Test Generation For Higher-Order Functions In Dynamic Languages

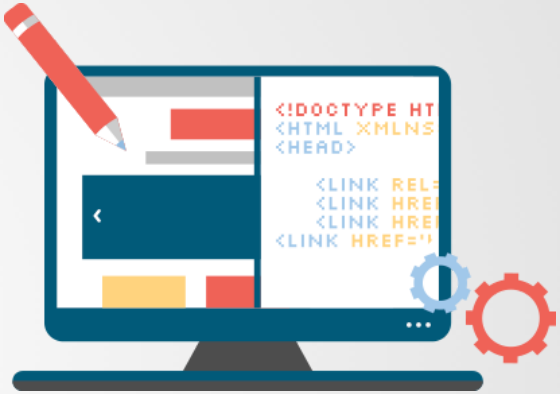
Marija Selakovic, Michael Pradel, Rezwana Karim, Frank Tip

OOPSLA 2018

TECHNISCHE
UNIVERSITÄT
DARMSTADT

Automatic Test Generation

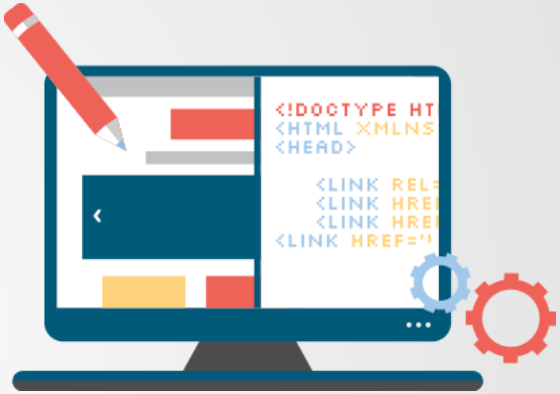
Testing



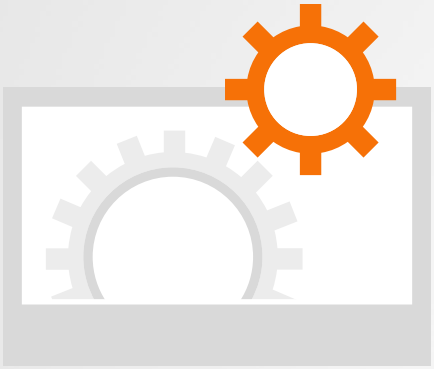
Automatic Test Generation

Testing

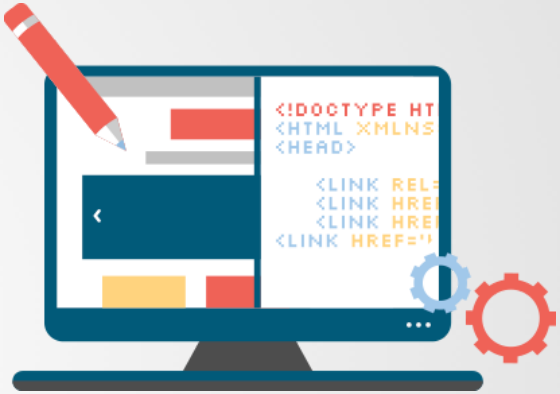
Limited resources



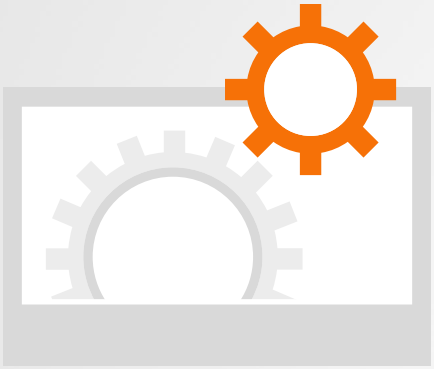
Automatic Test Generation



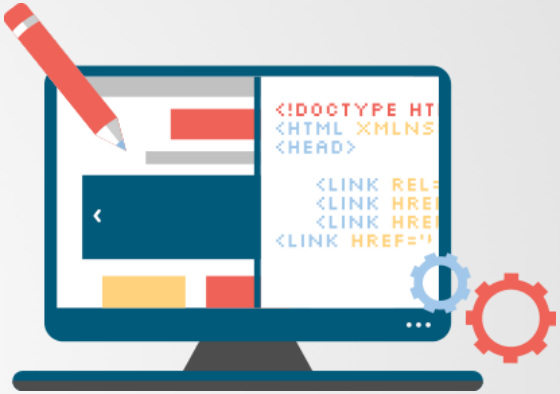
Testing



Automatic Test Generation

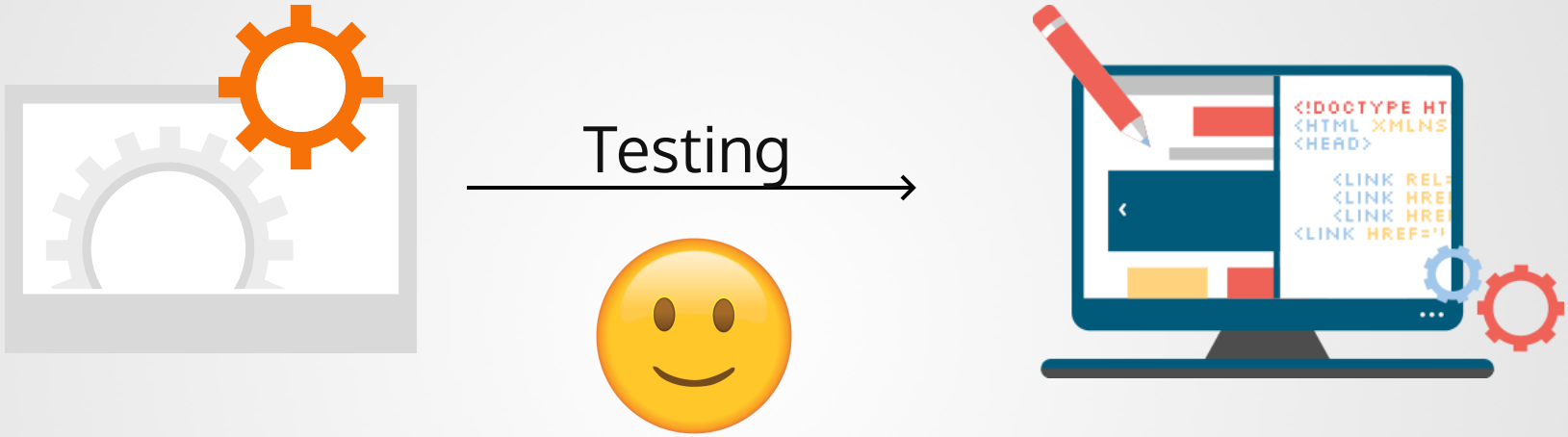


Testing



- Generates test cases in an automated way
- Reduces manual effort
- Effective in finding programming erros

Automatic Test Generation



- Generates test cases in an automated way
- Reduces manual effort
- Effective in finding programming erros

Example

Function:

```
function includes(str, s) {
  if (s === '') return true;
  return str.indexOf(s) != -1;
};
```

Tests:

```
includes('car','a'); // true
includes('car',''); // true
includes('car', 'km'); //false
```

```
function size(arr) {
  if (!arr) return 0;
  return arr.length;
};
```

```
size([1,2,3]); // 3
size(undefined); // 0
size([]); //0
```

Higher-Order Functions (HOF)

HOF: takes other functions as inputs that are *called back*

Function:

```
hof(x,cb){
  var r = cb(x);
  if (r && x.a){
    .....
  }
  ....
}
```

Test case:

?

Challenges

Goal: Generate Effective Tests for Higher-Order Functions

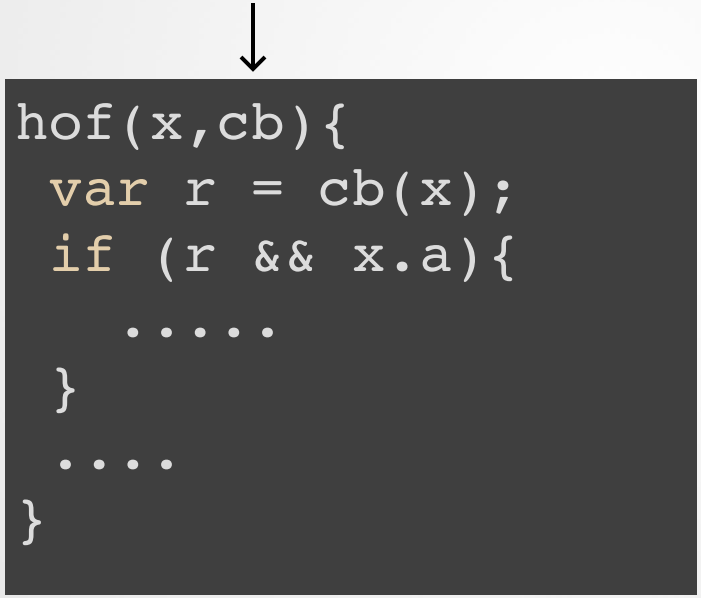
```
hof(x,cb){
 var r = cb(x);
 if (r && x.a){
    .....
 }
 ....
}
```

?

Challenges

Goal: Generate Effective Tests for Higher-Order Functions

Challenge 1:
callback position?

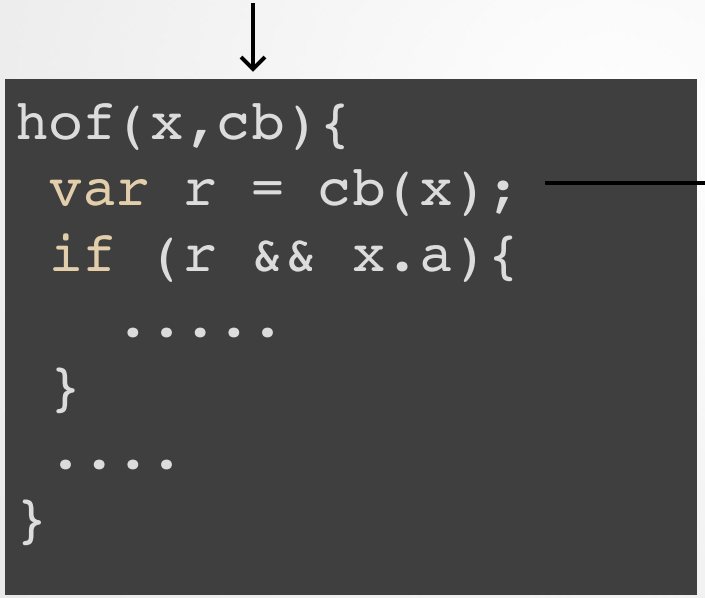


```
hof(x,cb){
 var r = cb(x);
 if (r && x.a){
    .....
 }
 ....
}
```

Challenges

Goal: Generate Effective Tests for Higher-Order Functions

Challenge 1:
callback position?



```
hof(x,cb){
 var r = cb(x);
 if (r && x.a){
    .....
 }
 ....
}
```

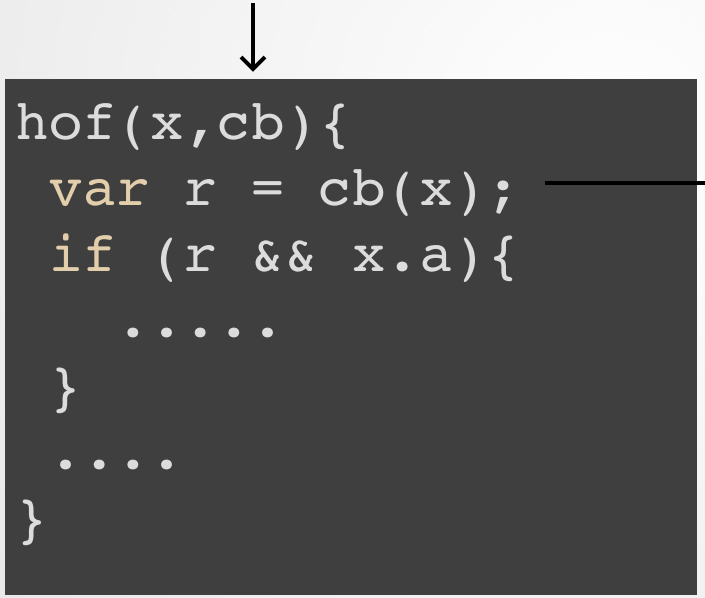
Challenge 2: callback that interacts with program?

```
cb(x){
    ?
}
```

Challenges

Goal: Generate Effective Tests for Higher-Order Functions

Challenge 1:
callback position?



```
hof(x,cb){
 var r = cb(x);
 if (r && x.a){
    .....
 }
 ....
}
```

Challenge 2: callback that interacts with program?

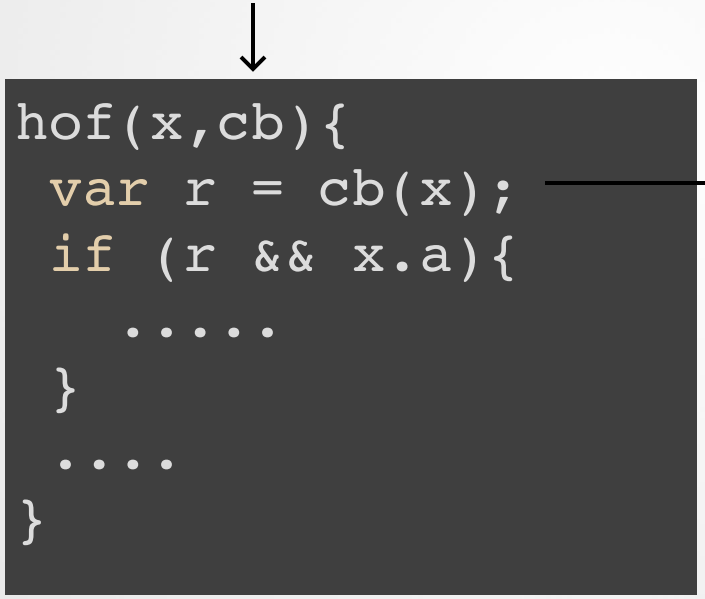
```
cb(x){
    ?
}
```

Challenge 3: Chaining multiple calls?

Challenges

Goal: Generate Effective Tests for Higher-Order Functions

Challenge 1:
callback position?



```
hof(x,cb){
 var r = cb(x);
 if (r && x.a){
    .....
 }
 ....
}
```

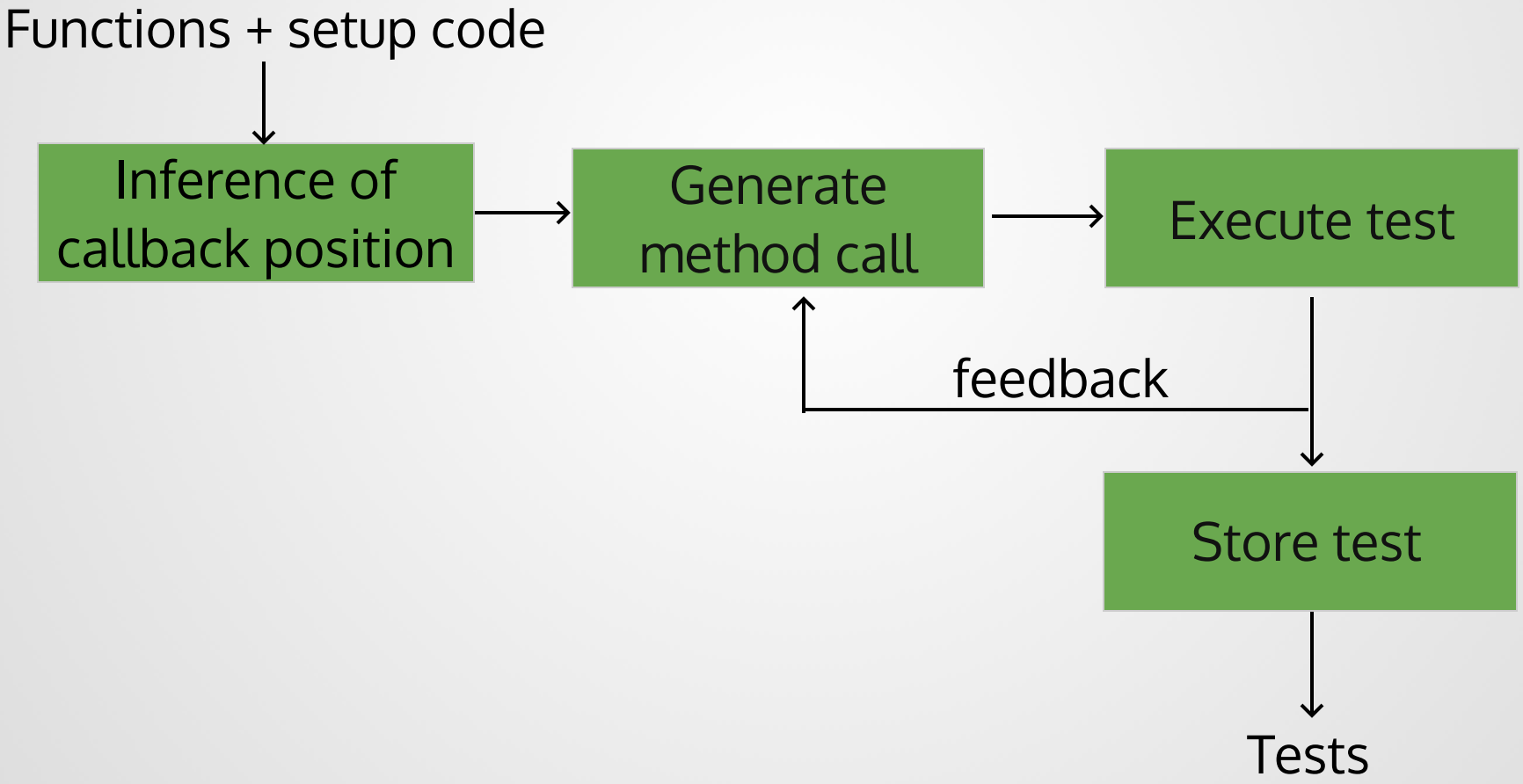
Challenge 2: callback that interacts with program?

```
cb(x){
    ?
}
```

Challenge 3: Chaining multiple calls?

Challenge 4: Callback related differences?

LambdaTester: Framework For Testing Higher-Order Functions

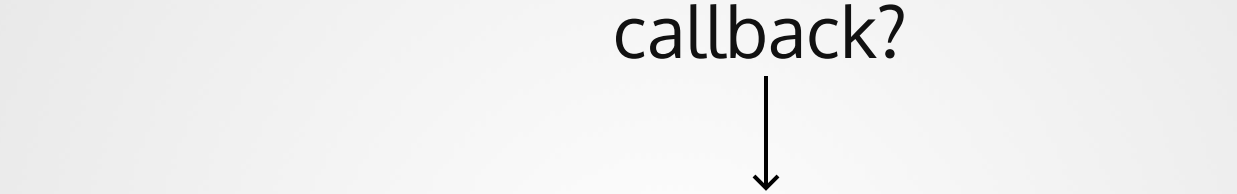


Inference of Callback Positions

higherOrderFunction(arg_1, arg_2,...,arg_n)

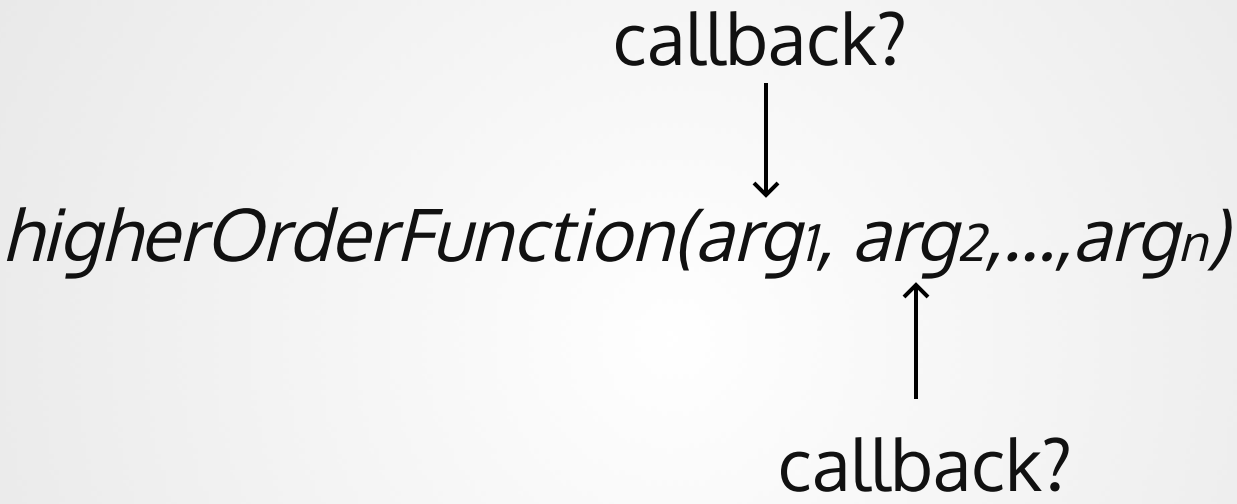
Is callback executed? Positions

Inference of Callback Positions



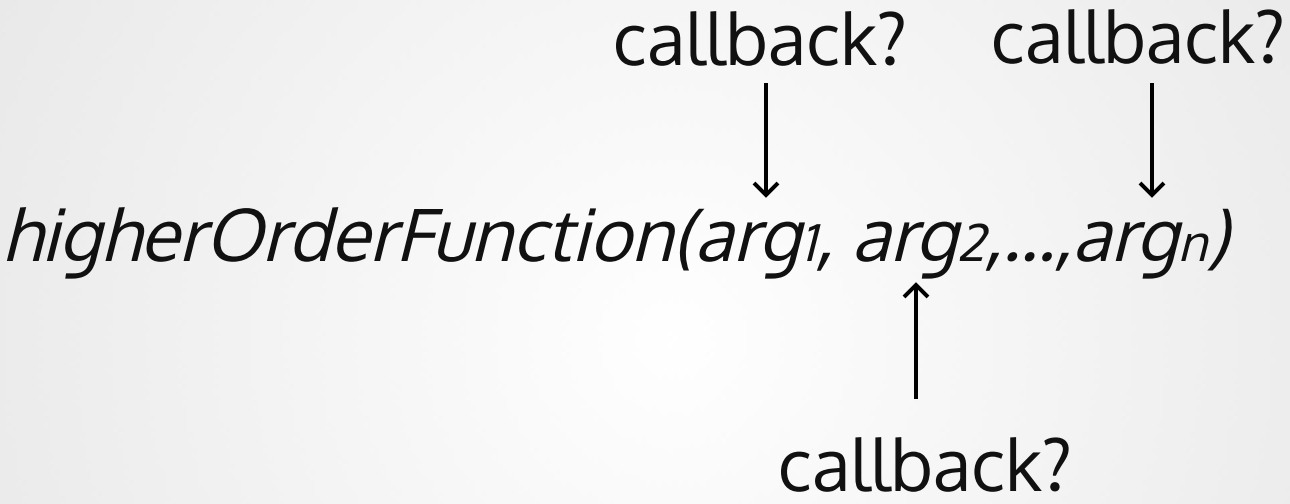
Is callback executed?	Positions
Yes	[1]

Inference of Callback Positions



Is callback executed?	Positions
Yes	[1]
Yes	[1, 2]

Inference of Callback Positions

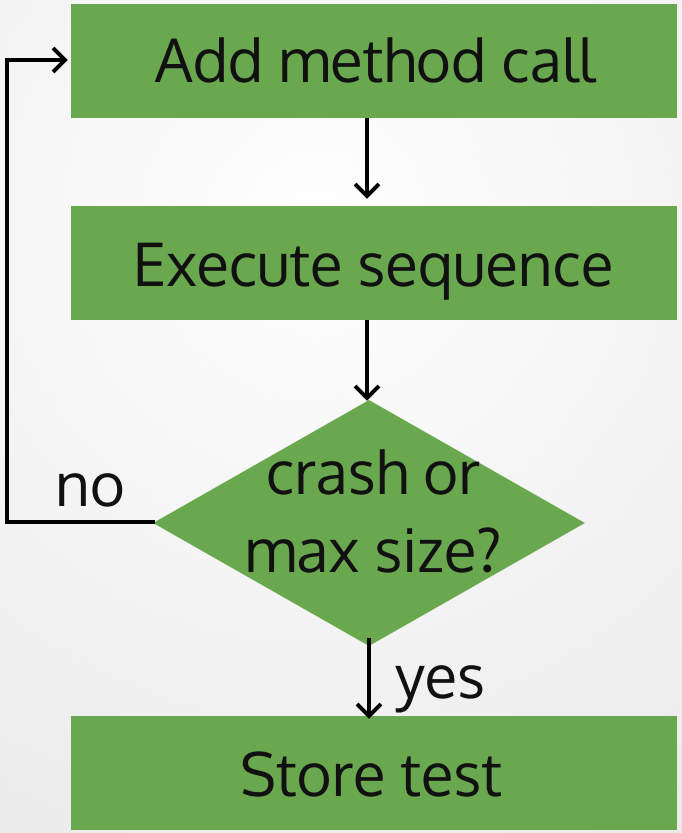


Is callback executed?	Positions
Yes	[1]
Yes	[1, 2]
No	[1, 2]

Background: Feedback-Directed Test Generation

Feedback[1]:

- no crash
- return values



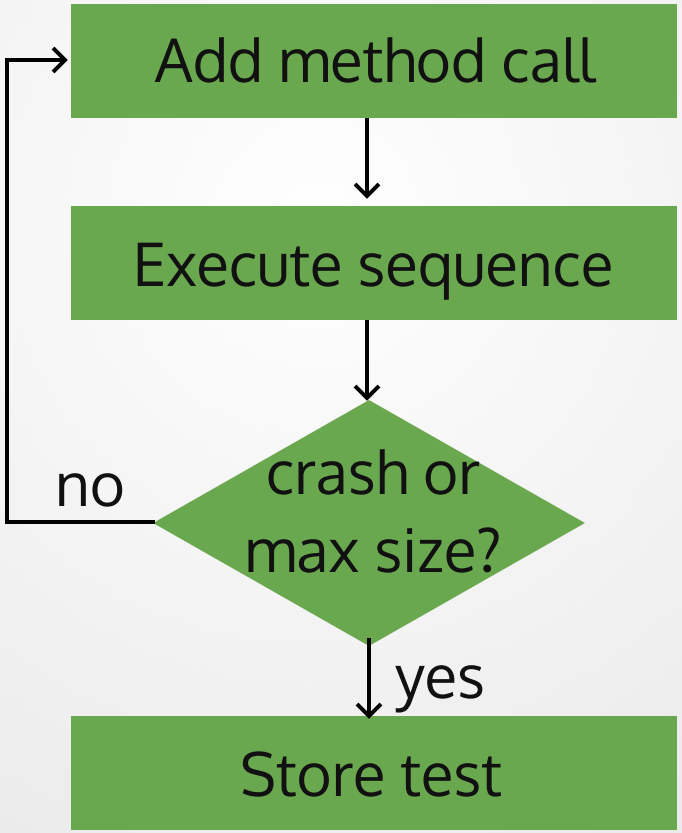
[1] Carlos Pacheco, Shuvendu K. Lahiri, Thomas Ball, Michael D. Ernst. *Feedback-Directed Random Test Generation*, ICSE '07

Background: Feedback-Directed Test Generation

Test: Setup code + Sequence of method calls

Feedback[1]:

- no crash
- return values



[1] Carlos Pacheco, Shuvendu K. Lahiri, Thomas Ball, Michael D. Ernst. *Feedback-Directed Random Test Generation*, ICSE '07

Callback Generation Approaches

```
function () {

}
```

cb-empty

```
function () {
  return 17;
}
```

cb-quick[2]

```
function (err) {
  if (err)
  throw err;
}
```

cb-mined

cb-writes

[2] Koen Claessen and John Hughes. 2011. *QuickCheck: A Lightweight Tool for Random Testing of Haskell Programs*. SIGPLAN Not. 46

Callback Generation Approaches

```
function () {

}
```

cb-empty

- no computation
- no return value

```
function (err) {
  if (err)
  throw err;
}
```

cb-mined

Callback Generation Approaches

- returns random values
- no additional computation

```
function () {
  return 17;
}
```

cb-quick[2]

cb-writes

[2] Koen Claessen and John Hughes. 2011. *QuickCheck: A Lightweight Tool for Random Testing of Haskell Programs*. SIGPLAN Not. 46

Callback Generation Approaches

```
function () {

}
```

cb-empty

```
function (err) {
  if (err)
  throw err;
}
```

cb-mined

- extracted from existing code
- function expressions passed to methods with the same name

Callback Generation Approaches

```
function () {

}
```

cb-empty

```
function () {
  return 17;
}
```

cb-quick

```
function (err) {
  if (err)
  throw err;
}
```

cb-mined

cb-writes

Dynamic Analysis

```
hof(x,cb){
  ...
 var r = cb(x);
 if (r && x.a){
    ...
 }
 ...
}
```

```
cb(x){
    ?
}
```

Dynamic Analysis

- Analysis of memory reads after callback executes

```
hof(x,cb){
  ...
 var r = cb(x);
 if (r && x.a){
    ...
 }
 ...
}
```

```
cb(x){
    ?
}
```

Dynamic Analysis

- Analysis of memory reads after callback executes

```
hof(x,cb){
  ...
  var r = cb(x);
  if (r && x.a){
    ...
  }
  ...
}
```

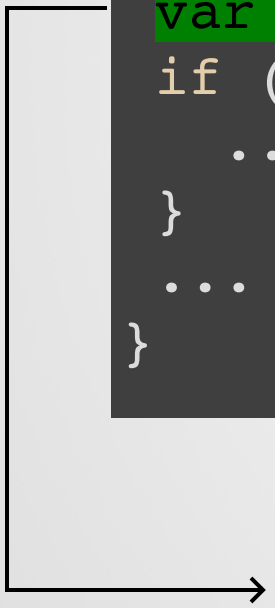
```
cb(x){
    ?
}
```

Dynamic Analysis

- Analysis of memory reads after callback executes

```
hof(x,cb){
  ...
  var r = cb(x);
  if (r && x.a){
    ...
  }
  ...
}
```

```
cb(x){
    ?
}
```



- r
- x.a

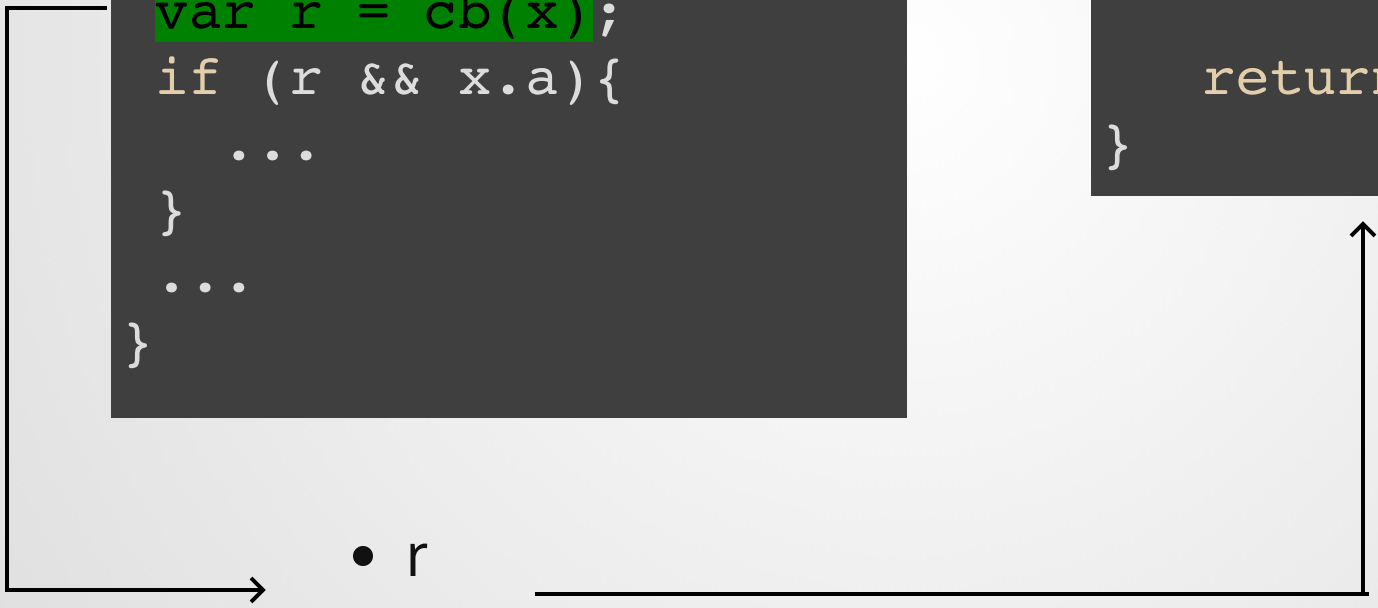
Dynamic Analysis

- Analysis of memory reads after callback executes

```
hof(x,cb){
  ...
  var r = cb(x);
  if (r && x.a){
    ...
  }
  ...
}
```

```
cb(x){
    x.a = true;

    return 23;
}
```

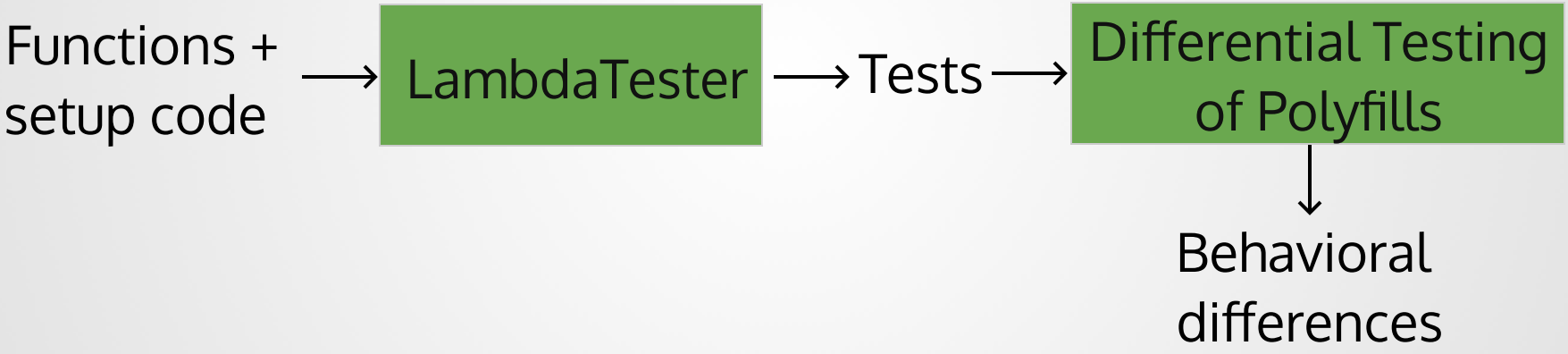


- r
- x.a

Application: Differential Testing of Polyfills

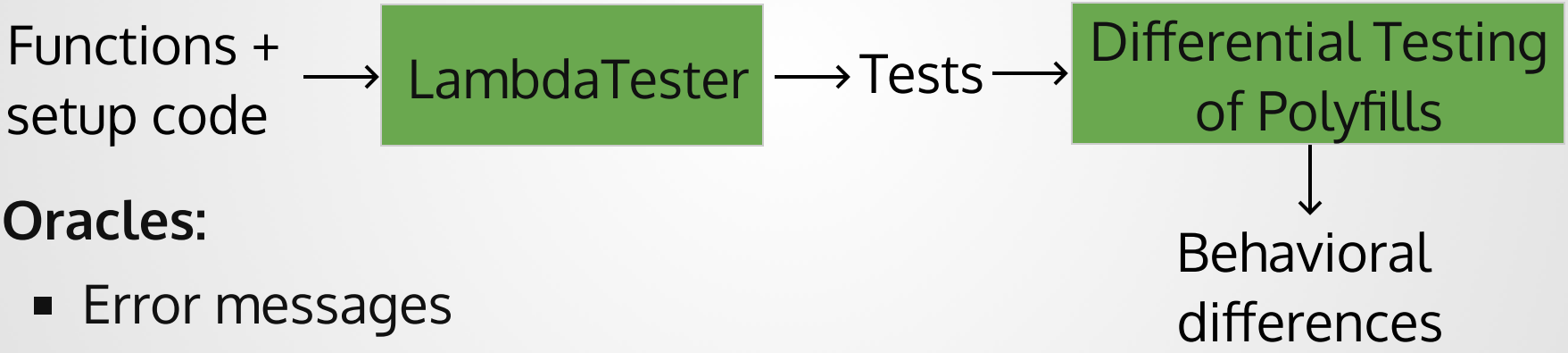
Application: Differential Testing of Polyfills

- **Polyfill**: implementation of an API not supported by older browsers
- **Differential testing**: testing the same program on different implementations



Application: Differential Testing of Polyfills

- **Polyfill**: implementation of an API not supported by older browsers
- **Differential testing**: testing the same program on different implementations



- **Oracles:**
  - Error messages
  - Non-termination
  - Standard output
  - Receiver and return object
  - Callback arguments
  - Number of callback invocations

Evaluation

Setup:

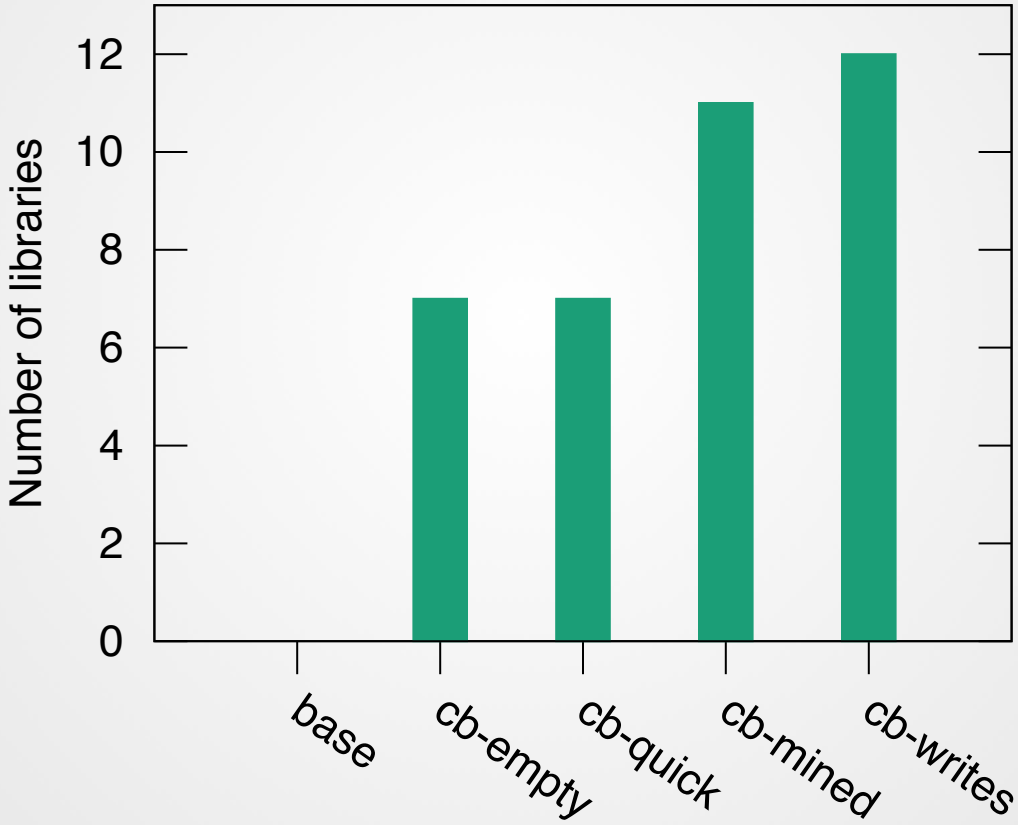
- Polyfill.io, es5shim, mozilla and 10 Promise libraries
- Base approach: unaware of callbacks
- 1000 generated tests for each polyfill/approach combination

Research Questions:

- **Effectiveness?** *LambdaTester* finds differences in 12 libraries
- **Coverage?** *LambdaTester* achieves on average 75% statement coverage
- **Efficiency?** Time to generate single test ranges between 0.12 and 12 seconds

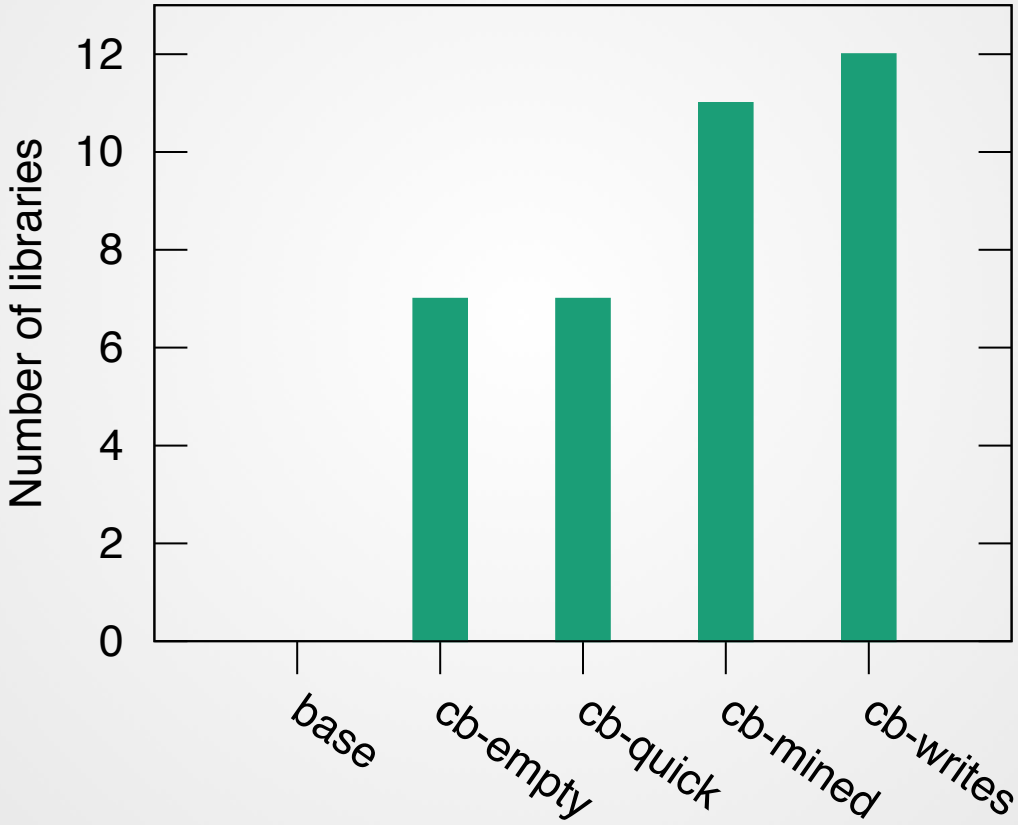
Effectiveness

- In how many libraries LambdaTester finds behavioral differences?



Effectiveness

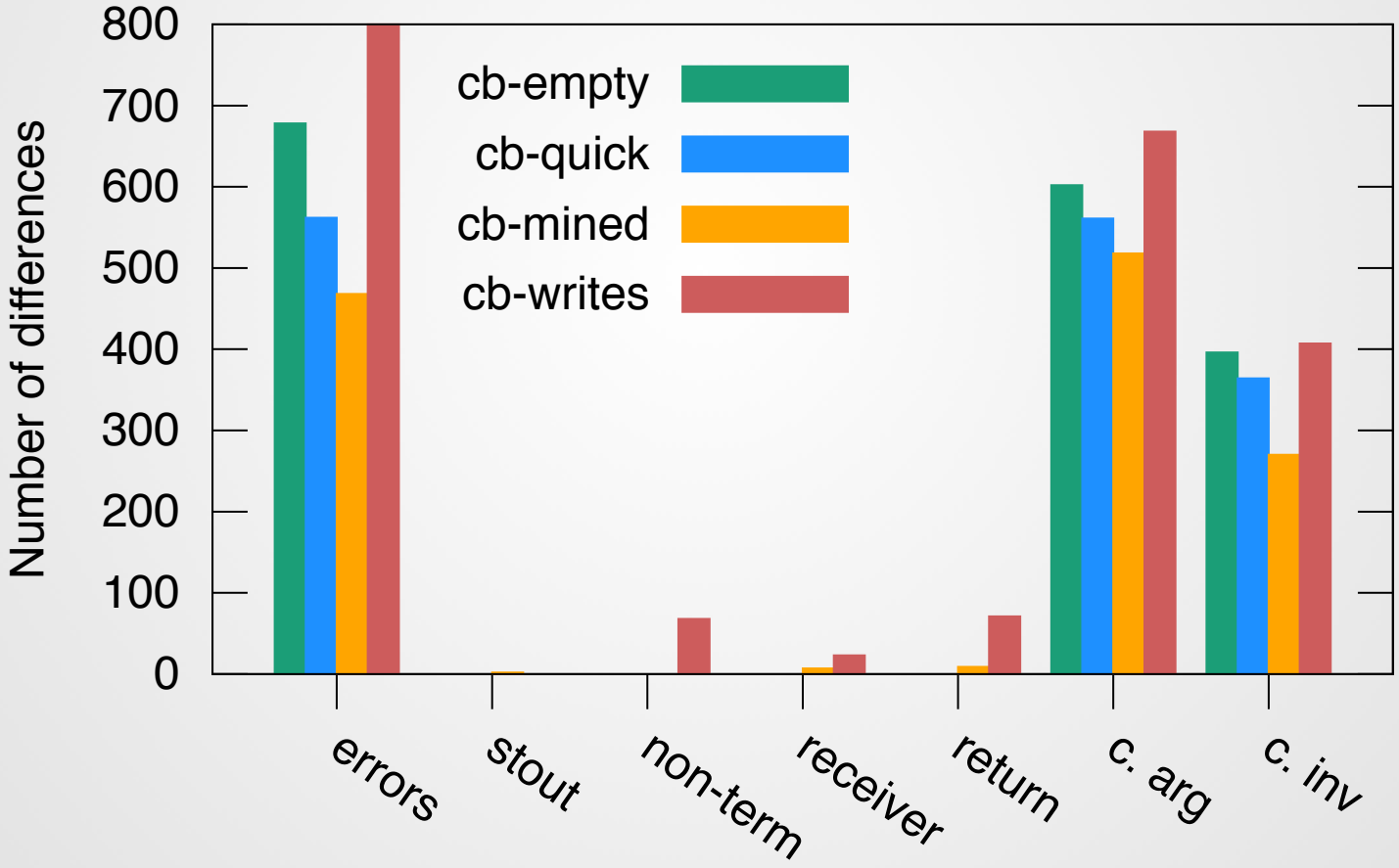
- In how many libraries LambdaTester finds behavioral differences?



- cb-writes finds behavioral differences in 12 out of 13 libraries

Behavioral Differences

- What types of behavioral differences LambdaTester finds?



Example: Q library

```
var Promise = require('q');
var p1 = Promise.resolve(18);
var p2 = Promise.reject(17);

var r1 = p1.then(function(){ return null; }, null);
var r2 = p2.then(function(){ return r1; });
var r3 = r2.catch(function(){ return p2; });
var r4 = r1.then(function(){  return r4; }); //non-termination
```

Example: Q library

```
var Promise = require('q');
var p1 = Promise.resolve(18);
var p2 = Promise.reject(17);

var r1 = p1.then(function(){ return null; }, null);
var r2 = p2.then(function(){ return r1; });
var r3 = r2.catch(function(){ return p2; });
var r4 = r1.then(function(){  return r4; }); //non-termination
```

Output*:*

- Native*: TypeError: Chaining cycle detected*
- Q*: No output, non-termination*

Example: Polyfill.io

```
var base = ["w", "I", 126];
base.map(function(a,b,c){
    base['length'] = false;
    return a;
});
```

write to location suggested by dynamic analysis

Example: Polyfill.io

```
var base = ["w", "I", 126];
base.map(function(a,b,c){
    base['length'] = false;
    return a;
});
```

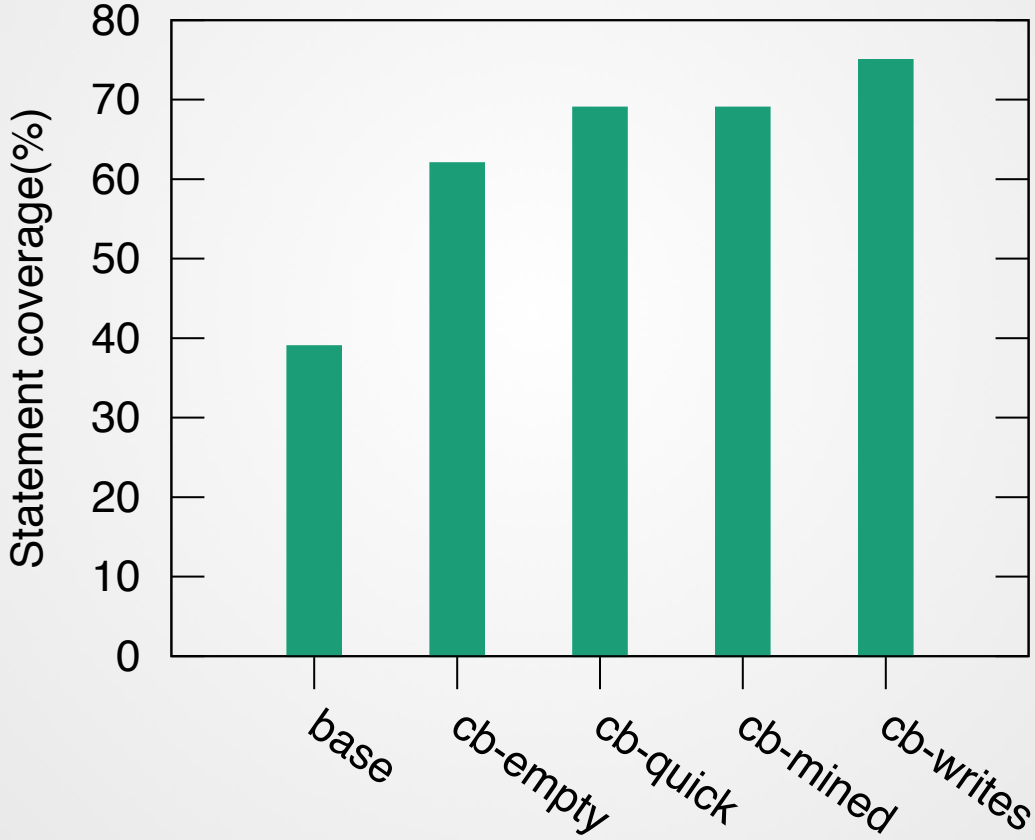
write to location suggested by dynamic analysis

Output *:*

- Native *: return object = ["w", undefined, undefined]*
- Polyfill.io *: return object = ["w"]*

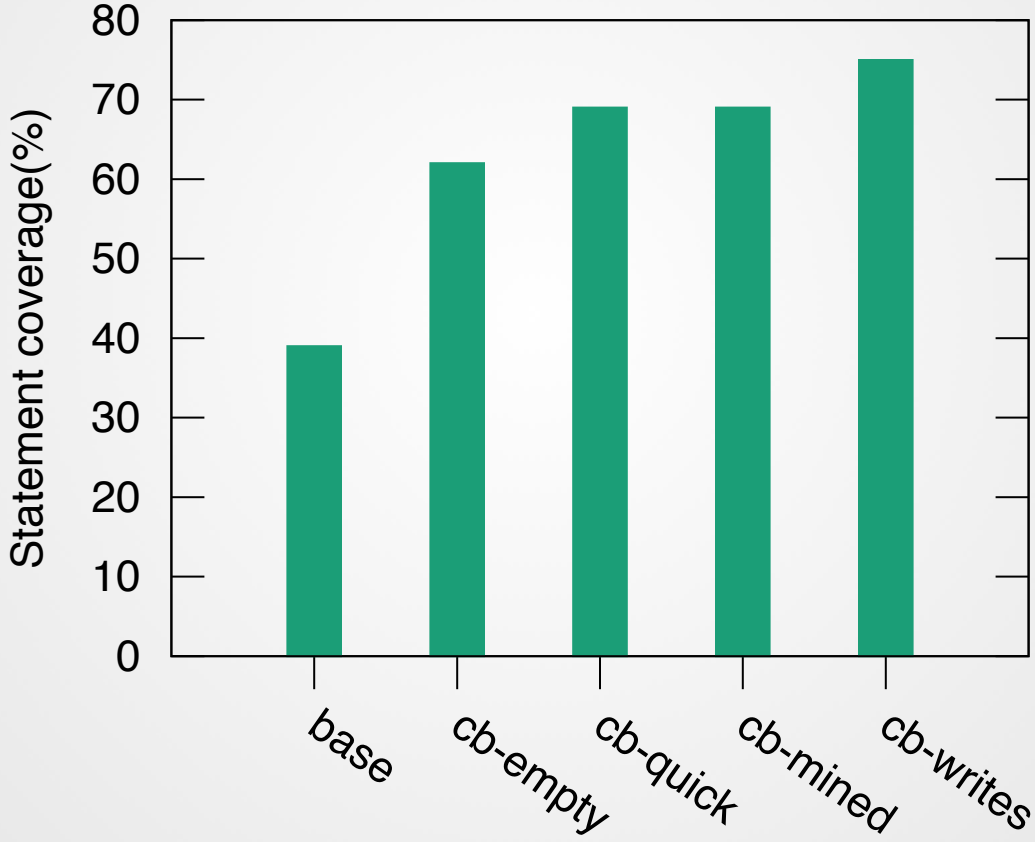
Coverage

- How much code is covered by generated tests?



Coverage

- How much code is covered by generated tests?



- cb-writes on average achieves 75% statement coverage

Conclusions

LambdaTester: Test generation for higher-order functions

- Automatic and feedback-directed
- Generates tests with non-trivial callbacks
- Several instances of the framework based on callback type
- Effective in:
  - Finding unknown behavioral differences
  - Increasing code coverage

https://github.com/sola-da/LambdaTester

Conclusions

LambdaTester: Test generation for higher-order functions

- Automatic and feedback-directed
- Generates tests with non-trivial callbacks
- Several instances of the framework based on callback type
- Effective in:
  - Finding unknown behavioral differences
  - Increasing code coverage

THANK YOU!

https://github.com/sola-da/LambdaTester

Setup code

Promise libraries:

```
var p1 = Promise.resolve(18);
var p2 = Promise.reject(17);
var p3 = Promise.resolve(null);
```

Array libraries:

```
var a1 = [1,2,3,4];
var a2 = new Array(10);
```