EMI Testing: Finding 1000+ Bugs in GCC and LLVM in 3 Years

Zhendong Su

University of California, Davis

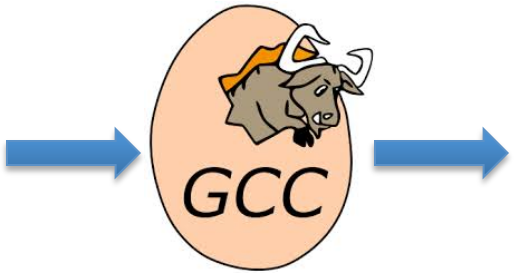



Mehrdad Afshari **Vu Le** **Chengnian Sun** **Qirun Zhang**

http://web.cs.ucdavis.edu/~su/emi-project/

compilers

```
int main ()
{
  // do something
  …
  return 0;
}
```




```
…
call    puts
movl    $0, %eax
popq    %rbp
ret
```

Developers' **belief**:

Compilers are ***faithful*** translators

let’s reflect on this trust

How trustworthy are compilers?

How they impact security & reliability?

In 2014, GCC 4.9 miscompiled the Linux kernel!

*"Ok, so I'm looking at the code generation and your compiler is pure and utter ****.*

……

Adding Jakub to the cc, because gcc-4.9.0 seems to be terminally broken."

-- Linus Torvalds

Type #1 bugs

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

Type #2 “bugs”

gcc “bug” 8537

```
#include <string>
using std::string;
#include <memory>

// The specifics of this function are not important for
// demonstrating this bug.
const string getPasswordFromUser() const;

bool isPasswordCorrect() {
   bool isPasswordCorrect = false;
   string Password("password");

   if(Password == getPasswordFromUser()) {
      isPasswordCorrect = true;
   }

   // Removed from the optimized code although it secures
   // the code by wiping the password from memory
   memset(Password, 0, sizeof(Password));
   return isPasswordCorrect;
}
```

debatable …

From: "Joseph D. Wagner" <wagnerjd@prodigy.net>
To: <fw@gcc.gnu.org>, gcc-bugs@gcc.gnu.org, <gcc-prs@gcc.gnu.org>,
nobody@gcc.gnu.org, wagnerjd@prodigy.net,<gcc-gnats@gcc.gnu.org>
Cc:
Subject: RE: optimization/8537: Optimizer Removes Code Necessary for Security
Date: Sun, 17 Nov 2002 08:59:53 -0600

Direct quote from:
http://gcc.gnu.org/onlinedocs/gcc-3.2/gcc/Bug-Criteria.html

"If the compiler produces valid assembly code that does not correctly execute the input source code, that is a compiler bug."

So to all you naysayers out there who claim this is a programming error or poor coding, YES, IT IS A BUG!

Type #3 “bugs”

how about this one?

```
char *buf = ...;
char *buf_end = ...;
unsigned int len = ...;

if (buf + len >= buf_end)
    return; // len too large

if (buf + len < buf)
    return; //overflow, buf+len wrapped around

// write to buf[0..len-1]
...
```

even more debatable …

- Pointer overflow is **undefined behavior**
  - Compiler assumes code has no undefined behavior
  - Thus compiler assumes: **buf + len** cannot overflow
  - Then compiler infers: **if (buf + len < buf) ⇒ if (0)**
- But this is a **security threat**
  - Use a large **len** to trigger **buffer overflow**

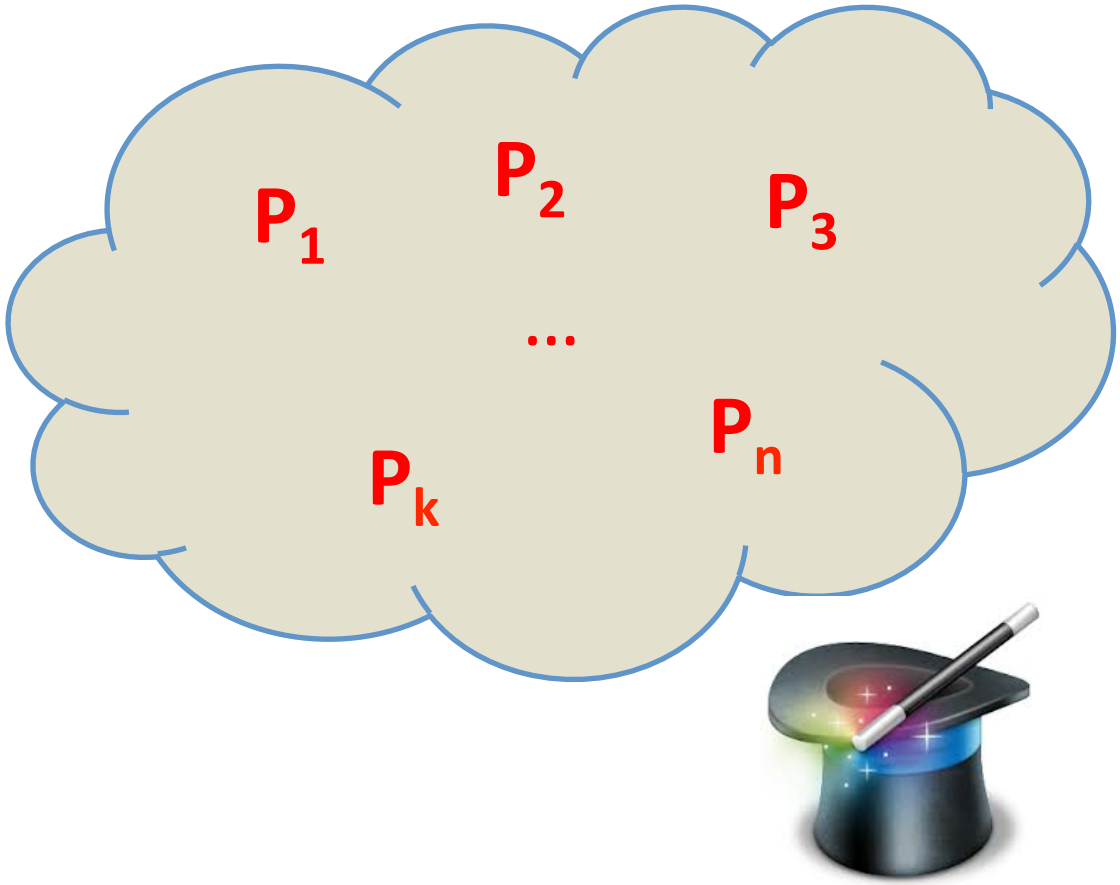
Type #1 bugs: EMI Testing

How to find 1000+ GCC/LLVM bugs in 3 years?

real

vision

$P \equiv$



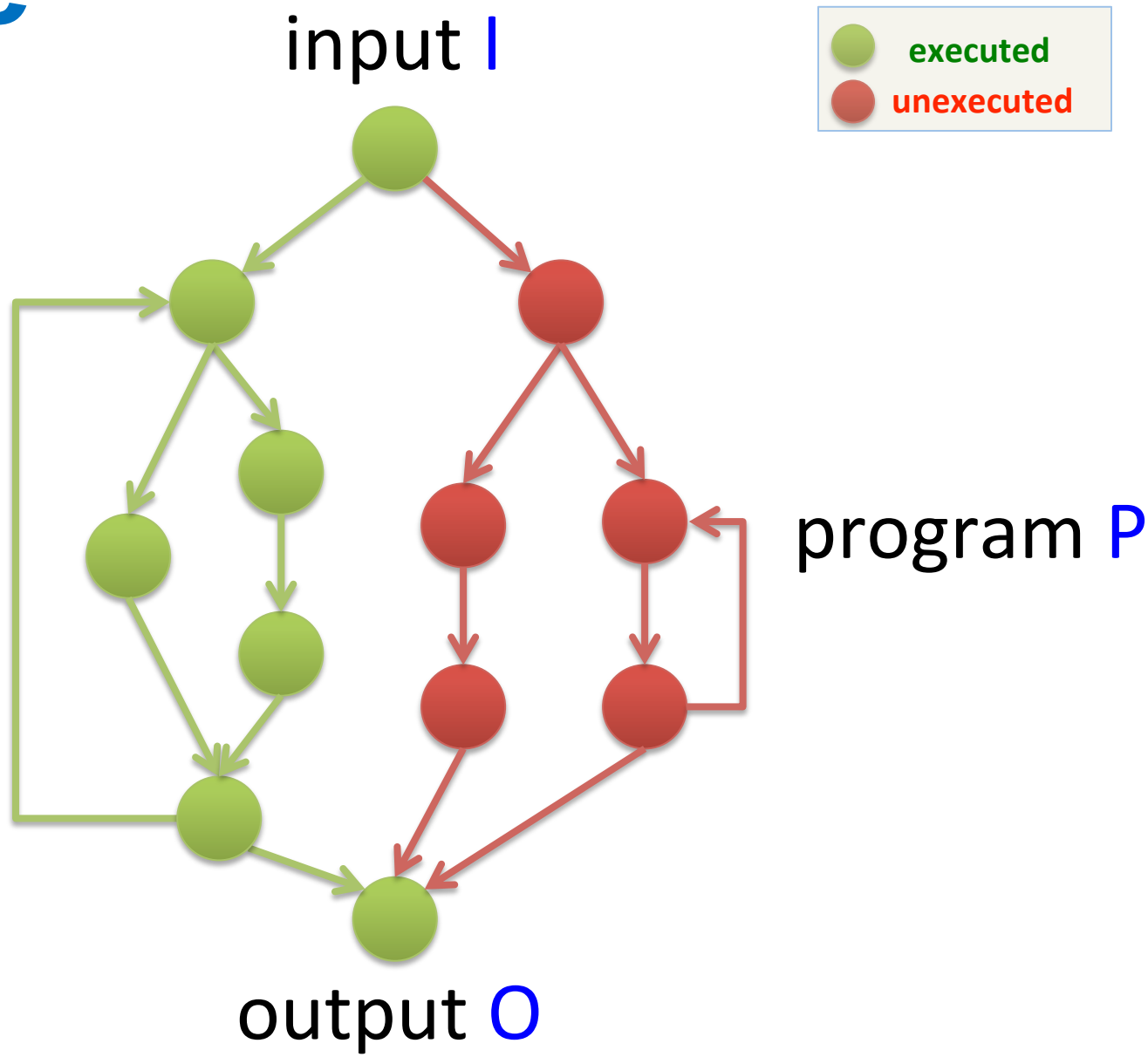
key challenges

- Generation
  - How to generate **different**, yet **equivalent** tests?
- Validation
  - How to check that tests are **indeed equivalent**?
- Both are long-standing hard issues

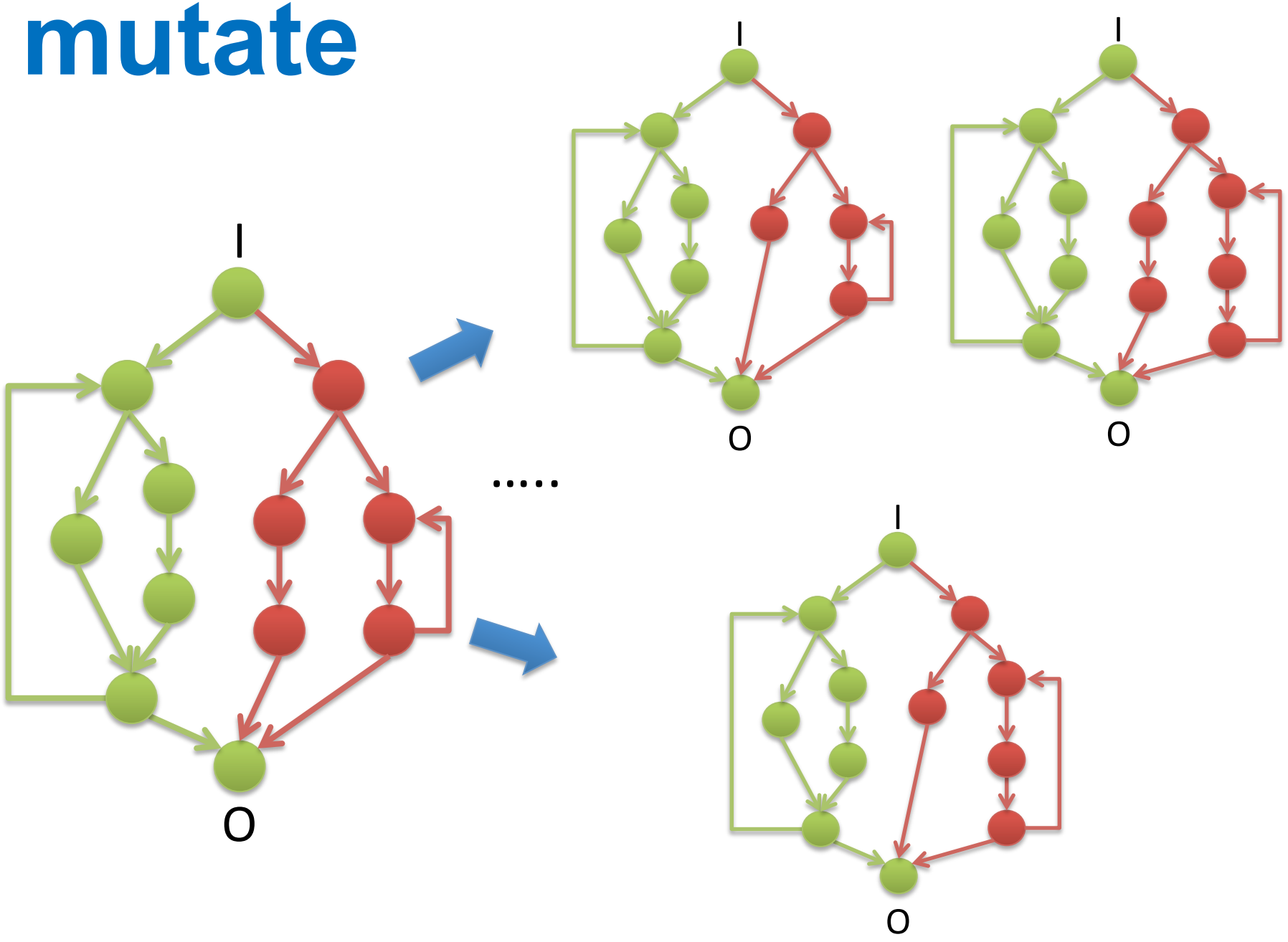
equiv. modulo inputs

- 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**

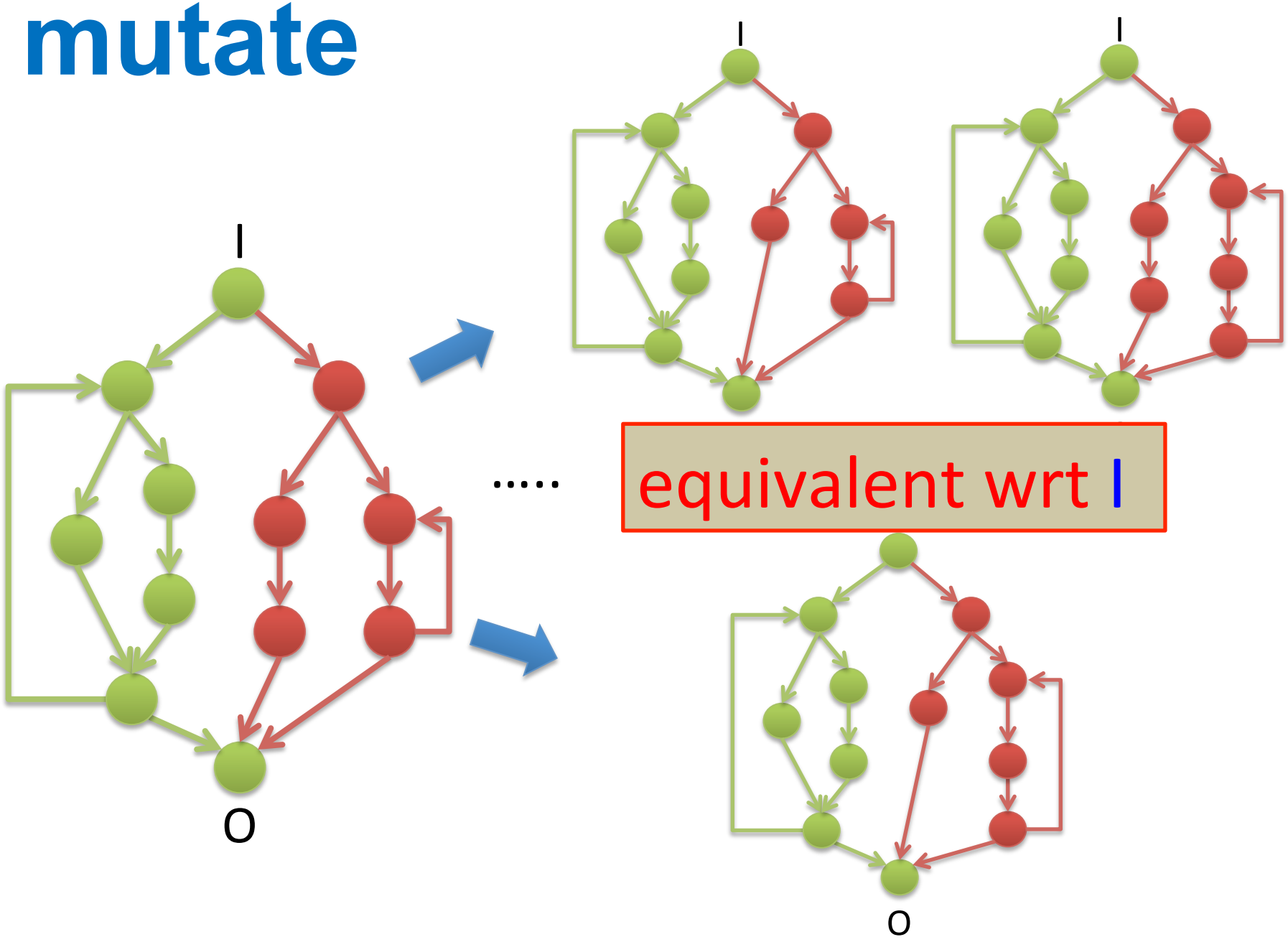
profile



mutate



mutate



revisit challenges

- Generation (**easy**)
  - How to generate **different**, yet **equivalent** tests?
- Validation (**easy**)
  - How to check that tests are **indeed equivalent**?
- ~~Both are long-standing hard issues~~

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)
```

seed file

```
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

```
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

```
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);
}
```

```
$ 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

```
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;
}
```

GVN: load struct using 32-bit load

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

llvm bug autopsy

```
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;
}
```

GVN: load struct using 32-bit load

SRoA: read past the struct's end

➔

undefined behavior

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

llvm bug autopsy

```
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;
}
```

GVN: load struct using 32-bit load

SRoA: read past the struct's end → undefined behavior

remove

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

seed file

```
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

```
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);
}
```

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

gcc bug 58731

```
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

```
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;
}
```

PRE: loop invariant

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

gcc bug autopsy

```
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;
}
```

LIM

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

gcc bug autopsy

```
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;
}
```

integer overflow

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

seed program

```
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;
}
```

no longer a loop invariant

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

orion

- First and simplest EMI realization
- Targeting C compilers
  - Randomly **prune** unexecuted code
  - Very **effective**

bug counts (till 03/2014)

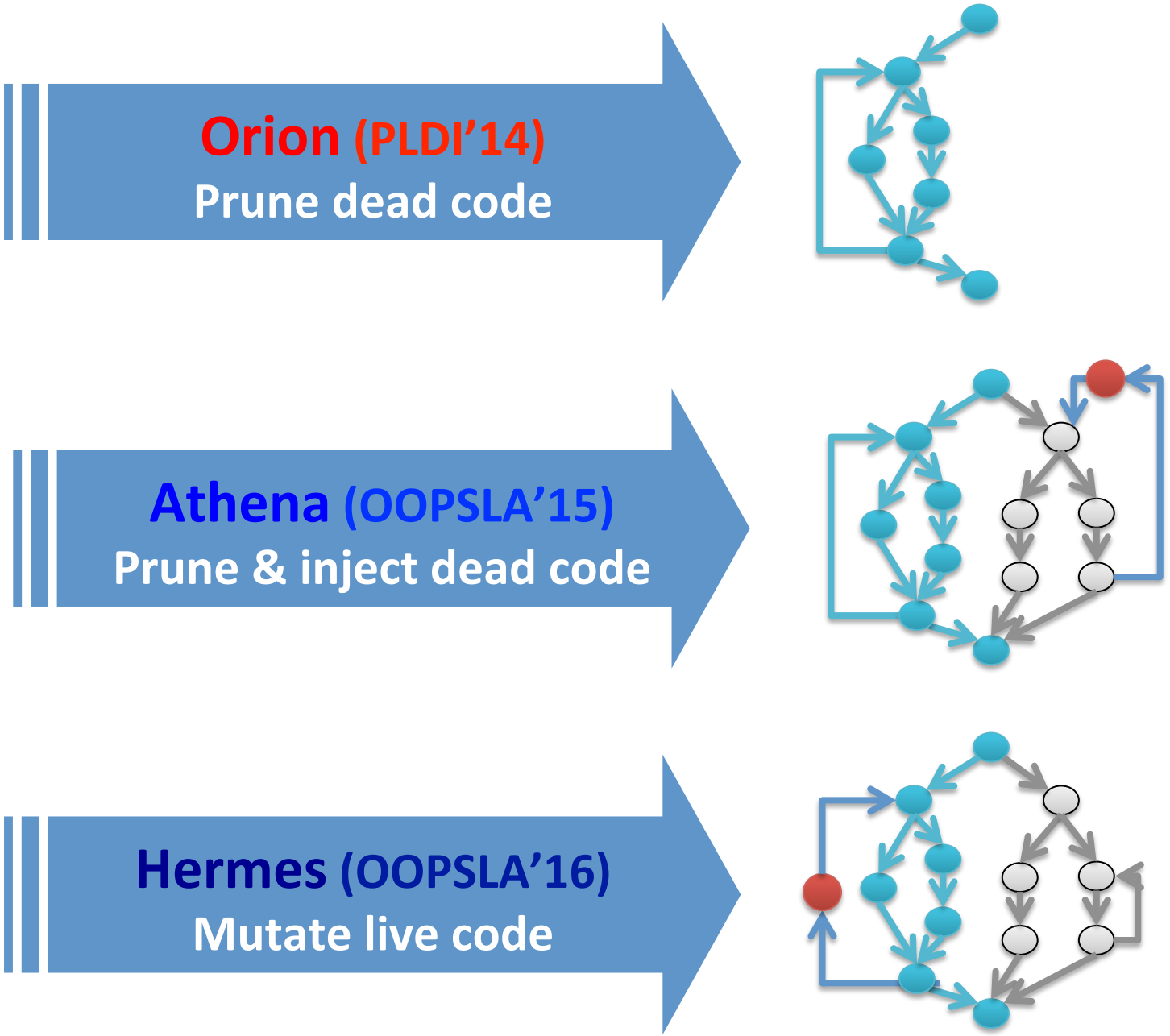
	GCC	LLVM	TOTAL
Reported	111	84	**195**
Marked Duplicate	28	7	**35**
Confirmed	79	68	**147**
Fixed	56	54	**110**

bug types

	GCC	LLVM	TOTAL
Wrong code	46	49	**95**
Crash	23	10	**33**
Performance	10	9	**19**

bug importance

- Most bugs have already been **fixed**
- Many were **critical**, **release-blocking**
- Some affected **real-world** projects


Orion (PLDI'14)
Prune dead code
Athena (OOPSLA'15)
Prune & inject dead code
Hermes (OOPSLA'16)
Mutate live code

Athena: gcc bug 61383

Seed Program P

```
int a, c, d, e = 1, f;
int fn1 () {
 int h;
 for (; d < 1; d = e) {
   h = (f == 0) ? 0 : 1 % f;
   if (f < 1) c = 0;
   else c = 1;
 }
}
int main () {
 fn1 ();
 return 0;
}
```

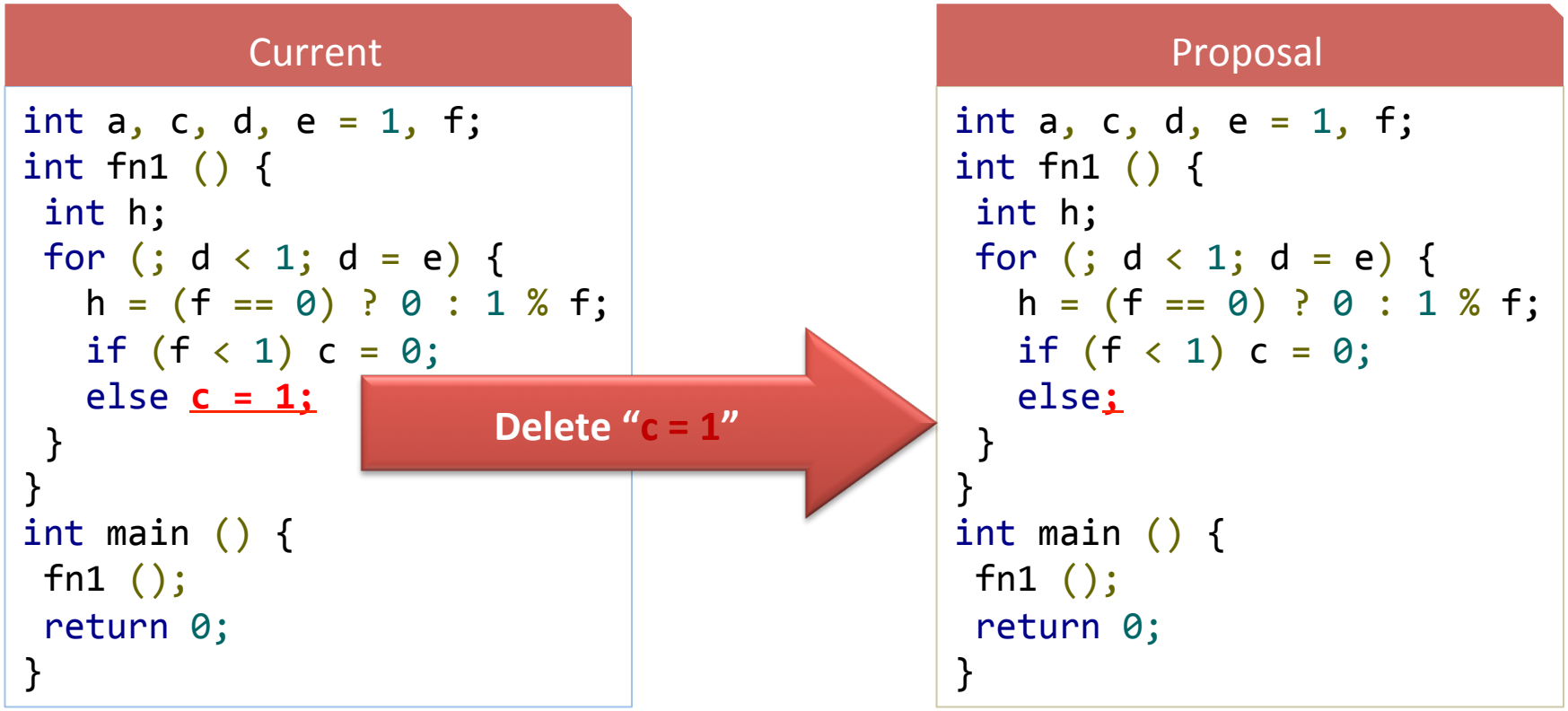
Bug-triggering Variant

```
int a, c, d, e = 1, f;
int fn1 () {
 int h;
 for (; d < 1; d = e) {
   h = (f == 0) ? 0 : 1 % f;
   if (f < 1) c = 0;
   else if (h) break;
 }
}
int main () {
 fn1 ();
 return 0;
}
```

```
$ gcc -O0 test.c ; ./a.out
$ gcc -O2 test.c ; ./a.out
Floating point exception (core dumped)
```

Current

```
int a, c, d, e = 1, f;
int fn1 () {
 int h;
 for (; d < 1; d = e) {
   h = (f == 0) ? 0 : 1 % f;
   if (f < 1) c = 0;
   else c = 1;
 }
}
int main () {
 fn1 ();
 return 0;
}
```


Current
int a, c, d, e = 1, f;
int fn1 () {
int h;
for (; d < 1; d = e) {
h = (f == 0) ? 0 : 1 % f;
if (f < 1) c = 0;
else c = 1;
}
}
int main () {
fn1 ();
return 0;
}
Delete “c = 1”
Proposal
int a, c, d, e = 1, f;
int fn1 () {
int h;
for (; d < 1; d = e) {
h = (f == 0) ? 0 : 1 % f;
if (f < 1) c = 0;
else;
}
}
int main () {
fn1 ();
return 0;
}

Current

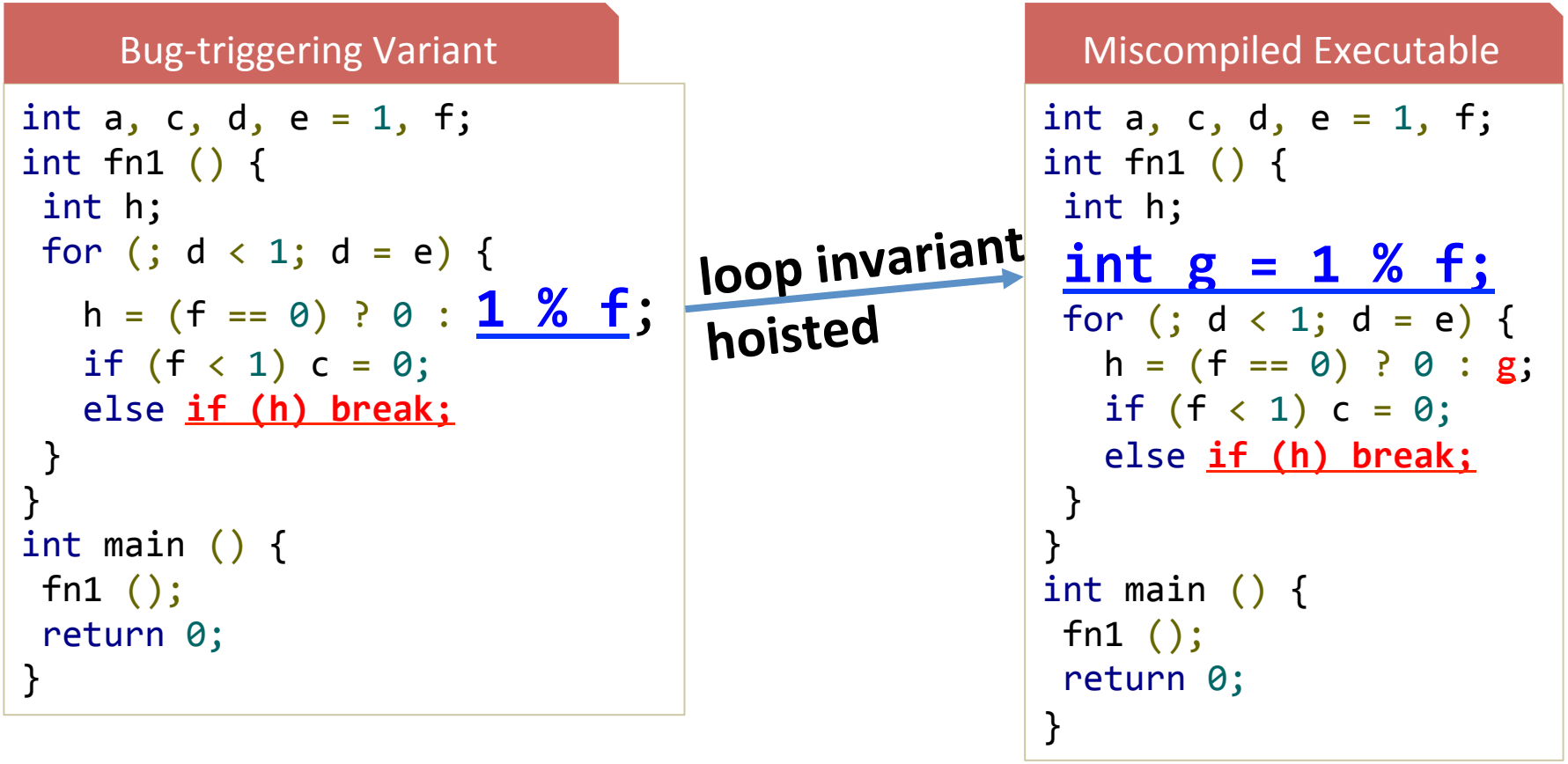
```
int a, c, d, e = 1, f;
int fn1 () {
 int h;
 for (; d < 1; d = e) {
   h = (f == 0) ? 0 : 1 % f;
   if (f < 1) c = 0;
   else;
 }
}
int main () {
 fn1 ();
 return 0;
}
```

Insert the statement below

Proposal

```
int a, c, d, e = 1, f;
int fn1 () {
 int h;
 for (; d < 1; d = e) {
   h = (f == 0) ? 0 : 1 % f;
   if (f < 1) c = 0;
   else if (h) break;
 }
}
int main () {
 fn1 ();
 return 0;
}
```

Context	Statement
1. Requires loop	`if (i)` `  break;`
2. `int i;`	


Bug-triggering Variant
int a, c, d, e = 1, f;
int fn1 () {
int h;
for (; d < 1; d = e) {
h = (f == 0) ? 0 : 1 % f;
if (f < 1) c = 0;
else if (h) break;
}
}
int main () {
fn1 ();
return 0;
}
loop invariant hoisted
Miscompiled Executable
int a, c, d, e = 1, f;
int fn1 () {
int h;
int g = 1 % f;
for (; d < 1; d = e) {
h = (f == 0) ? 0 : g;
if (f < 1) c = 0;
else if (h) break;
}
}
int main () {
fn1 ();
return 0;
}

Hermes

- **Profile** and **record variable values**
- Synthesize code snippets
  - Ensure **no undefined behavior**
  - Ensure **EMI property locally**
    - Maintain same program state at entry & exit

Hermes: llvm 26266

Seed Program **P**

```
char a;
int b, c = 9, d, e;

void fn1() {
 unsigned f = 1;
 int g = 8, h = 5;
 for (; a != 6; a--) {
   int *i = &h, *j;
   for (;;) {
// b=0,c=9,e=0,f=1,g=8,h=5
     if (d <= 8) break;
     *i = 0;
     for (; *j <= 0;);
   }
 }
}
int main() {fn1(); return 0;}
```

EMI Variant

```
   int *i = &h, *j;
   for (;;) {
// b=0,c=9,e=0,f=1,g=8,h=5
int backup_g = e, backup_f = ~1;
if (g && h) {
  backup_g = g;
  backup_f = f;
  f = -(~(c && b)|-~(e*~backup_f));
  if (c < f) abort();
}
g = backup_g;
f = backup_f;
     if (d <= 8) break;
     *i = 0;
     for (; *j <= 0;);
   }
```

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

EMI Variant

```
    int *i = &h, *j;
  for (;;) {
// b=0,c=9,e=0,f=1,g=8,h=5
int backup_g = e, backup_f = ~1;
if (g && h) {
  backup_g = g;
  backup_f = f;
  f = -(~(c && b)|-~(e*~backup_f));
  if (c < f) abort();
}
g = backup_g;
f = backup_f;
    if (d <= 8) break;
    *i = 0;
    for (; *j <= 0;);
  }
```

Clang (mistakenly) deems this predicate always **true**

```
if/while (P) { // P=false wrt profiled states
  S  // S any synthesized compilable code
}
```

```
if (P) { // P=true wrt profiled states
  S  // S is original code at this program point
}
```

```
int backup_v = valid-expression
if (true-predicate) {
 backup_v = v
 v = valid-expression
 if/while(false-predicate) { print v }
}
v = backup_v
```

```
~(c && b)|-~(e*~f)
```

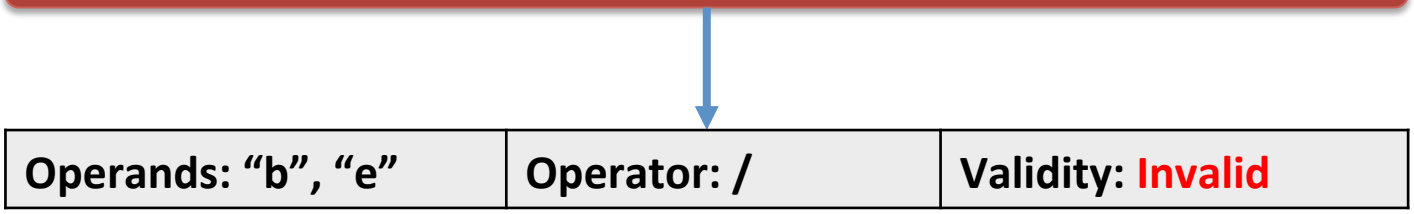
env: b=0,c=9,e=0,f=1,g=8,h=5

worklist = [“b”, “c”, “e”, “f”]

```
~(c && b)|-~(e*~f)
```

env: b=0,c=9,e=0,f=1,g=8,h=5

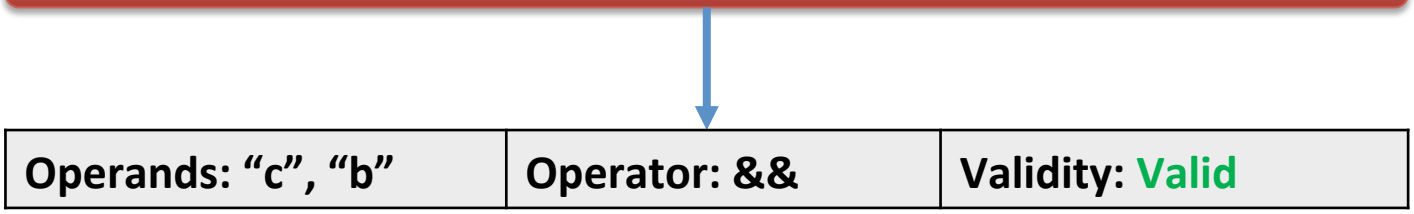
worklist = [“b”, “c”, “e”, “f”]



```
~(c && b)|-~(e*~f)
```

env: b=0,c=9,e=0,f=1,g=8,h=5

worklist = [“b”, “c”, “e”, “f”]

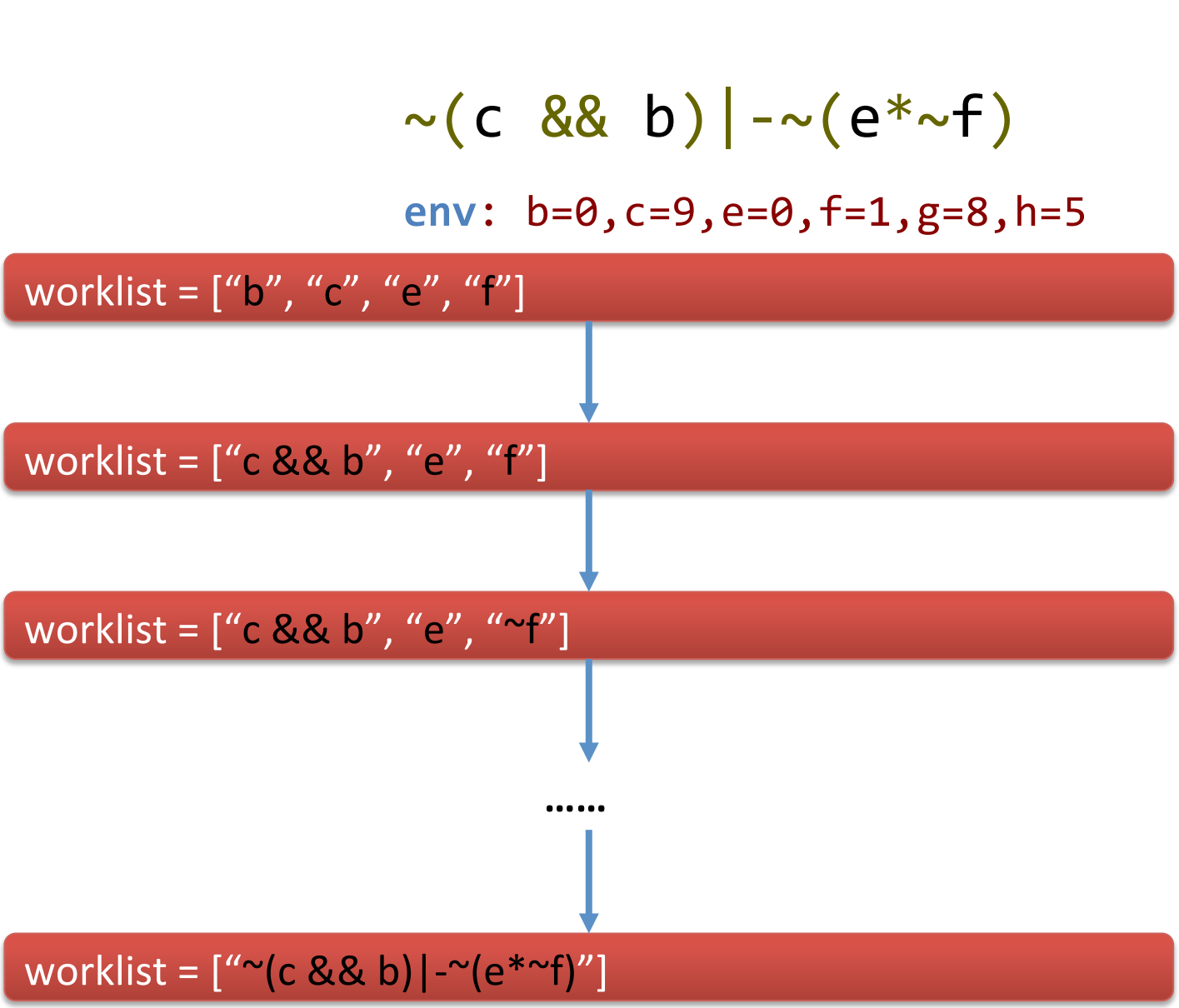


```
~(c && b)|-~(e*~f)
```

env: b=0,c=9,e=0,f=1,g=8,h=5

worklist = [“b”, “c”, “e”, “f”]

worklist = [“c && b”, “e”, “f”]


~(c && b)|-~(e*~f)
env: b=0,c=9,e=0,f=1,g=8,h=5
worklist = [“b”, “c”, “e”, “f”]
worklist = [“c && b”, “e”, “f”]
worklist = [“c && b”, “e”, “~f”]
.....
worklist = [“~(c && b)|-~(e*~f)”]

bug counts (till recently)

	GCC	LLVM	TOTAL
Reported	643	498	**1141**
Fixed	397	239	**636**

- **ISSTA'15: Stress-testing link-time optimization**
- **ICSE'16: Analyzing compilers' diagnostic support**
- **Recent: Skeletal program enumeration**

ongoing & future work

- Extend EMI to handle **floating-point** code
- Adapt EMI to other **languages** & **settings**
- Support **bounded compiler verification**

how about CompCert?

```
// test.c
int main ()
{
   int a[2] = 0;
   return 0;
}
```

```
// test.light.c
int main (void)
{
   int a[2];
   *(a+0) = 0;
   *(a+1) = 0;
   return 0;
}
```

```
$ ccomp test.c; ./a.out
$ ccomp -interp test.c
Time 21: program terminated (exit code = 0)
$
```

how about CompCert?

```
// test.c
#include <stdio.h>
int main(){
  int i = '\214';
  printf("%d\n", i);
  return 0;
}
```

```
$ ccomp-2.4 test.c; ./a.out
$ -116
$ ccomp-2.0 test.c; ./a.out
$ 140
```

how about CompCert?

```
// test.c
volatile long long a;
unsigned b;
int main () {
  a = b;
  return 0;
}
```

```
$ ccomp-2.6 test.c
Fatal error: exception File "ia32/Asmexpand.ml", line 191, …
…
$
```

how about CompCert?

```
// test.c
#include <stdio.h>
int main () {
  int t = printf ("0\n");
  printf ("%d\n", t);
  return 0;
}
```

```
$ ccomp-2.6 -interp -quiet test.c
0
0
$ ccomp-2.7 -interp -quiet test.c
0
2
$
```

how about CompCert?

```
// test.c
int a, b, c, d, e, f, g;
void fn1 () {
  int h, i, j;
  if (g) {
     g = 1;
   L1: if (1) ;
    }
  short k = ~j;
  int l = 1 / (h & e & ~d + ((k & ~h) - ((1 | i) & (a | c)))), m = ~~j / (~h | d + a);
  j = l & m | h;
  if (j) ;
  j = k;
  int n = ~i | f, o = ~b - 1 / -n * ~i, p = f;
  goto L1;
}
int main () {
if (a) fn1 ();
  return 0;
}
```

```
$ ccomp-2.7 test.c
...
Fatal error: exception File "backend/Regalloc.ml", line 741, ...
...
$
```

EMI

- EMI is **general** and **widely applicable**
  - Can test compilers, analysis and transformation tools
  - Generates real-world tests
  - Requires no reference compilers
- EMI realizations are very **effective**
  - Have uncovered **1141 bugs** in GCC and LLVM
  - Many were **long latent** and **miscompilations**

conclusion

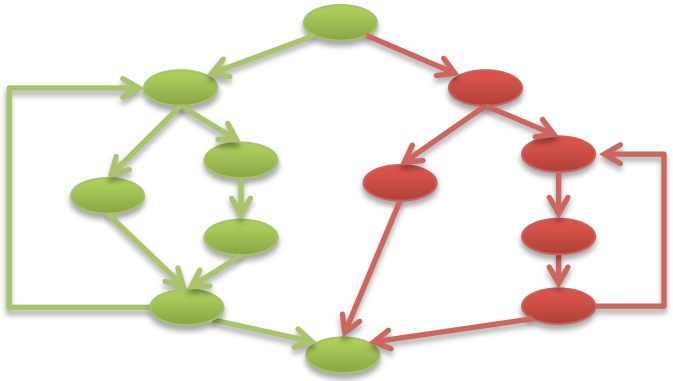
- Compilers are important & complex
- They impact reliable & secure coding
- EMI offers powerful compiler testing
- Much more work is needed

EMI Testing: Finding 1000+ Bugs in GCC and LLVM

```
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;
}
```

```
char *buf = ...;
char *buf_end = ...;
unsigned int len = ...;

if (buf + len >= buf_end)
  return; // len too large

if (buf + len < buf)
  return;//overflow, buf+len wrapped
// write to buf[0..len-1]
```

```
#include <string>
using std::string;
#include <memory>
const string getPasswordFromUser() const;
bool isPasswordCorrect() {
   bool isPasswordCorrect = false;
   string Password("password");
   if(Password == getPasswordFromUser()) {
      isPasswordCorrect = true;
   }
   memset(Password, 0, sizeof(Password));
   return isPasswordCorrect;
}
```



	GCC	LLVM	TOTAL
Reported	643	498	**1141**
Fixed	397	239	**636**

Thank you!