td>Cat plants (cute!)</td>
<td><img src="image3" alt="Cat" /></td>
</tr>
</tbody>
</table>
Query generation

```
SELECT picture, description
FROM animal_pictures
WHERE animal = 'Cat'
    AND description LIKE '%cute%'
```
Verifying the result

Randomly Generate Database → Select Pivot Row → Generate Query for the Pivot Row → Verify that the Pivot Row is contained

```
SELECT picture, description
FROM animal_pictures
WHERE animal = 'Cat'
AND description LIKE '%cute%'
```

<table>
<thead>
<tr>
<th>animal</th>
<th>description</th>
<th>picture</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cat</td>
<td>A cute toast cat</td>
<td>![Image]</td>
</tr>
<tr>
<td>Cat</td>
<td>Cat plants (cute!)</td>
<td>![Image]</td>
</tr>
</tbody>
</table>

pivot row ∈ result set
Verifying the result

SELECT picture, description
FROM animal_pictures
WHERE animal = 'Cat'
    AND description LIKE '%cute%'

result set

<table>
<thead>
<tr>
<th>animal</th>
<th>description</th>
<th>picture</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cat</td>
<td>A cute toast cat</td>
<td>![Cat Pic]</td>
</tr>
<tr>
<td>Dog</td>
<td>Cute dog pic</td>
<td>![Dog Pic]</td>
</tr>
</tbody>
</table>

pivot row

<table>
<thead>
<tr>
<th>animal</th>
<th>description</th>
<th>picture</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cat</td>
<td>Cat plants (cute!)</td>
<td>![Cat Pic]</td>
</tr>
</tbody>
</table>

pivot row ∉ result set
Approach

- Randomly Generate Database
- Select Pivot Row
- Generate Query for the Pivot Row
- Verify that the Pivot Row is contained
Approach

Randomly Generate Database -> Select Pivot Row -> Generate Query for the Pivot Row -> Verify that the Pivot Row is contained
Approach

- Randomly Generate Database
- Select Pivot Row
- Generate Query for the Pivot Row
- Verify that the Pivot Row is contained
Approach

Randomly Generate Database → Select Pivot Row → Generate Query for the Pivot Row → Verify that the Pivot Row is contained

How do we generate this query?
How to generate queries?

```
SELECT picture, description
FROM animal_pictures
WHERE
```

Generate an expression that yields TRUE for the pivot row.
How to generate queries?

1. Randomly Generate Expression
2. Evaluate Expression on Pivot Row
3. Modify expression to yield TRUE
4. Use in WHERE clause
Random exp. generation

We first generate a random expression

https://www.sqlite.org/syntax/expr.html

animal_pictures

animal  description  picture

We first generate a random expression
Random exp. generation

\[ \text{animal} = 'Cat' \]
\[ \text{AND description LIKE} \ '\%cute\%' \]
Random exp. generation

```
AND

animal = 'Cat'
AND description LIKE '%cute%'
```

Evaluate the tree based on the pivot row
Random exp. evaluation

AND

animal = 'Cat'

LIKE

description = 'Cute'

Constant nodes return their assigned literal values

<table>
<thead>
<tr>
<th>animal</th>
<th>description</th>
<th>picture</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cat</td>
<td>Cat plants (cute!)</td>
<td></td>
</tr>
</tbody>
</table>
Random exp. evaluation

Column references return the values from the pivot row

<table>
<thead>
<tr>
<th>animal</th>
<th>description</th>
<th>picture</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cat</td>
<td>Cat plants (cute!)</td>
<td>![Picture]</td>
</tr>
</tbody>
</table>
Random exp. evaluation

= 'Cat' 'Cat plants (cute!)'

LIKE 'Cat' '%cute%'

Compound nodes compute their result based on their children

<table>
<thead>
<tr>
<th>animal</th>
<th>description</th>
<th>picture</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cat</td>
<td>Cat plants (cute!)</td>
<td><img src="image" alt="Image" /></td>
</tr>
</tbody>
</table>
Random exp. evaluation

AND

animal = 'Cat' AND 'Cat'

LIKE

description = 'Cat plants (cute!)' AND '%cute%'

<table>
<thead>
<tr>
<th>animal</th>
<th>description</th>
<th>picture</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cat</td>
<td>Cat plants (cute!)</td>
<td></td>
</tr>
</tbody>
</table>
Query synthesis

```sql
SELECT picture, description
FROM animal_pictures
WHERE animal = 'Cat' AND description LIKE '%cute%'
```
Random exp. evaluation

What about when the expression does not evaluate to TRUE?
Random exp. evaluation

What about when the expression does not evaluate to TRUE?

```
animal = 'Dog'
```

![Diagram showing the evaluation of `animal = 'Dog'` resulting in FALSE.](image)

<table>
<thead>
<tr>
<th>animal</th>
<th>description</th>
<th>picture</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cat</td>
<td>Cat plants (cute!)</td>
<td><img src="image" alt="Cat" /></td>
</tr>
</tbody>
</table>
switch (result) {
    case TRUE:
        result = randexpr;
    case FALSE:
        result = NOT randexpr;
    case NULL:
        result = randexpr ISNULL;
}

Random exp. rectification
Random exp. rectification

```java
switch (result) {
    case TRUE:
        result = randexpr;
    case FALSE:
        result = NOT randexpr;
    case NULL:
        result = randexpr ISNULL;
}
```

animal = 'Dog'

<table>
<thead>
<tr>
<th>animal</th>
<th>description</th>
<th>picture</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cat</td>
<td>Cat plants (cute!)</td>
<td><img src="image" alt="picture" /></td>
</tr>
</tbody>
</table>
Random exp. rectification

```java
switch (result) {
    case TRUE:
        result = randexpr;
        break;
    case FALSE:
        result = NOT randexpr;
        break;
    case NULL:
        result = randexpr ISNULL;
        break;
}
```

```
NOT(animal = 'Dog')
```

```
<table>
<thead>
<tr>
<th>animal</th>
<th>description</th>
<th>picture</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cat</td>
<td>Cat plants (cute!)</td>
<td></td>
</tr>
</tbody>
</table>
```
How to generate queries?

```
SELECT picture, description
FROM animal_pictures
WHERE NOT(animal = 'Dog')
```
Tested DBMS

PostgreSQL

We tested these (and other DBMS) in a period of 3-4 months

SQLite

MySQL
<table>
<thead>
<tr>
<th>DBMS</th>
<th>DB-Engines</th>
<th>Stack Overflow</th>
<th>LOC</th>
<th>First Release</th>
<th>Age</th>
</tr>
</thead>
<tbody>
<tr>
<td>SQLite</td>
<td>11</td>
<td>4</td>
<td>0.3M</td>
<td>2000</td>
<td>19 years</td>
</tr>
<tr>
<td>MySQL</td>
<td>2</td>
<td>1</td>
<td>3.8M</td>
<td>1995</td>
<td>24 years</td>
</tr>
<tr>
<td>PostgreSQL</td>
<td>4</td>
<td>2</td>
<td>1.4M</td>
<td>1996</td>
<td>23 years</td>
</tr>
<tr>
<td>DBMS</td>
<td>DB-Engines</td>
<td>Stack Overflow</td>
<td>LOC</td>
<td>First Release</td>
<td>Age</td>
</tr>
<tr>
<td>------------</td>
<td>------------</td>
<td>----------------</td>
<td>-------</td>
<td>---------------</td>
<td>---------</td>
</tr>
<tr>
<td>SQLite</td>
<td>11</td>
<td>4</td>
<td>0.3M</td>
<td>2000</td>
<td>19 years</td>
</tr>
<tr>
<td>MySQL</td>
<td>2</td>
<td>1</td>
<td>3.8M</td>
<td>1995</td>
<td>24 years</td>
</tr>
<tr>
<td>PostgreSQL</td>
<td>4</td>
<td>2</td>
<td>1.4M</td>
<td>1996</td>
<td>23 years</td>
</tr>
<tr>
<td>DBMS</td>
<td>DB-Engines</td>
<td>Stack Overflow</td>
<td>LOC</td>
<td>First Release</td>
<td>Age</td>
</tr>
<tr>
<td>-----------</td>
<td>------------</td>
<td>----------------</td>
<td>------</td>
<td>---------------</td>
<td>-------</td>
</tr>
<tr>
<td>SQLite</td>
<td>11</td>
<td>4</td>
<td>0.3M</td>
<td>2000</td>
<td>19 years</td>
</tr>
<tr>
<td>MySQL</td>
<td>2</td>
<td>1</td>
<td>3.8M</td>
<td>1995</td>
<td>24 years</td>
</tr>
<tr>
<td>PostgreSQL</td>
<td>4</td>
<td>2</td>
<td>1.4M</td>
<td>1996</td>
<td>23 years</td>
</tr>
</tbody>
</table>
# Bugs overview

<table>
<thead>
<tr>
<th>DBMS</th>
<th>Fixed</th>
<th>Verified</th>
</tr>
</thead>
<tbody>
<tr>
<td>SQLite</td>
<td>65</td>
<td>0</td>
</tr>
<tr>
<td>MySQL</td>
<td>15</td>
<td>10</td>
</tr>
<tr>
<td>PostgreSQL</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td><strong>Sum</strong></td>
<td><strong>85</strong></td>
<td><strong>14</strong></td>
</tr>
</tbody>
</table>

**99 real bugs**: code fixes or verified as bugs
## Bugs overview

The SQLite developers quickly responded to all our bug reports → we focused on this DBMS

<table>
<thead>
<tr>
<th>DBMS</th>
<th>Fixed</th>
<th>Verified</th>
</tr>
</thead>
<tbody>
<tr>
<td>SQLite</td>
<td>65</td>
<td>0</td>
</tr>
<tr>
<td>MySQL</td>
<td>15</td>
<td>10</td>
</tr>
<tr>
<td>PostgreSQL</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>Sum</td>
<td>85</td>
<td>14</td>
</tr>
</tbody>
</table>

The SQLite developers **quickly responded** to all our bug reports → we focused on this DBMS
Bugs overview

<table>
<thead>
<tr>
<th>DBMS</th>
<th>Fixed</th>
<th>Verified</th>
</tr>
</thead>
<tbody>
<tr>
<td>SQLite</td>
<td>65</td>
<td>0</td>
</tr>
<tr>
<td>MySQL</td>
<td>15</td>
<td>10</td>
</tr>
<tr>
<td>PostgreSQL</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>Sum</td>
<td>85</td>
<td>14</td>
</tr>
</tbody>
</table>

All MySQL bug reports were **verified quickly**
# Bugs overview

MySQL’s trunk is **unavailable**, and it has a long release cycle

<table>
<thead>
<tr>
<th>DBMS</th>
<th>Fixed</th>
<th>Verified</th>
</tr>
</thead>
<tbody>
<tr>
<td>SQLite</td>
<td>65</td>
<td>0</td>
</tr>
<tr>
<td>MySQL</td>
<td>15</td>
<td>10</td>
</tr>
<tr>
<td>PostgreSQL</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>Sum</td>
<td>85</td>
<td>14</td>
</tr>
</tbody>
</table>

---

Real Bugs
Bugs overview

<table>
<thead>
<tr>
<th>DBMS</th>
<th>Fixed</th>
<th>Verified</th>
</tr>
</thead>
<tbody>
<tr>
<td>SQLite</td>
<td>65</td>
<td>0</td>
</tr>
<tr>
<td>MySQL</td>
<td>15</td>
<td>10</td>
</tr>
<tr>
<td>PostgreSQL</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>Sum</td>
<td>85</td>
<td>14</td>
</tr>
</tbody>
</table>

We found the **fewest bugs in PostgreSQL** and not all could be easily addressed.
## Oracles

<table>
<thead>
<tr>
<th>DBMS</th>
<th>Containment</th>
<th>Error</th>
<th>SEGFAULT</th>
</tr>
</thead>
<tbody>
<tr>
<td>SQLite</td>
<td>46</td>
<td>17</td>
<td>2</td>
</tr>
<tr>
<td>MySQL</td>
<td>14</td>
<td>10</td>
<td>1</td>
</tr>
<tr>
<td>PostgreSQL</td>
<td>1</td>
<td>7</td>
<td>1</td>
</tr>
<tr>
<td>Sum</td>
<td>61</td>
<td>34</td>
<td>4</td>
</tr>
</tbody>
</table>
Oracles

<table>
<thead>
<tr>
<th>DBMS</th>
<th>Containment</th>
<th>Error</th>
<th>SEGFAULT</th>
</tr>
</thead>
<tbody>
<tr>
<td>SQLite</td>
<td>46</td>
<td>17</td>
<td>2</td>
</tr>
<tr>
<td>MySQL</td>
<td>14</td>
<td>10</td>
<td>1</td>
</tr>
<tr>
<td>PostgreSQL</td>
<td>1</td>
<td>7</td>
<td>1</td>
</tr>
<tr>
<td>Sum</td>
<td>61</td>
<td>34</td>
<td>4</td>
</tr>
</tbody>
</table>

Our Containment oracle allowed us to detect most errors.
Result: bug in SQLite3

CREATE TABLE t0(c1 TEXT PRIMARY KEY) WITHOUT ROWID;
CREATE INDEX i0 ON t0(c1 COLLATE NOCASE);
INSERT INTO t0(c1) VALUES ('A');
INSERT INTO t0(c1) VALUES ('a');

An index is an auxiliary data structure that should not affect the query’s result.
CREATE TABLE t0(c1 TEXT PRIMARY KEY) WITHOUT ROWID;
CREATE INDEX i0 ON t0(c1 COLLATE NOCASE);
INSERT INTO t0(c1) VALUES ('A');
INSERT INTO t0(c1) VALUES ('a');
Result: bug in SQLite3

```sql
CREATE TABLE t0(c1 TEXT PRIMARY KEY) WITHOUT ROWID;
CREATE INDEX i0 ON t0(c1 COLLATE NOCASE);
INSERT INTO t0(c1) VALUES ('A');
INSERT INTO t0(c1) VALUES ('a');
```

```
c1
'A'
'a'
```

SELECT * FROM t0;

```
c1
  'A'
```

SQLite failed to fetch 'a'!
CREATE TABLE t0(c0 INT PRIMARY KEY, c1 INT);
CREATE TABLE t1(c0 INT) INHERITS (t0);

<table>
<thead>
<tr>
<th>t0</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>c0</td>
<td>c1</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>t1</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>c0</td>
<td>c1</td>
<td></td>
</tr>
</tbody>
</table>
Result: bug in PostgreSQL

<table>
<thead>
<tr>
<th>t0</th>
<th>c0</th>
<th>c1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

```sql
CREATE TABLE t0(c0 INT PRIMARY KEY, c1 INT);
CREATE TABLE t1(c0 INT) INHERITS (t0);
INSERT INTO t0(c0, c1) VALUES(0, 0);
```

<table>
<thead>
<tr>
<th>t1</th>
<th>c0</th>
<th>c1</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Real Bugs
Containment Oracle

Oracle Result: bug in PostgreSQL
CREATE TABLE t0(c0 INT PRIMARY KEY, c1 INT);
CREATE TABLE t1(c0 INT) INHERITS (t0);
INSERT INTO t0(c0, c1) VALUES(0, 0);
INSERT INTO t1(c0, c1) VALUES(0, 1);

The inheritance relationship causes the row to be inserted both in t0 and t1.

Result: bug in PostgreSQL
CREATE TABLE t0(c0 INT PRIMARY KEY, c1 INT);
CREATE TABLE t1(c0 INT) INHERITS (t0);
INSERT INTO t0(c0, c1) VALUES(0, 0);
INSERT INTO t1(c0, c1) VALUES(0, 1);

<table>
<thead>
<tr>
<th>c0</th>
<th>c1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

SELECT c0, c1 FROM t0
GROUP BY c0, c1;

Real Bugs
Containment Oracle

PostgreSQL failed to fetch the row 0 | 1
Result: bug in MySQL

<table>
<thead>
<tr>
<th>t0</th>
<th>CREATE TABLE t0(c0 TINYINT);</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>INSERT INTO t0(c0) VALUES(NULL);</td>
</tr>
</tbody>
</table>

Real Bugs

Containment
Oracle
CREATE TABLE t0(c0 TINYINT);
INSERT INTO t0(c0) VALUES(NULL);

SELECT * FROM t0
WHERE
NOT(t0.c0 <=> 2035382037);

The MySQL-specific equality operator <=> malfunctioned for large numbers
Oracles

<table>
<thead>
<tr>
<th>DBMS</th>
<th>Containment</th>
<th>Error</th>
<th>SEGFAULT</th>
</tr>
</thead>
<tbody>
<tr>
<td>SQLite</td>
<td>46</td>
<td>17</td>
<td>2</td>
</tr>
<tr>
<td>MySQL</td>
<td>14</td>
<td>10</td>
<td>1</td>
</tr>
<tr>
<td>PostgreSQL</td>
<td>1</td>
<td>7</td>
<td>1</td>
</tr>
<tr>
<td>Sum</td>
<td>61</td>
<td>34</td>
<td>4</td>
</tr>
</tbody>
</table>

We also found many bugs that trigger DB errors
**SQLite3 bug**

```
CREATE TABLE t1 (c0, c1 REAL PRIMARY KEY);
INSERT INTO t1(c0, c1) VALUES
(TRUE, 9223372036854775807), (TRUE, 0);
UPDATE t1 SET c0 = NULL;
UPDATE OR REPLACE t1 SET c1 = 1;
SELECT DISTINCT * FROM t1 WHERE (t1.c0 IS NULL);
```

The INSERT and UPDATE statements corrupted the database
## Oracles

We found only a low number of crash bugs, likely because DBMS are fuzzed extensively.

<table>
<thead>
<tr>
<th>DBMS</th>
<th>Containment</th>
<th>Error</th>
<th>SEGFAULT</th>
</tr>
</thead>
<tbody>
<tr>
<td>SQLite</td>
<td>46</td>
<td>17</td>
<td>2</td>
</tr>
<tr>
<td>MySQL</td>
<td>14</td>
<td>10</td>
<td>1</td>
</tr>
<tr>
<td>PostgreSQL</td>
<td>1</td>
<td>7</td>
<td>1</td>
</tr>
<tr>
<td><strong>Sum</strong></td>
<td><strong>61</strong></td>
<td><strong>34</strong></td>
<td><strong>4</strong></td>
</tr>
</tbody>
</table>
Average # statements

Half of all bugs can be reproduced with only 4 SQL statements
## Bug importance

<table>
<thead>
<tr>
<th>Severity Level</th>
<th>#</th>
</tr>
</thead>
<tbody>
<tr>
<td>Critical</td>
<td>14</td>
</tr>
<tr>
<td>Severe</td>
<td>8</td>
</tr>
<tr>
<td>Important</td>
<td>14</td>
</tr>
</tbody>
</table>
Validate
Object
Detectors
DL-based object detectors

Surveillance Camera

Medical Image Processing

Auto-Driving Systems
“autonomous driving system failed to recognize a white truck against a bright sky”

believed to be due to its failure to recognize the pedestrian in dark clothing
Typical image analysis tasks

- **Image Classification**: Identifying and classifying objects.
- **Object Detection** (localization + classification): Localizing and classifying objects.
- **Instance Segmentation**: Detecting and segmenting individual instances of objects.
- **Semantics Segmentation**: Segmenting objects based on semantic information.

**Examples**:
- **Entire Image**: Object detection in automated driving systems.
- **Multiple Objects**: Detecting and segmenting multiple objects.
- **No objects; just pixels**: No objects detected.

**Images**:
- Elephant
- Elephant; elephant
- Elephant; elephant
- Grass; elephant; tree
Typical image analysis tasks

- **Image Classification**: Entire Image
- **Object Detection** (localization + classification): Multiple Objects
- **Instance Segmentation**: No objects; just pixels
- **Semantics Segmentation**: Objects in the context of the image

*Focused on by existing DNN testing work*
Typical image analysis tasks

- **Image Classification**: Entire Image
- **Object Detection (localization + classification)**: Multiple Objects
- **Instance Segmentation**: No objects; just pixels
- **Semantics Segmentation**: Focused on by existing DNN testing work

6.5 USD for this figure
Basic idea
Basic idea
Examples

Background image

Missing vehicles and bicycle riders

Background image

Missing a frisbee

Background image

Missing a moon
Workflow

Image Dataset

Instance Segmentation

Object Pool

Object Refinement & Selection

Object Insertion

pick one background image

Objects of “bird”

Bird_m

Objects of “Person”

Person_n

what to insert

where to insert

Paste

...
Evaluation subjects

<table>
<thead>
<tr>
<th>Object Detector Name</th>
<th>speed</th>
<th>COCO mAP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Amazon Rekognition API [1]</td>
<td>fast</td>
<td>N/A</td>
</tr>
<tr>
<td>Google AutoML Vision API [2]</td>
<td>fast</td>
<td>N/A</td>
</tr>
<tr>
<td>Microsoft Azure Vision API [4]</td>
<td>fast</td>
<td>N/A</td>
</tr>
<tr>
<td>IBM Vision API [6]</td>
<td>fast</td>
<td>N/A</td>
</tr>
<tr>
<td>Tensorflow SSD Mobilenet [33]</td>
<td>fast</td>
<td>21</td>
</tr>
<tr>
<td>Tensorflow SSD Inception [42]</td>
<td>fast</td>
<td>24</td>
</tr>
<tr>
<td>Tensorflow faster RCNN Resnet [25]</td>
<td>medium</td>
<td>32</td>
</tr>
<tr>
<td>Tensorflow faster RCNN Inception Resnet [66]</td>
<td>slow</td>
<td>37</td>
</tr>
</tbody>
</table>

- Extract objects from 1000 images from the COCO dataset
- Take 500 images as the “background” to generate test cases
## Result summary

<table>
<thead>
<tr>
<th>Object Detector</th>
<th>#Synthetic Images</th>
<th>#Detected Objects</th>
<th>#Images Causing Detection Failures</th>
<th>Processing Time (Hours)</th>
<th>Total Cost (USD)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Amazon Rekognition API</td>
<td>38,939</td>
<td>3,750</td>
<td>6,060 (15.6%)</td>
<td>11.9</td>
<td>$21.5</td>
</tr>
<tr>
<td>Google AutoML Vision API</td>
<td>18,655</td>
<td>1,801</td>
<td>2,738 (14.7%)</td>
<td>13.0</td>
<td>$18.8</td>
</tr>
<tr>
<td>Microsoft Azure Vision API</td>
<td>20,453</td>
<td>1,985</td>
<td>2,494 (12.2%)</td>
<td>3.2</td>
<td>free</td>
</tr>
<tr>
<td>IBM Vision API</td>
<td>13,280</td>
<td>1,290</td>
<td>1,515 (11.4%)</td>
<td>2.2</td>
<td>free</td>
</tr>
<tr>
<td>SSD Mobilenet</td>
<td>24,796</td>
<td>2,387</td>
<td>3,460 (14.0%)</td>
<td>62.5</td>
<td>$53.4</td>
</tr>
<tr>
<td>SSD Inception</td>
<td>29,072</td>
<td>2,806</td>
<td>3,988 (13.7%)</td>
<td>64.3</td>
<td>$54.9</td>
</tr>
<tr>
<td>RCNN Resnet</td>
<td>70,754</td>
<td>6,914</td>
<td>7,442 (10.5%)</td>
<td>164.8</td>
<td>$140.7</td>
</tr>
<tr>
<td>RCNN Inception</td>
<td>76,257</td>
<td>7,349</td>
<td>10,648 (14.0%)</td>
<td>290.8</td>
<td>$248.3</td>
</tr>
<tr>
<td>Total</td>
<td>292,206</td>
<td>28,282</td>
<td>38,345 (13.1%)</td>
<td>612.7</td>
<td>$537.6</td>
</tr>
</tbody>
</table>
Other domains

- SMT solvers (Z3 & CVC4)
- SMC tools (CPAchecker, CBMC & Seahorn)
- Android mobile apps (going beyond crashes)
- Machine translation systems
- Smart contracts
- ...

THE SOFTWARE SPECIFICATION CRISIS
Finding a/the right balance between human & machine collaboration
Finding a/the right balance between human & machine collaboration

Thank you!