

# Client-Specific Equivalence Checking

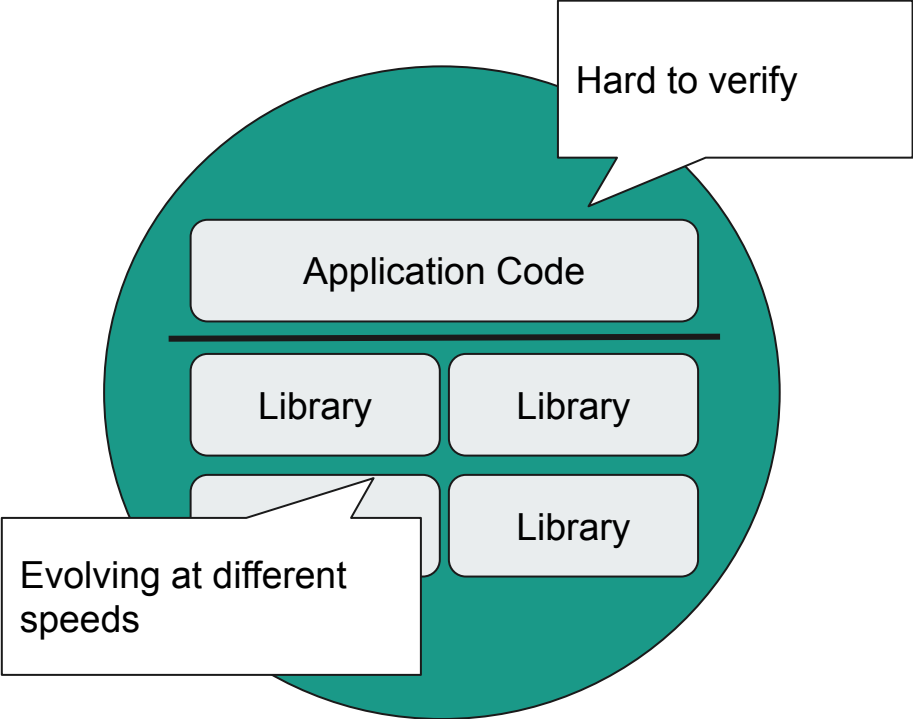
Federico Mora (University of Toronto)

Yi Li (Nanyang Technological University)

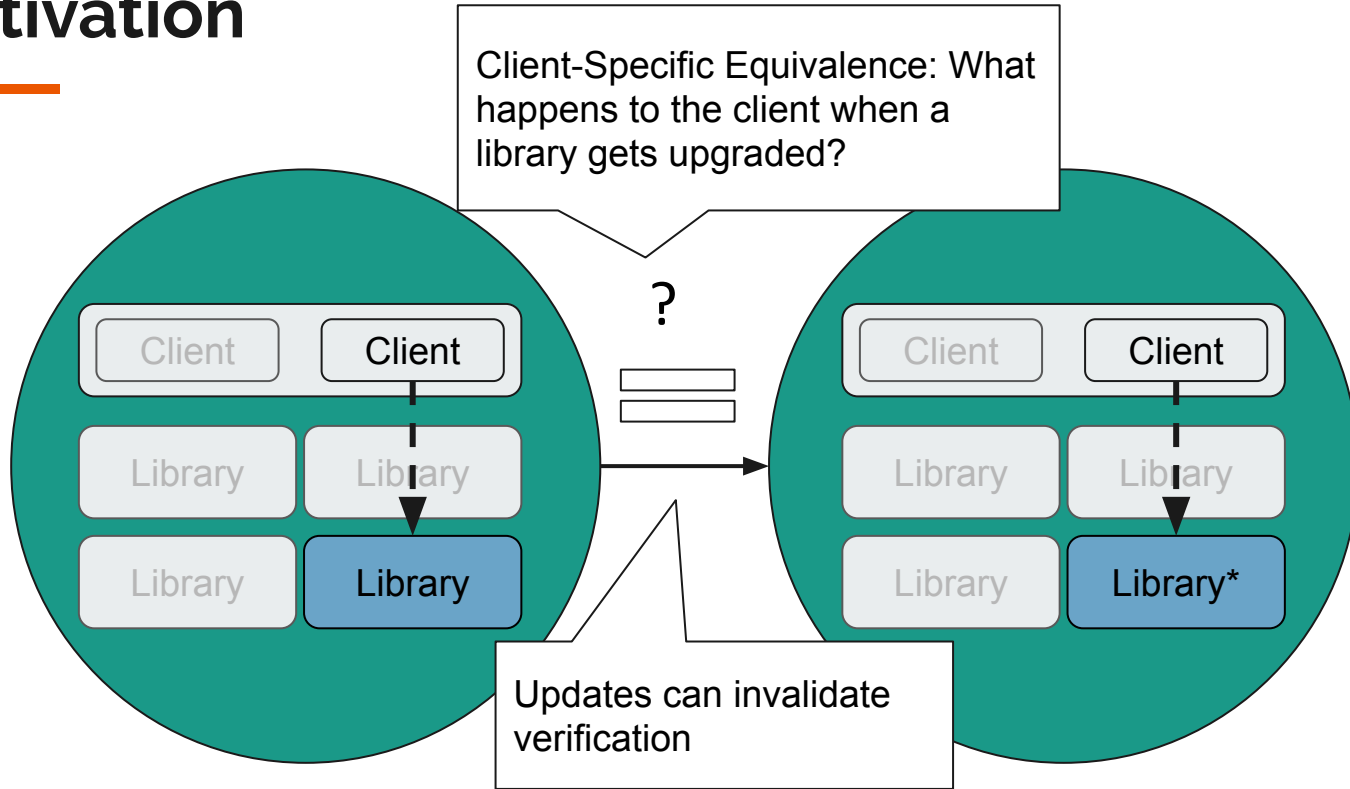
Julia Rubin (University of British Columbia)

Marsha Chechik (University of Toronto)

# Motivation



# Motivation



# Example

```
double mpf_get_d_2exp(signed long int *expptr, mpf_srcptr src) {
    mp_size_t size, abs_size;
    mp_srcptr ptr;
    int cnt;
    +double d;

    size = SIZ(src);
    if (UNLIKELY(size == 0))
    {
        *expptr = 0;
        return 0.0;
    }
    ptr = PTR(src);
    abs_size = ABS(size);
    count_leading_zeros(cnt, ptr[abs_size - 1]);
    cnt -= GMP_NAIL_BITS;
    *expptr = EXP(src) * GMP_NUMB_BITS - cnt;

    - return mpn_get_d(ptr, abs_size, 0, -(abs_size * GMP_NUMB_BITS - cnt));
    + d = mpn_get_d(ptr, abs_size, 0, -(abs_size * GMP_NUMB_BITS - cnt));
    + return size >= 0 ? d : -d;}

```

Library with changes

New version always returns positive number

```
REAL log_real(REAL x) {
    double d;
    double ln_app;
    signed long int exp;

    d = mpf_get_d_2exp(&exp, x.get_mpf_ptr());
    ln_app = (double)exp;
    return ln_app;
}

```

Client that is affected by change

Log of negative number undefined

Client that is **NOT** affected by change

```
double F_mpz_poly_eval_horner_d_2exp(
    long *exp, F_mpz_poly_t poly, double val)
{
    ... res = mpf_get_d_2exp(exp, output);
    // work around bug in earlier versions of GMP/MPPIR
    if ((mpf_sgn(output) < 0) && (res >= 0.0))
        res = -res;
    ...
}

```

Takes absolute value of output

---

**How Often Is a Client Unaffected  
by a Change?**

# Applicability Study



Inspected 66 client-library function pairs

- Popular libraries on GitHub (>1,000 stars)
- Written in C and Python
- Went through 100 most recent commits which do not alter signatures
  - mostly bug fixes and
  - new behaviour introductions
- Searched for unique clients on GitHub

# Applicability Study Results

~71% of the clients are unaffected

Projects	Library Functions	# Client	#Affected	#Unaffected
OpenSSL	RN_is_prime_fasttest_ex	10	5	5
OpenSSL	RSA_check_key	32	5	27
Linux	gcd	11	8	3
GMP	mpf_get_d_2exp	7	1	6
Delorean	Delorean	3	0	3
Delorean	Delorean2	3	0	3

---

# What's Wrong With Existing Solutions?



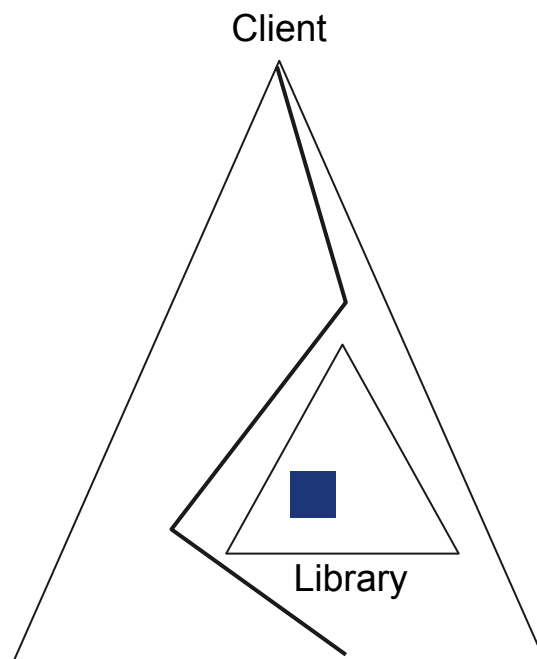
# Preliminaries

We consider partial functional equivalence

- Loops and recursion unrolled to configurable depth,  $d$
- Two unrolled programs  $P, P'$  are equal iff for all  $x, P(x) = P'(x)$

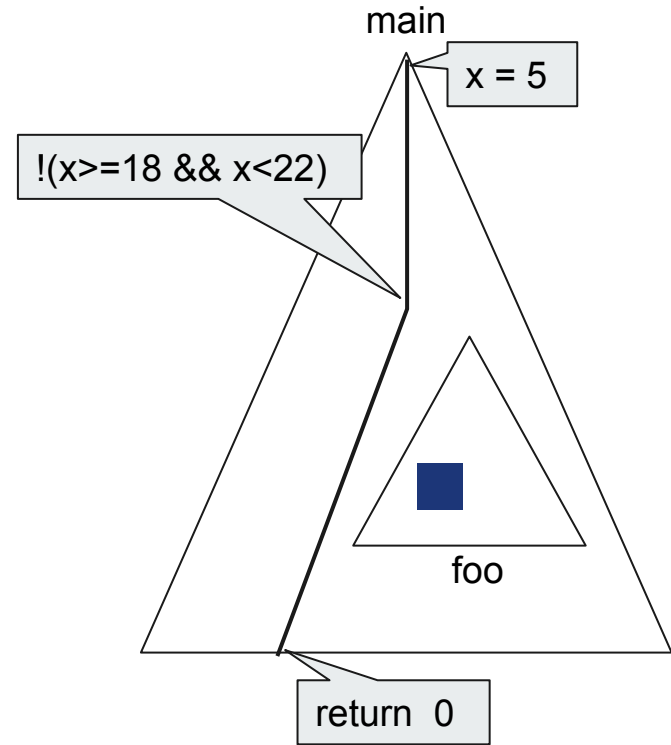
For this presentation we represent

- Programs as triangles
  - Single entry point at top
- Libraries calls are triangles inside a larger triangle
- Program paths are lines inside the triangles
- Updates are purple squares inside libraries

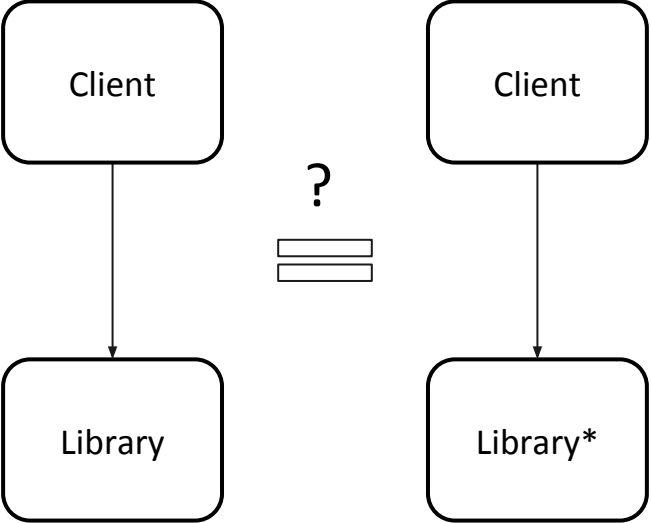
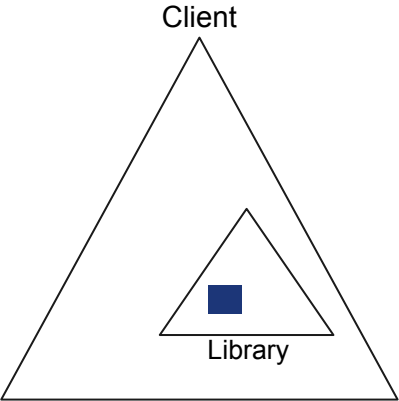


# Example Diagram

```
int main(int x) {  
    if (x>=18 && x<22)  
        return foo(x,20);  
    return 0;  
}  
  
int foo(int a, int b) {  
    int c=0;  
-   for (int i=1;i<=b;++i)  
-       c+=a;  
+   for (int i=1;i<=a;++i)  
+       c+=b;  
    return c;  
}
```



# Different Ways to Apply Existing Solutions

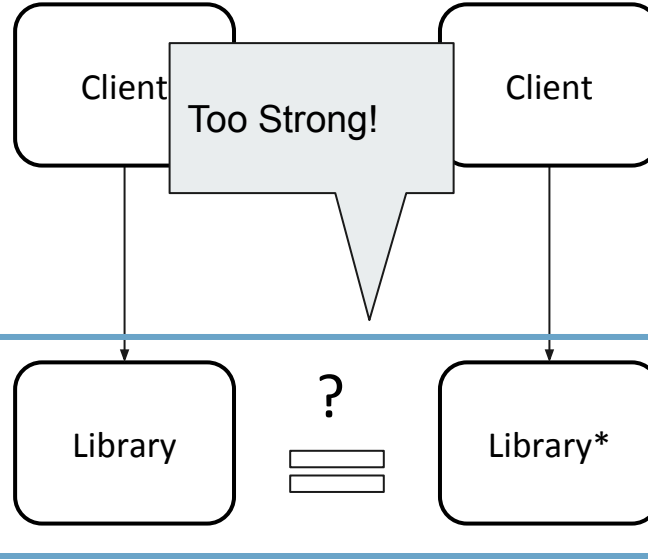
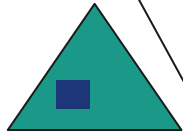


# Different Ways to Apply Existing Solutions

## 1. Checking Equivalence of Libraries

Exploring all of the library, and none of the client

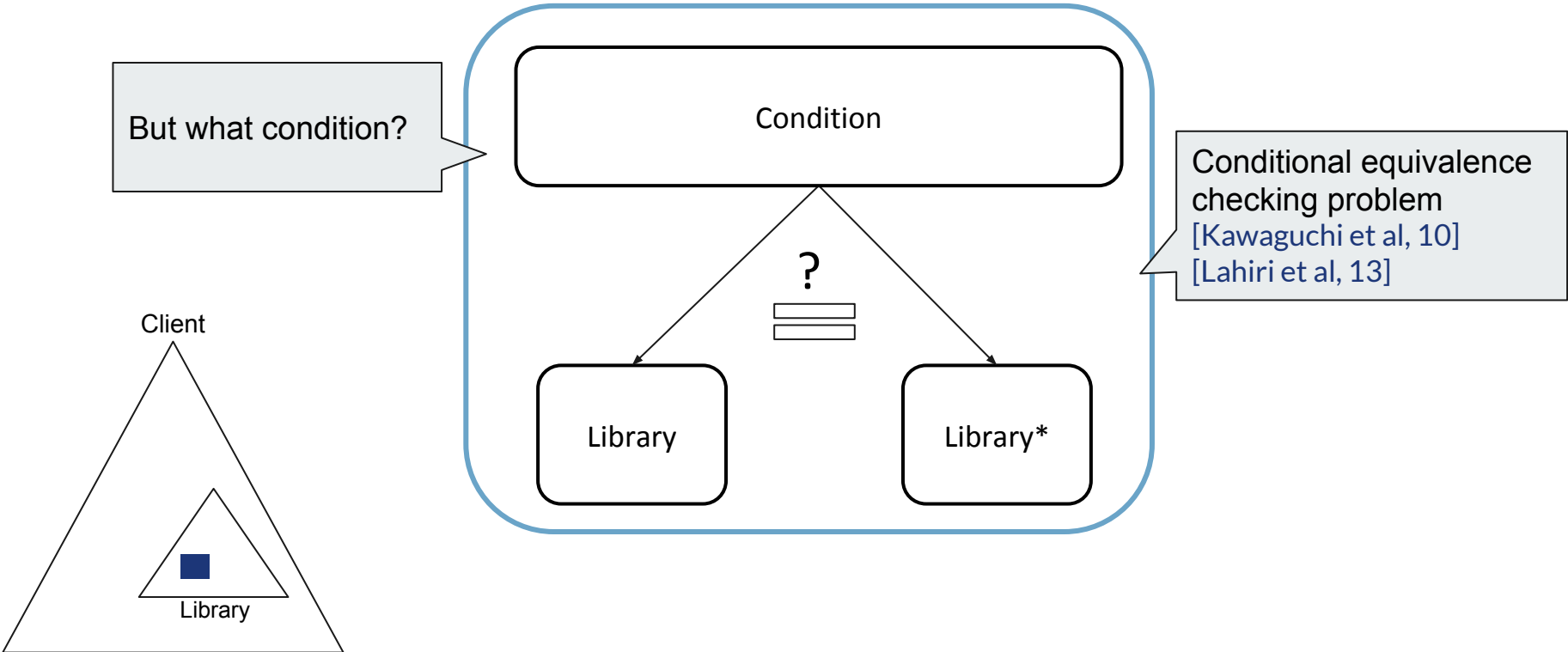
Client



Classic equivalence checking problem  
[Person, 08]

# Different Ways to Apply Existing Solutions

## 2. Checking Equivalence Of Libraries Under a Condition



# Different Ways to Apply Existing Solutions

## 2. Checking Equivalence Of Libraries Under a Condition

Exploring part of client that calls library, and part of library used by client

Client

Library

Let  $C$  be the set of all the calling contexts of the library in the client

For all  $c$  in  $C$ ,  $c[\text{lib}] = c[\text{lib}^*]$ ?

Library

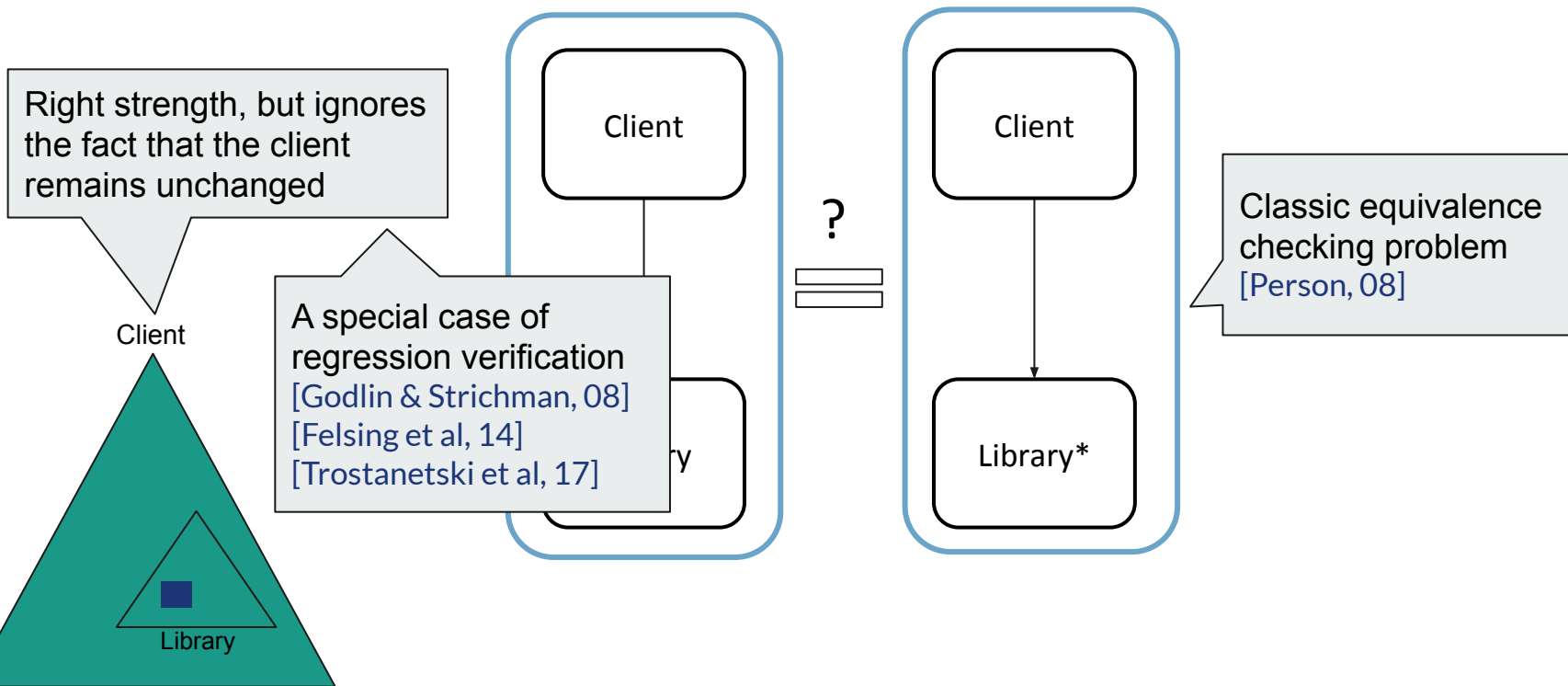
Library\*

But too strong again!

- $\text{lib}(x) = -\text{lib}'(x)$
- $\text{client} := \text{lib}(1) + \text{lib}(-1)$

# Different Ways to Apply Existing Solutions

## 3. Checking Equivalence of Client-Library Pairs



# Different Ways to Apply Existing Solutions



## 3. Client Equivalence of Client-Library Pairs

Ignores  
client re

ence  
m

Can we do better?

Client

Library



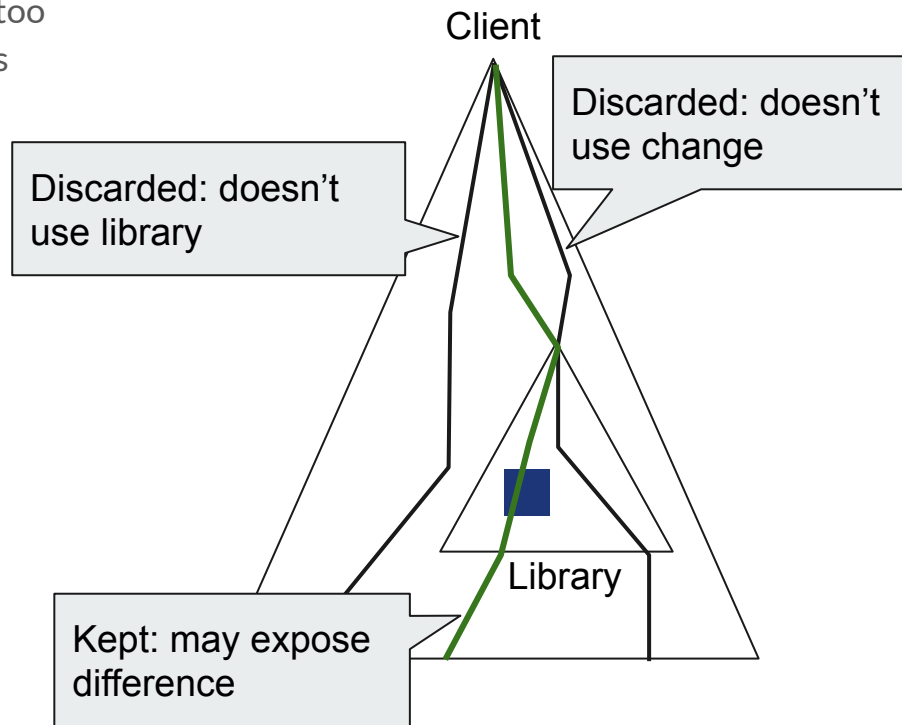
---

# Our Technique In A Nutshell

# Client-Specific EquiValence CheckER

Insight: existing techniques are too strong, or consider too much. To get the most precise and efficient analysis let's consider only

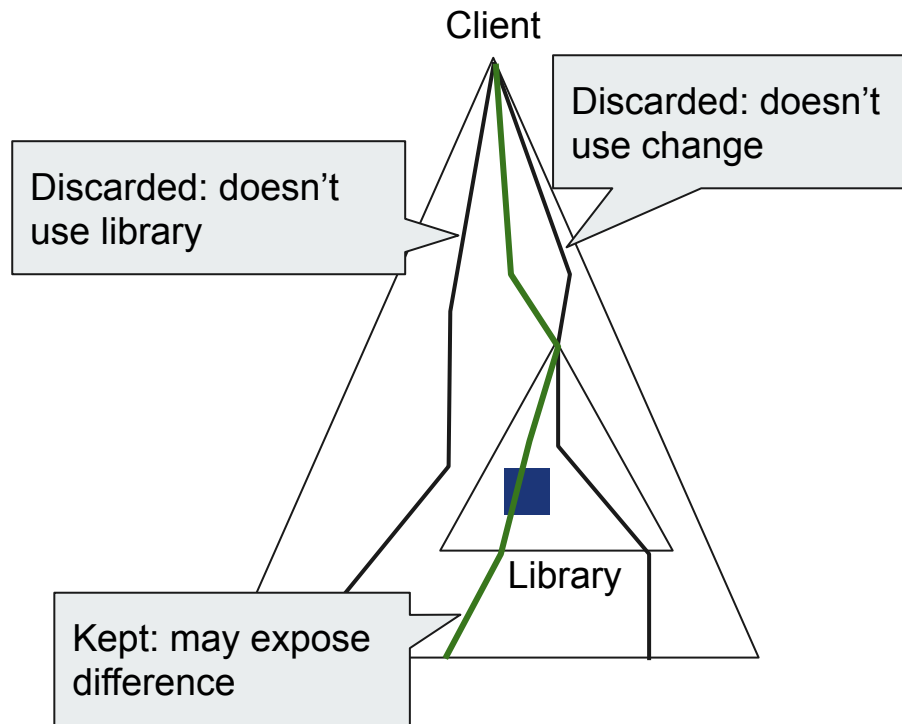
- how the client uses the library and
- where the library change is active.



# CLEVER

## Algorithm

- Explore Client with library uninterpreted
  - Collect uses/contexts of the library
- For each client context
  - Explore the library restricted to this context
  - If change is inactive, discard
  - Else, check for quick counterexample
    - If counterexample found, return
    - Else store paths
- Create equivalence assertion from stored paths
- Dispatch to existing verifier, or SMT solver



# Example Savings

```
int main(int x) {  
    if (x>=18 && x<22)  
        return foo(x,20);  
    return 0;  
}
```

How the client uses the library

```
int foo(int a, int b) {  
    int c=0;  
-   for (int i=1;i<=b;++i)  
-       c+=a;  
+   for (int i=1;i<=a;++i)  
+       c+=b;  
    return c;  
}
```

Saves us from computing a non-linear loop invariant:  $c == a*b == a*b'$

main

$x \geq 18 \ \&\& \ x < 22$

foo



# Evaluation

# Implementation & Evaluation



Available at: <https://github.com/Client-Specific-Equivalence-Checker/CLEVER>

Explores client contexts using symbolic execution

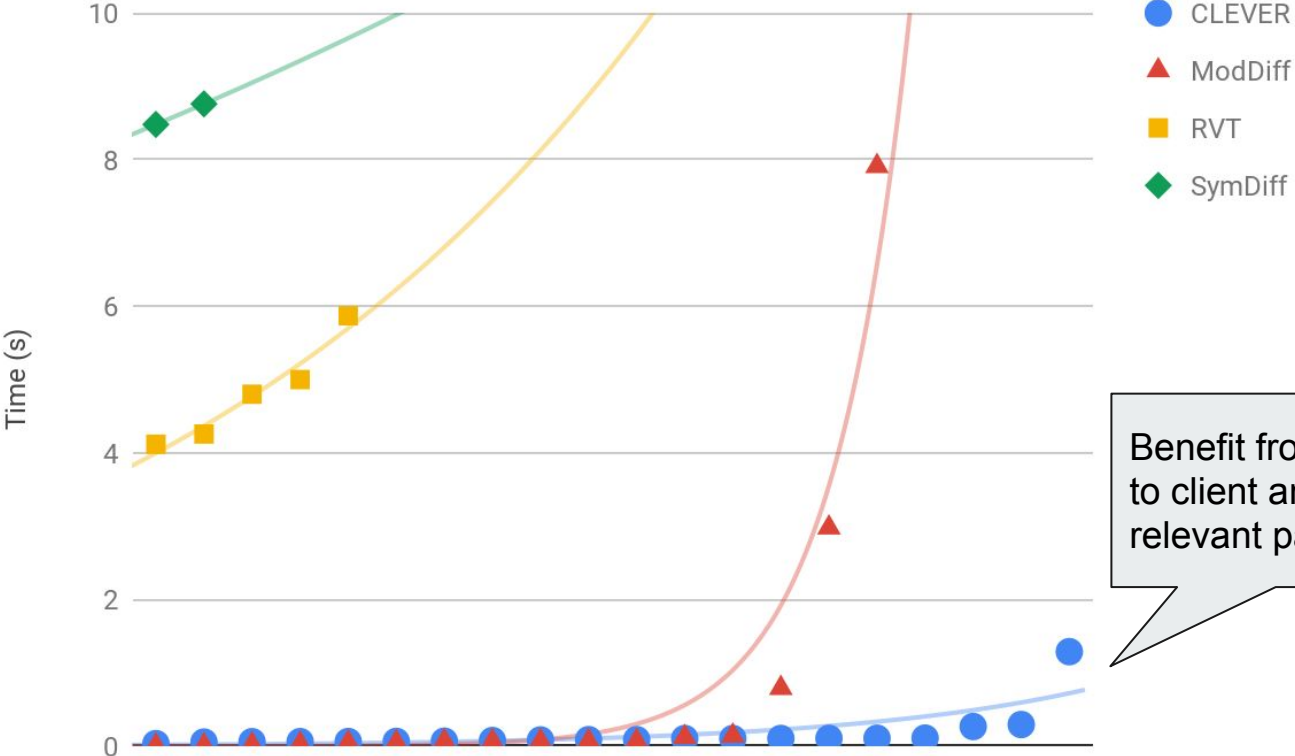
- PyExSMT (<https://github.com/FedericoAureliano/PyExSMT>)

We compare with SymDiff, RVT, and ModDiff (treating client-lib pair as a whole).

Subjects:

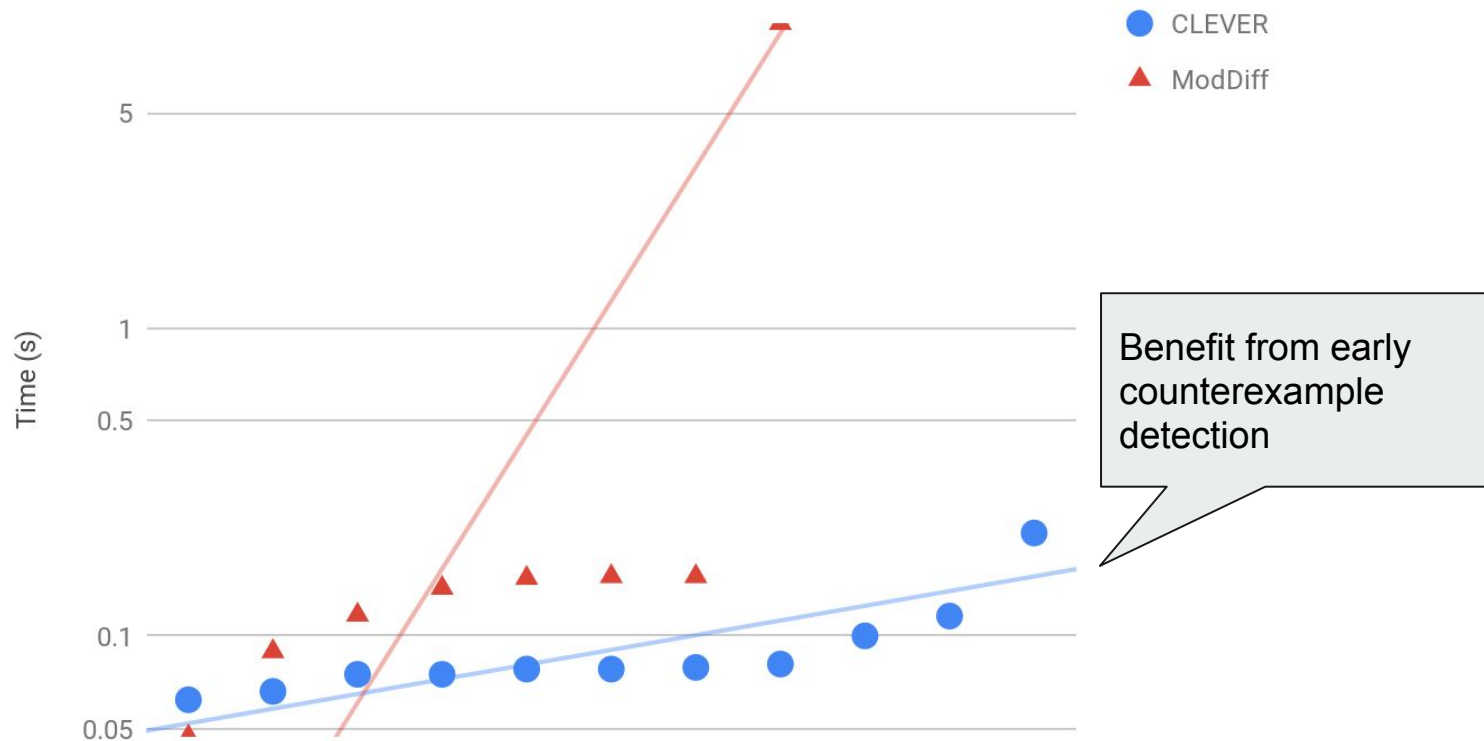
- 39 client-library pairs with library updates (23 equivalent / 16 inequivalent)
- 23 come from the ModDiff suite (small programs)
- 16 come from our pre-study

# Cactus Plot: Equivalent Cases



Benefit from pruning to client and change relevant paths

# Cactus Plot (Log Scale): Non-Equivalent Cases





---

# Conclusions & Beyond

# Summary



We consider a special case of equivalence where usage patterns can be exploited

- We show that this special case is relevant
- Devise an extension/enhancement to classic regression verification
  - Optimized for early discovery of counterexamples
- It does well when compared against the state-of-the-art

Lots of details are not considered, yet

- Go beyond functional equivalence
  - Total path equivalence: maintaining all intermediate executions of the client etc.
- Improvements on usability
  - Explain reasons for equivalence
  - Suggest changes/updates to clients

Benchmark size is still quite limited

- Call backs, side effects, heap, etc.
- Increase support for primitive types
  - E.g. floating-point numbers, strings, and algebraic datatypes

# Thank You!



CLEVER available at <https://github.com/Client-Specific-Equivalence-Checker/CLEVER>

Benchmarks and more available at <https://client-specific-equivalence-checker.github.io/>

PyExSMT available at <https://github.com/FedericoAureliano/PyExSMT>

