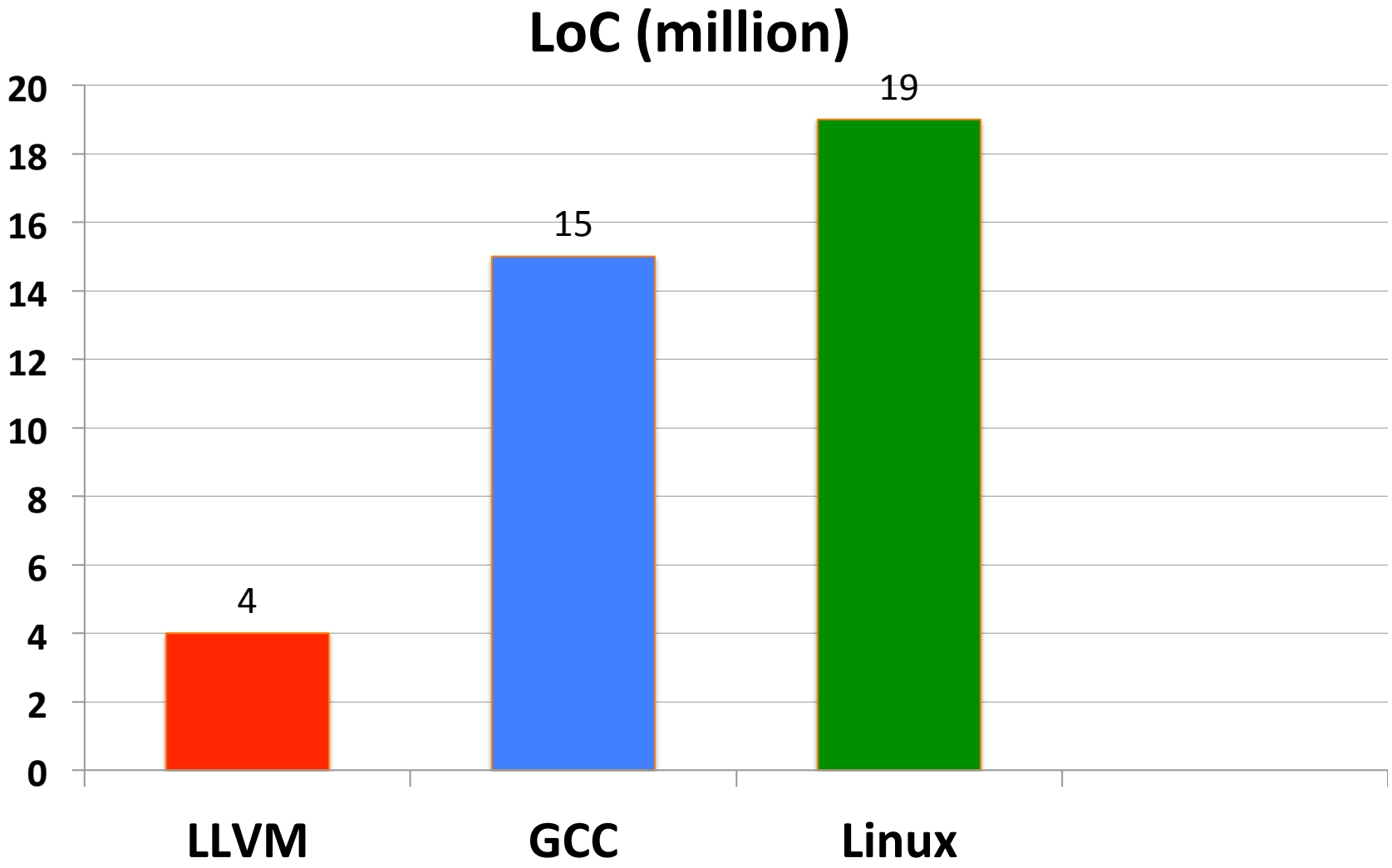
EMI and SPE Testing: Finding 1400+ Bugs in GCC and LLVM

Zhendong Su

University of California, Davis

compiler complexity



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)
```

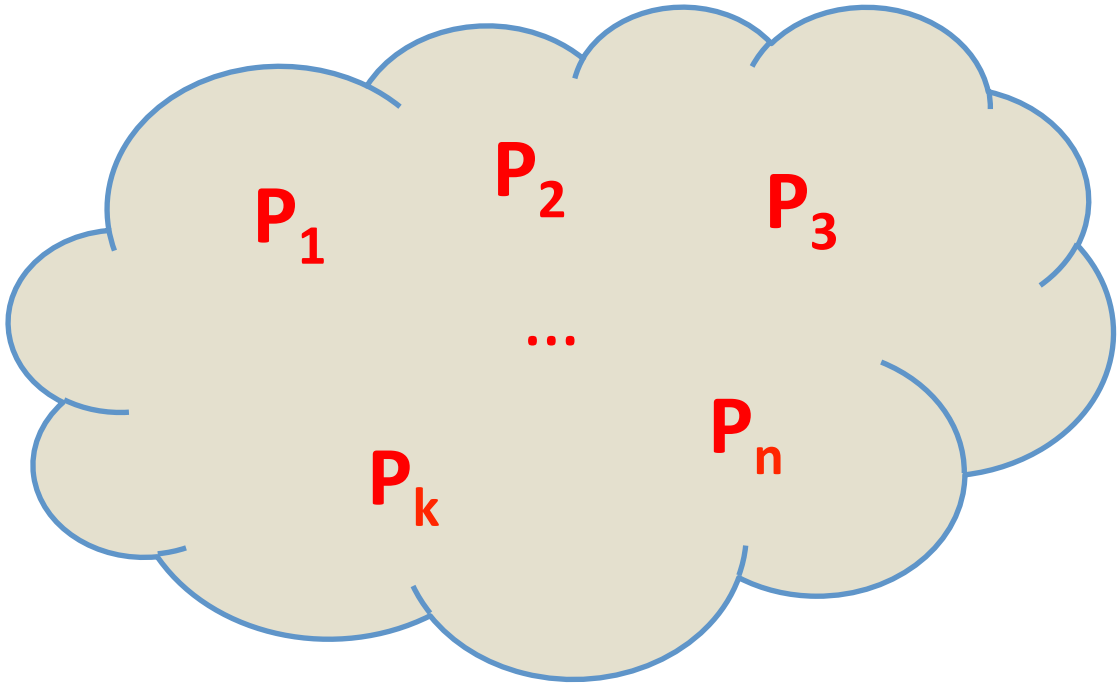
developer comment

"... 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

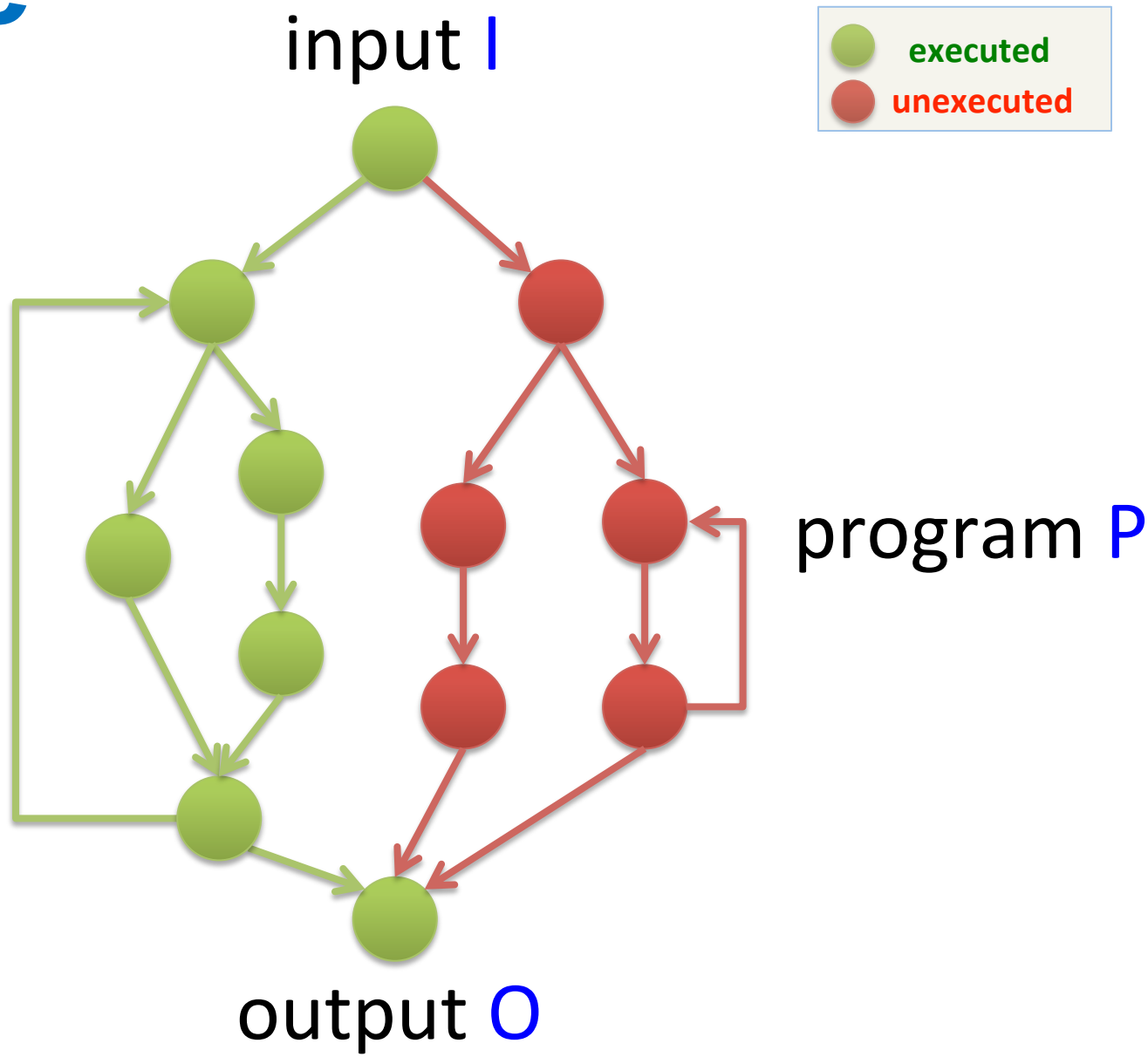
$P \equiv$



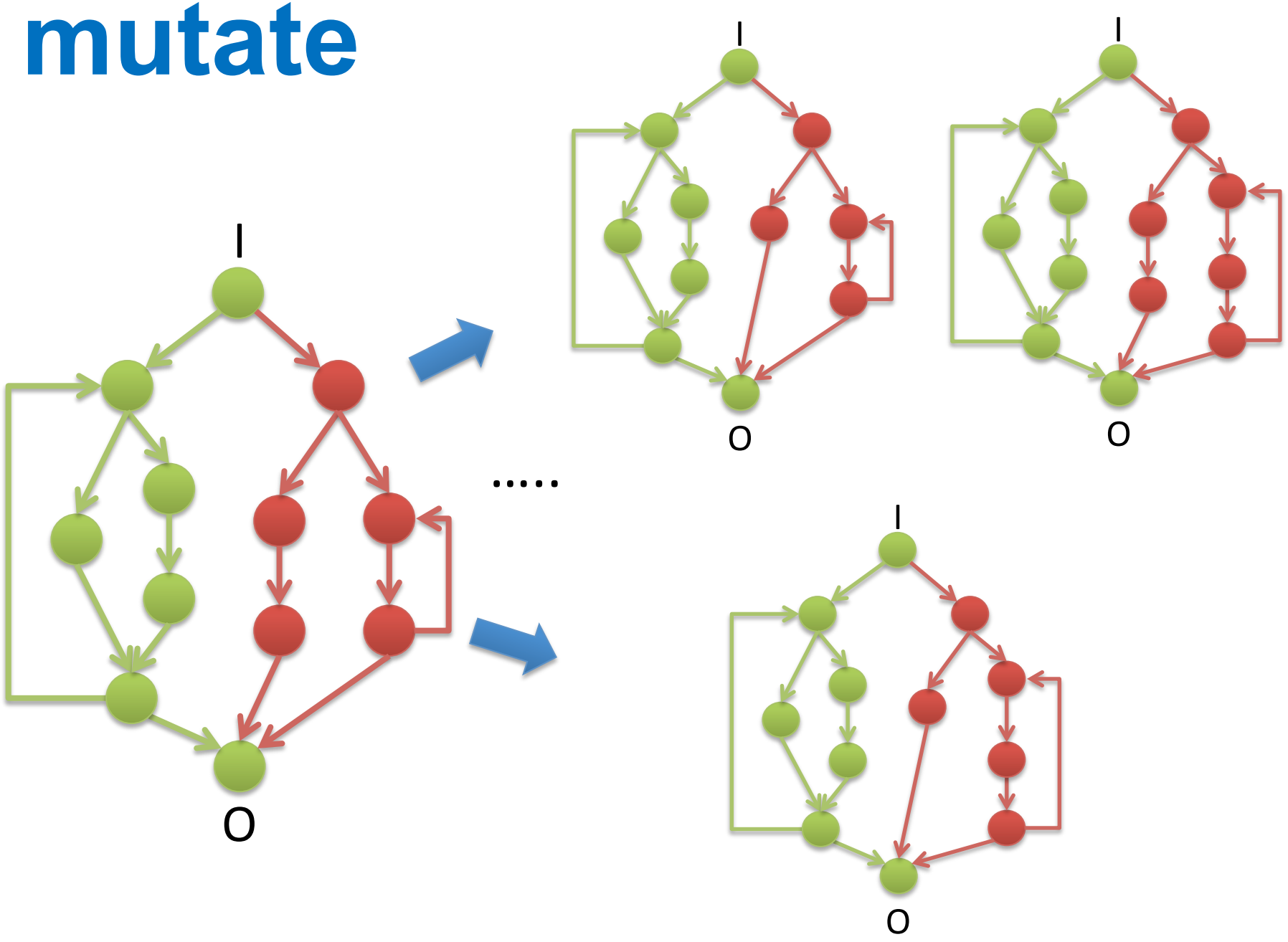
idea: equiv. modulo inputs

- **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**

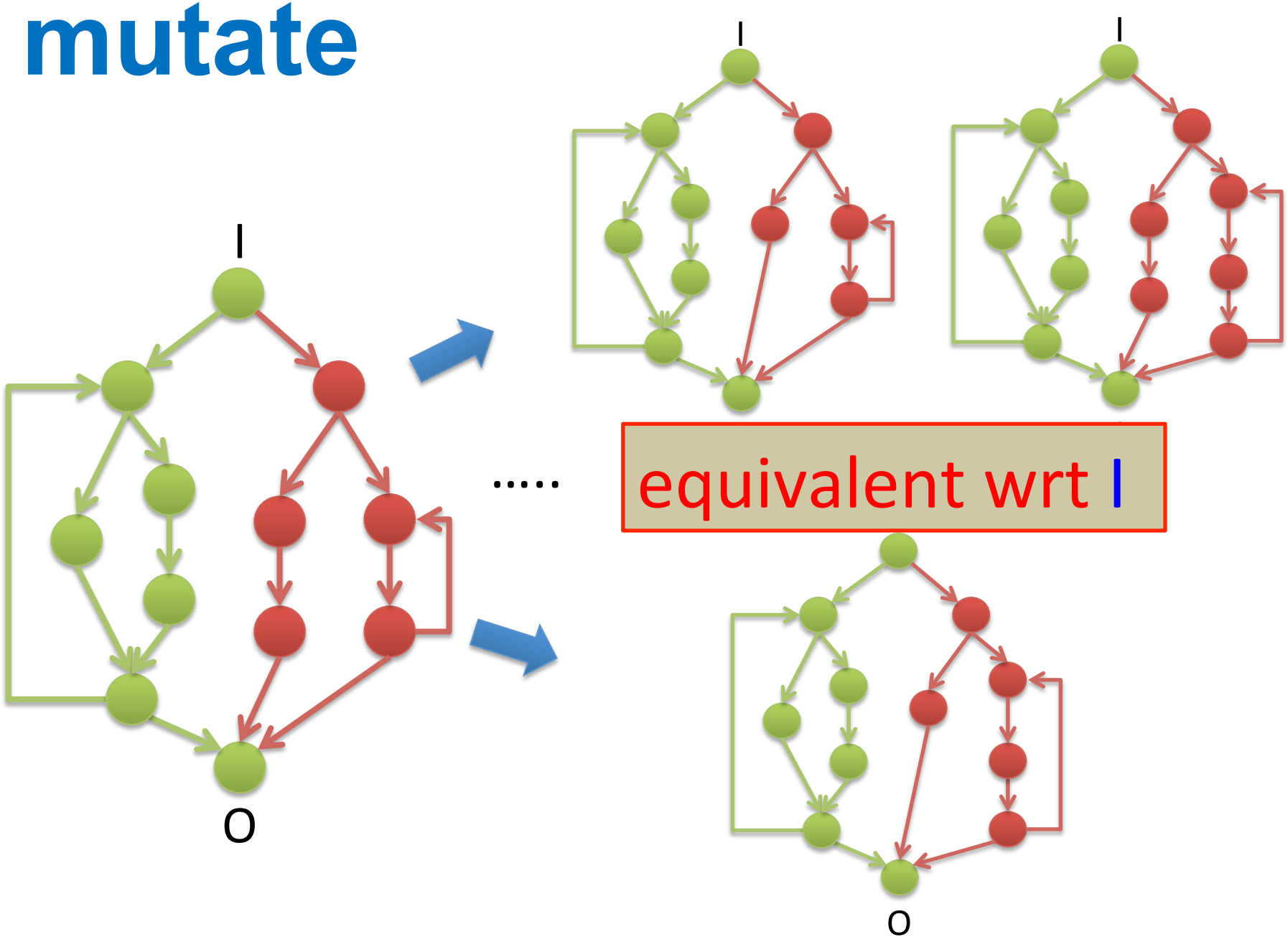
profile



mutate



mutate



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

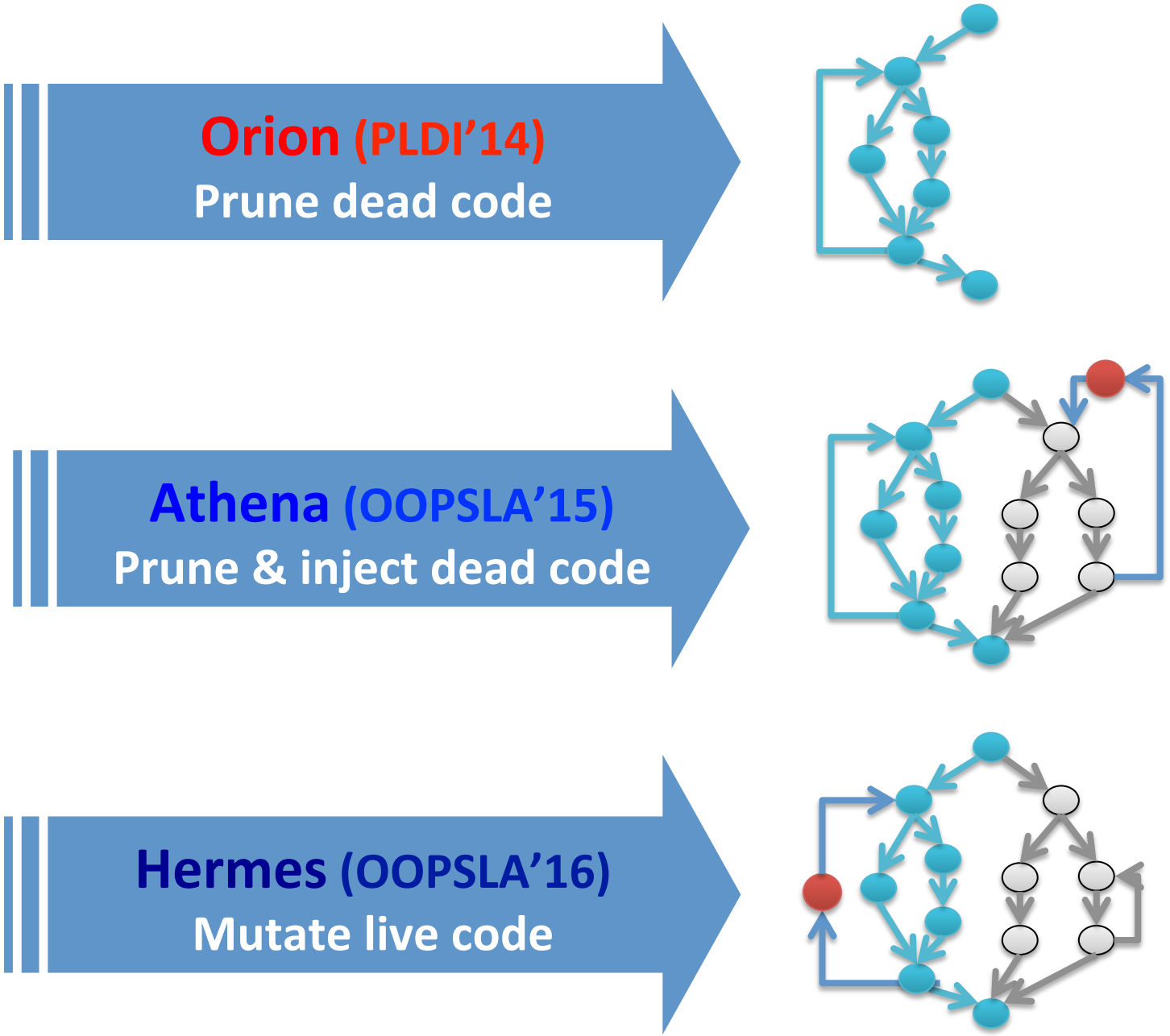
```
struct tiny { char c; char d; char e; };
f(int n, struct tiny x, struct tiny y,
  struct tiny z, long l) {
    if (x.c != 10) abort();
    if (x.d != 20) abort();
    if (x.e != 30) abort();
    if (y.c != 11) abort();
    if (y.d != 21) abort();
    if (y.e != 31) abort();
    if (z.c != 12) abort();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


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

bug counts

	GCC	LLVM	TOTAL
Reported	744	674	**1418**
Fixed	495	358	**853**

- **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 …"

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;
}
```

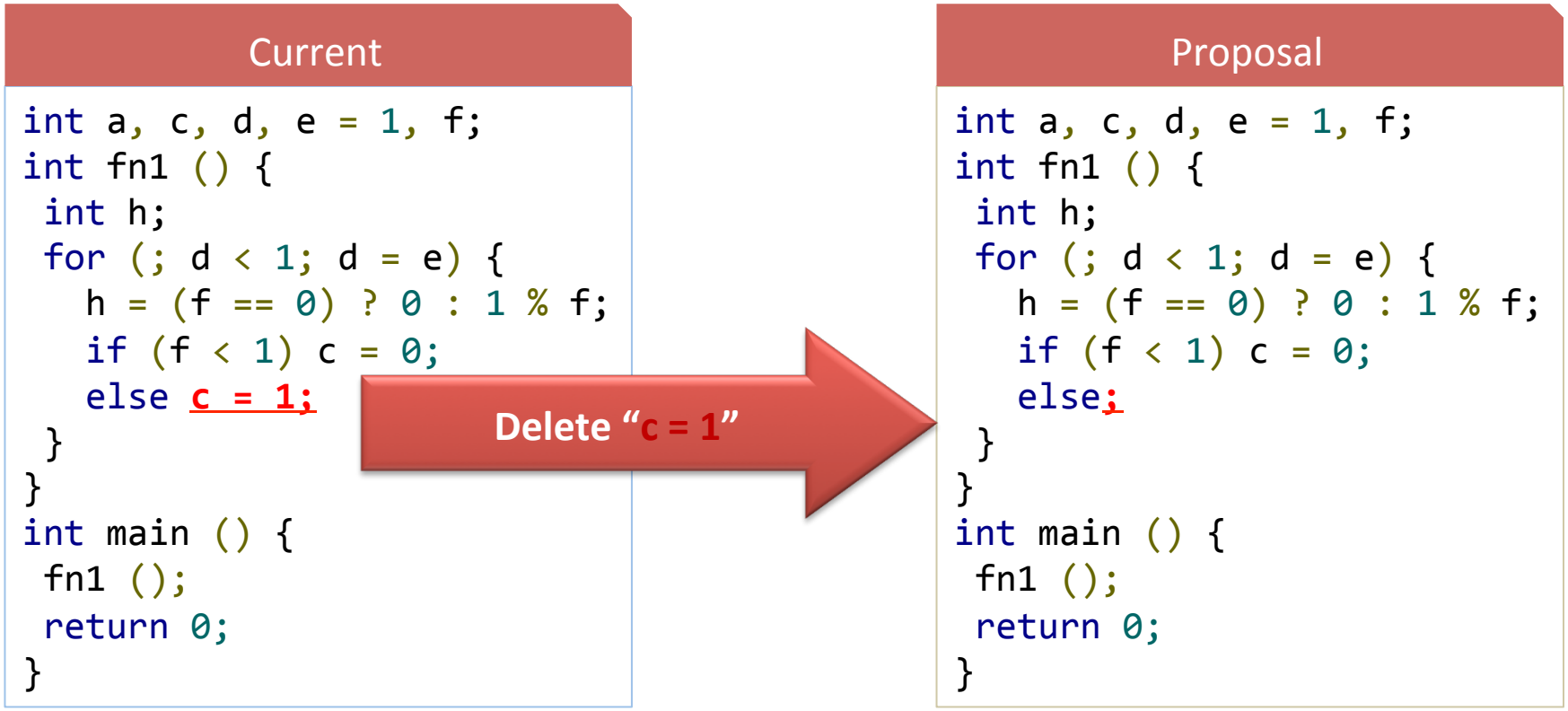
Bug-triggering Variant

```
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

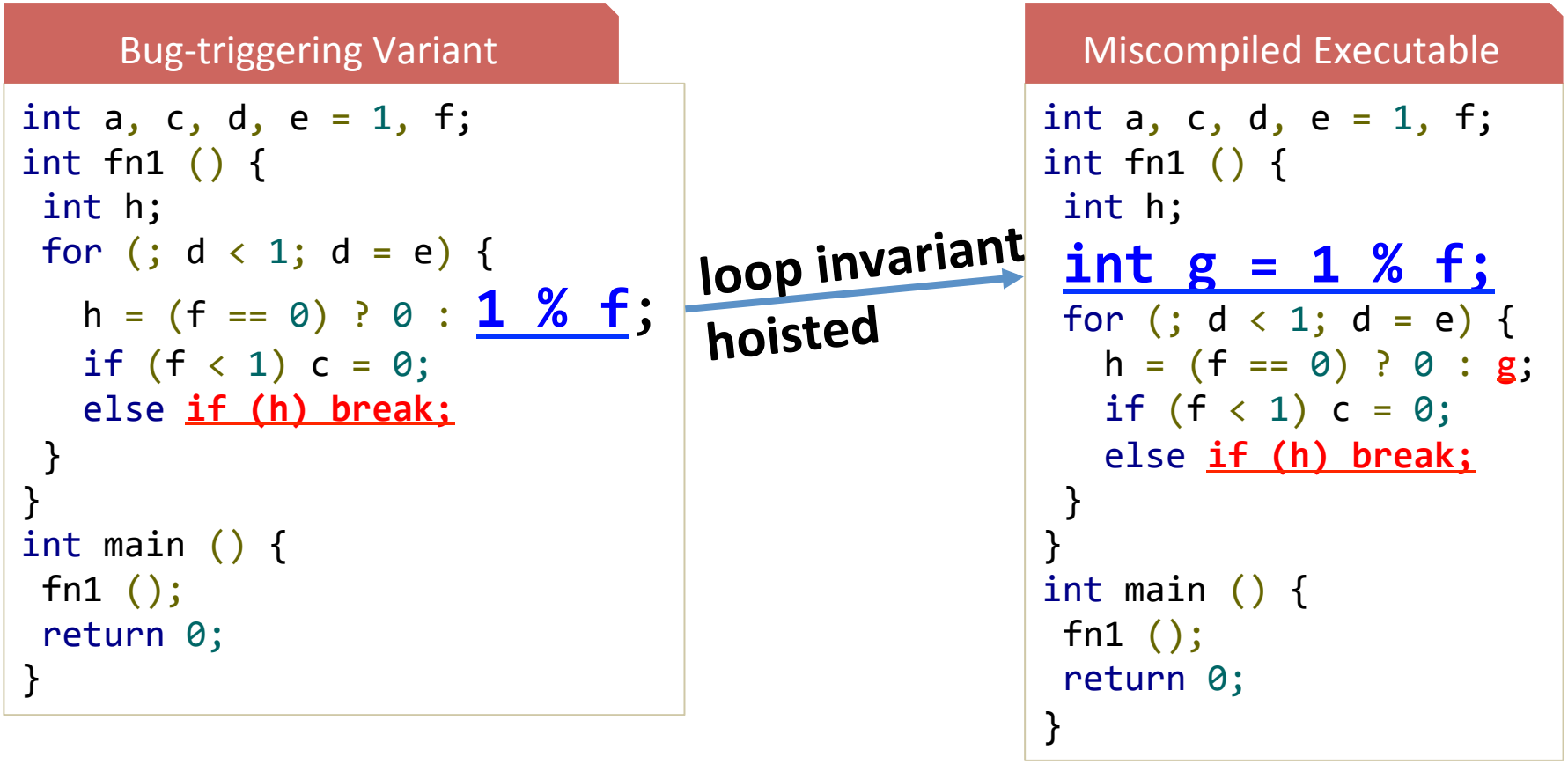
```
int a, c, d, e = 1, f;
int fn1 () {
 int h;
 for (; d < 1; d = e) {
   h = (f == 0) ? 0 : 1 % f;
   if (f < 1) c = 0;
   else;
 }
}
int main () {
 fn1 ();
 return 0;
}
```

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 (OOPSLA'16)

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

```
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
```

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**

Q: How about verification?

A: Skeletal Program Enumeration (SPE)

program enumeration

- **Vision**: Bounded compiler verification
- **Goal**: Program enumeration
  - **Exhaustive** (all small test programs)
  - **Practical**
- **Idea**:
  - Context-free grammar to token sequences
  - Token sequences to programs (**SPE**)

SPE (PLDI'17)

```
a := 10;
b := 1;
while(a) do
  a := a - b;
```

(a) Program P

```
□ := 10;
□ := 1;
while(□) do
  □ := □ - □;
```

(b) Skeleton $\mathbb{P}$

```
b := 10;
a := 1;
while(b) do
  b := b - a;
```

(c) Program P_1

```
a := 10;
b := 1;
while(b) do
  b := a - b;
```

(d) Program P_2

example 1: wrong code

```
1  int a = 0;
2  extern int b __attribute__ ((alias (''a'')));
3
4  int main ()
5  {
6    int *p = &a, *q = &b;
7    *p = 1;
8    *q = 2;
9
10 //return b;
11   return a; // Bug: the program exits with 1
12 }
```

example 1: wrong code

```
1  int a = 0;
```

Marek Polacek 2016-02-25 09:13:03 UTC Comment 1

Hm, this really seems like a wrong code. Started a looooong time ago, before r104500.

Richard Biener 2016-02-25 10:22:43 UTC Comment 2

We have a duplicate for this - basically (most) of alias analysis doesn't handle aliases (hah).

```
11    return a; // Bug: the program exits with 1
12  }
```

example 2: gcc crash

```
1 struct s { char c[1]; };
2 struct s a, b, c;
3 int d; int e;
4
5 void bar (void)
6 {
7 //e ? (d==0 ? b : c).c : (e==0 ? b : c).c;
8   e ? (d==0 ? b : c).c : (d==0 ? b : c).c;
9 }
```

example 2: gcc crash

```
1 struct s { char c[1]; };
2 struct s a, b, c;
3 int d; int e;
4
5 void bar (void)
6 {
7 //e ? (d=       c).c : (e==0 ? b : c).c;
8   e ? (d=     b : c).c : (d==0 ? b : c).c;
9 }
```

Release blocking

example 3: clang crash

```
1  int a;
2  double b, *c;
3
4  void fn1(int p1) {
5    for (;; p1--) {
6      a = p1;
7      for (; p1 >= a; a--)
8        b = c[p1];
9    }
10 }
```

general, effective

- Found bugs in **many compilers**
  - Scala, Rust, Go, …
- Very **simple to apply**
- **No need to reduce**

Q: How about CompCert?

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;
}
```

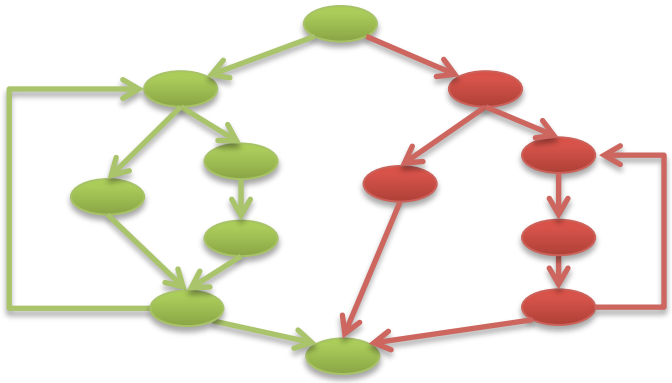
```
$ ccomp-2.7 test.c
...
Fatal error: exception File "backend/Regalloc.ml", line 741, ...
...
$
```

EMI & SPE: Finding 1400+ 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;
}
```



```
a := 10;
b := 1;
while(a) do
  a := a − b;
```
(a) Program P

```
□ := 10;
□ := 1;
while(□) do
  □ := □ − □;
```
(b) Skeleton $\mathbb{P}$

```
b := 10;
a := 1;
while(b) do
  b := b − a;
```
(c) Program P_1

```
a := 10;
b := 1;
while(b) do
  b := a − b;
```
(d) Program P_2

	GCC	LLVM	TOTAL
Reported	744	674	**1418**
Fixed	495	358	**853**