Debun: Detecting Bundled JavaScript Libraries on Web using Property-Order Graphs

Seojin Kim*, Sungmin Park*, and Jihyeok Park

Department of Computer Science and Engineering, Korea University, South Korea

Artifact Video

1. Background

- Detecting **vulnerable JavaScript libraries** in web applications is essential for security.
- **Bundlers** (i.e., Webpack) **modify and compress JS code**, which complicates library detection.

user code
libraries
terser
BABEL
...
Web App
Bundlers with Transpilers
Bundled App

Existing approaches

- **LDC** (Library Detector for Chrome)
  - Manually collected **property patterns** and check them **at runtime**

```
typeof (_ = window._) == "function")
typeof (chain = _ && _.chain) == "function"
```

- **PTDetector** (ASE'23)
  - Automatic extraction of **property patterns**
- **Limitation**
  - Prior work **may miss libraries** not revealed to window object
  - Prior work does **NOT utilize bundled code** to detect libraries
  - Why? It is **difficult to check code equivalence** correctly, precisely, and quickly

```
// loadsh v4.17.21
(function() {
...
loadsh.chain = function () {
  ...
}
...
window._= loadsh;
...
}.call(this);
```

2. Key Idea - Property-Order Graph

- **Observation** - What is **preserved** after **code transpilation** through bundlers?

1. Property names are *preserved* **to support JavaScript's** *dynamic property access*

```
array['len' + 'gth']
// == array.length
```

2. Execution order between **property reads/writes** is *preserved* **for** *correct side effects*

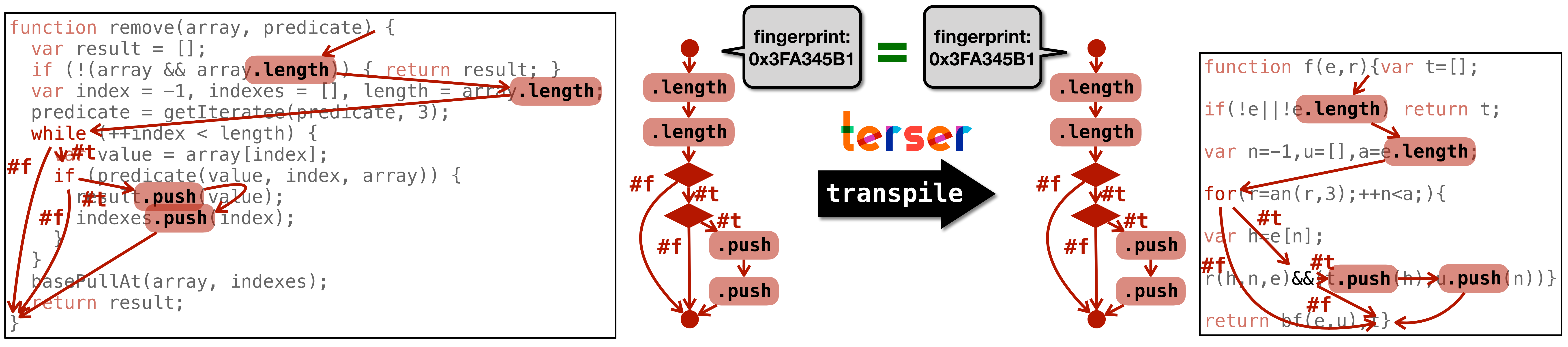
```
obj = {
  get p() { console.log(1); }
  set q() { console.log(2); }
}
```

```
obj.p;
// print 1
obj.q = 42;
// print 2
```

≠

```
obj.q = 42;
// print 2
obj.p;
// print 1
```

- **Property-Order Graph (POG)** is a directed graph that represents **1)** which **property operations** on **2)** which **property names** are executed in **3)** which **order** in a function body

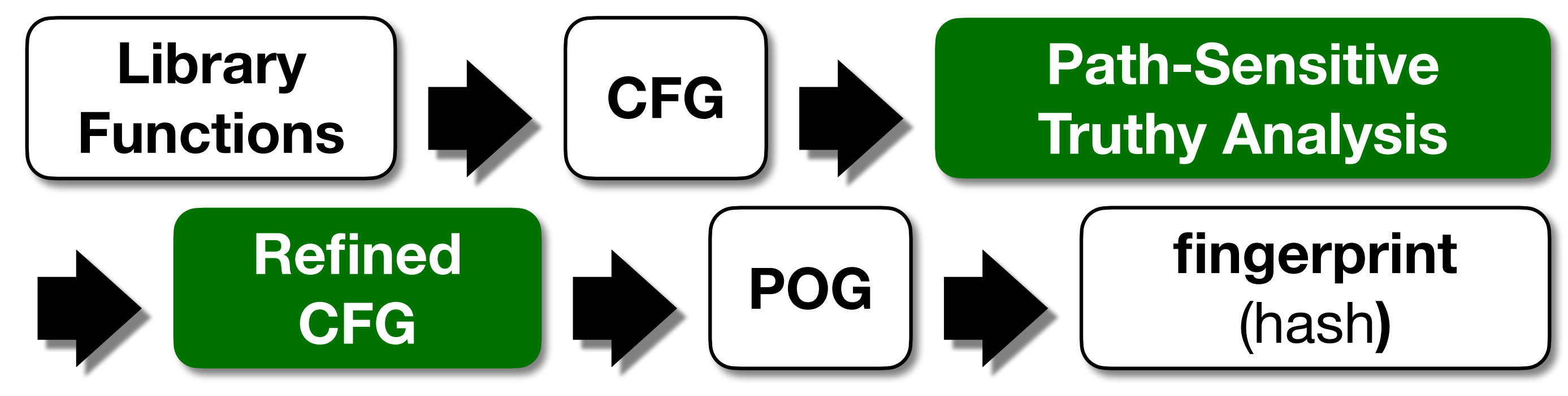


```
function remove(array, predicate) {
  var result = [];
  if (!(array && array.length)) { return result; }
  var index = -1, indexes = [], length = array.length;
  predicate = getIteratee(predicate, 3);
  while (++index < length) {
    var value = array[index];
    if (predicate(value, index, array)) {
      result.push(value);
      indexes.push(index);
    }
  }
  basePullAt(array, indexes);
  return result;
}
```

```
function f(e,r){var t=[];
if(!e||!e.length) return t;
var n=-1,u=[],a=e.length;
for(r=an(r,3);++n<a;){
var h=e[n];
r(h,n,e)&&(t.push(h),u.push(n))}
return bf(e,u),t}
```

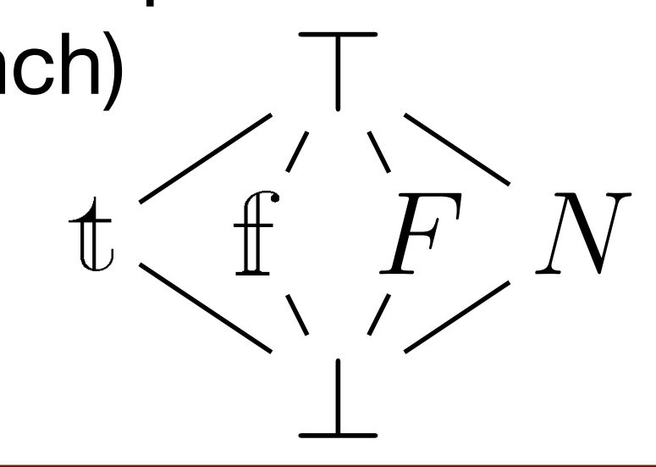
"remove" function in **Lodash.js v4.17.21** **POG of original code** **POG of transpiled code** **Transpiled code**

3. Path-Sensitive Truthy Analysis



Track **truthiness** of variables along execution paths!
(**Path** = Control flow from each branch)

Truthiness Domain	$\mathbb{t}$: Truthy	F : Flipped
	$\mathbb{f}$: Falsy	N : Non-Flipped

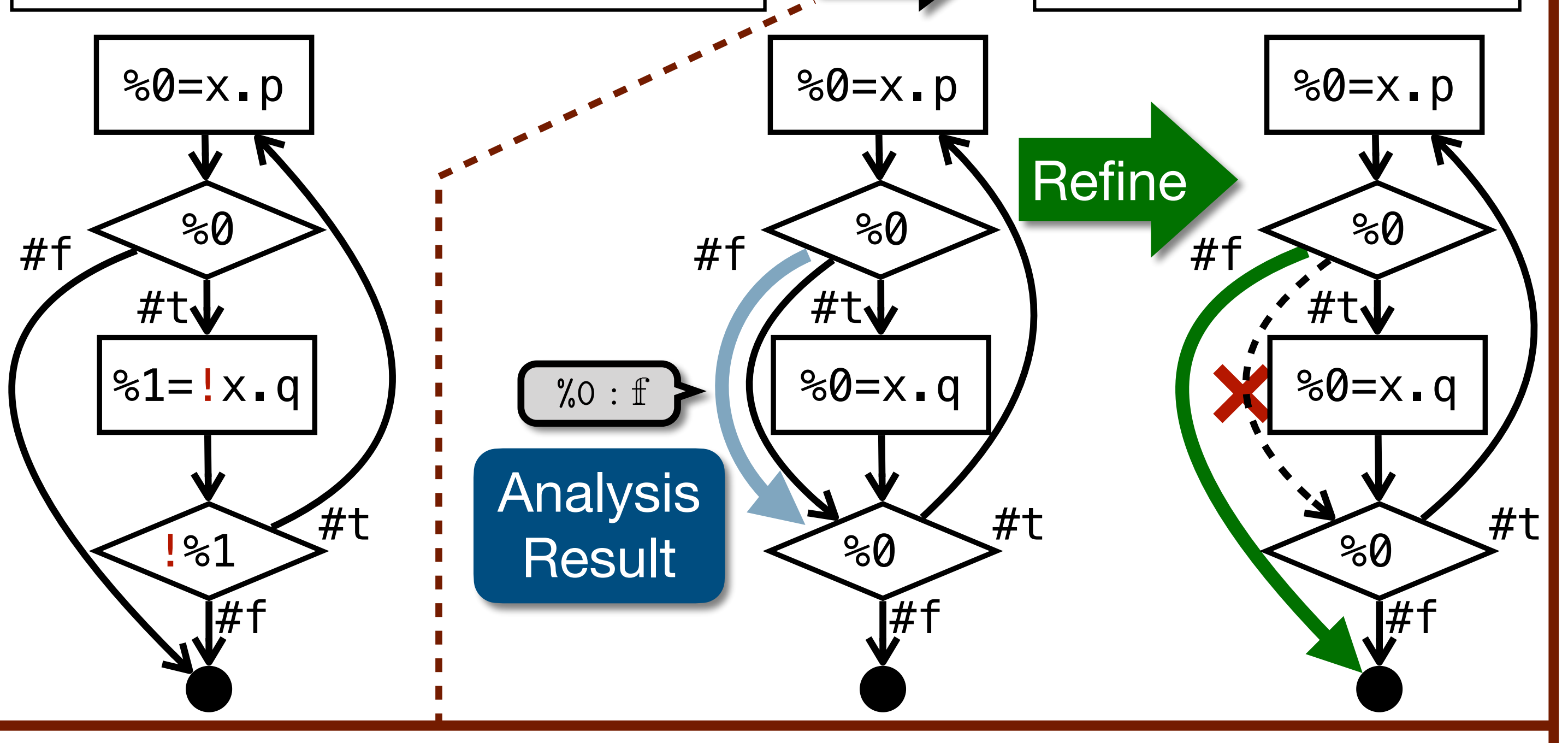


1. Branch Flipping - consistently flip branches

`while(x.p) if(!x.q) break;` → terser → `for(;x.p&&x.q;);`

2. Branch Bypassing - bypath always truthy/falsy branch conditions

`while(x.p) if(!x.q) break;` → terser → `for(;x.p&&x.q;);`

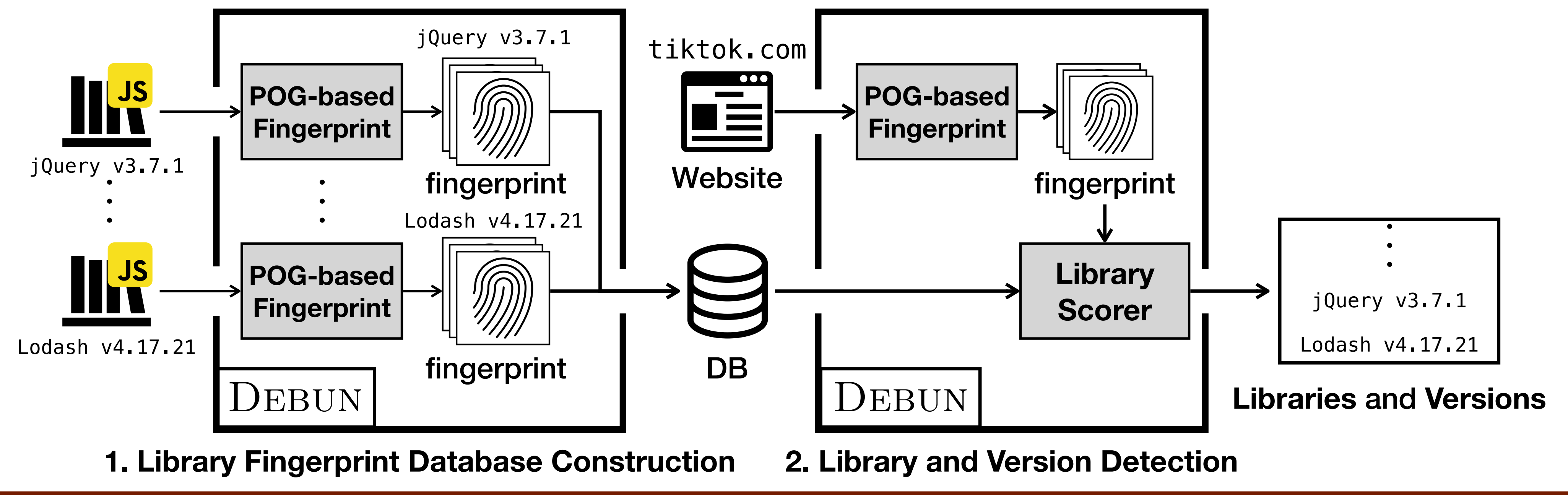


3. Path Cloning - clone merged paths

`while(x.p) if(!x.q) break;` → terser → `for(;x.p&&x.q;);`

4. Overall Structure

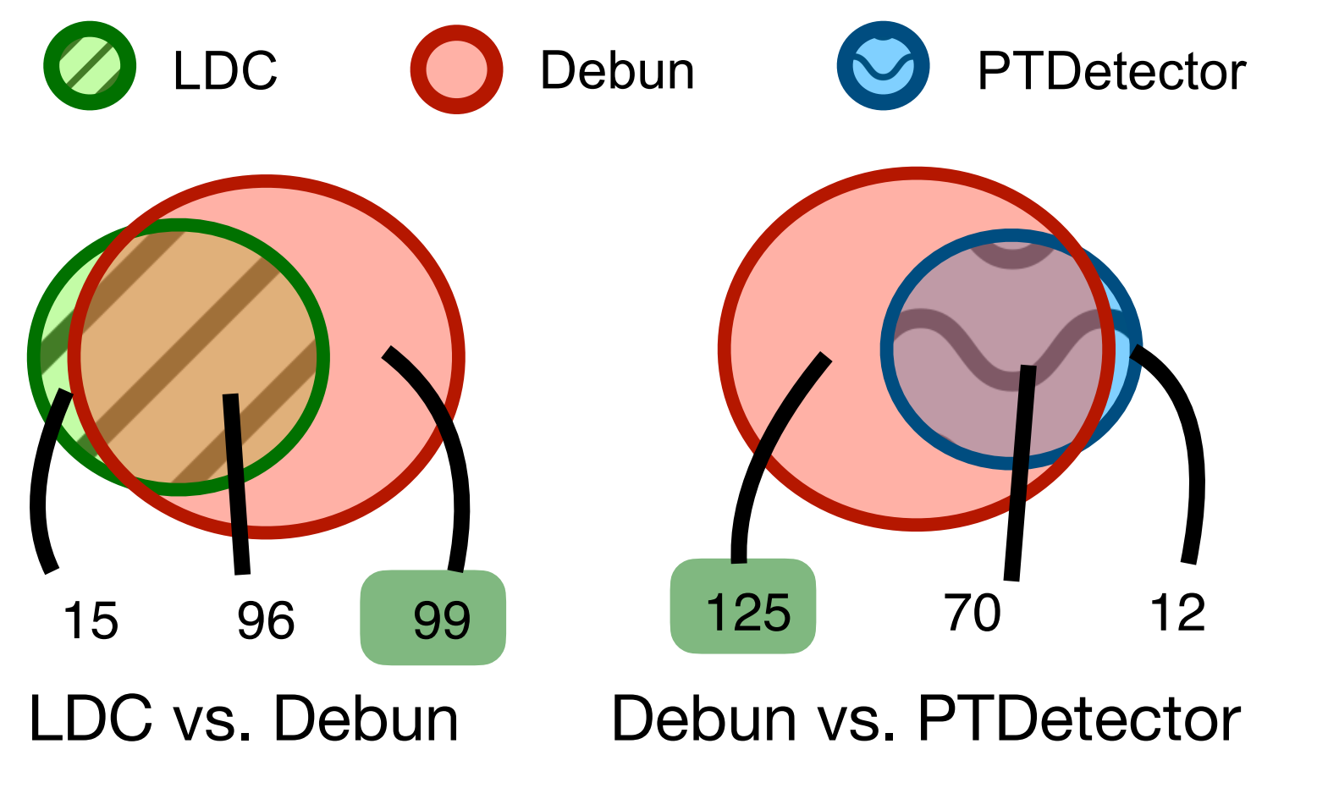
Debun - A POG-based Library Detector



5. Evaluation

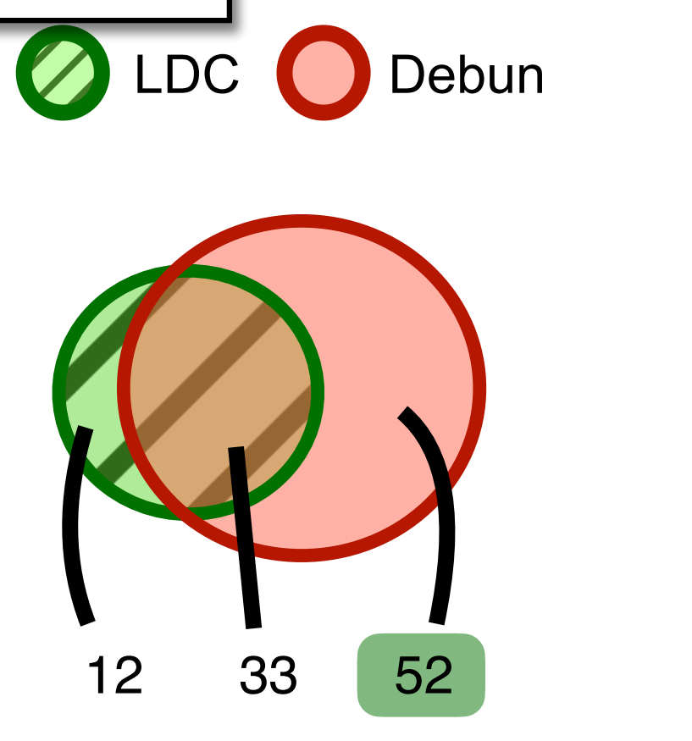
RQ1. Library Detection

Metric	LDC	PTDetector	Debun
TP	111	82	195
FP	3	9	7
FN	112	141	28
Precision	97.37%	90.11%	96.53%
Recall	49.78%	36.77%	87.44%
F1-score	65.88%	52.23%	91.76%



RQ2. Library Version Detection

Metric	LDC	Debun
TP	45	85
FP	0	16
FN	60	20
Precision	100.00%	84.16%
Recall	42.86%	80.95%
F1 score	60.00%	82.52%



RQ3. Ablation Study - Analysis-based Refinement

F: Branch Flipping / **B**: Branch Bypassing / **C**: Path Cloning
Count: count property names w/o execution order

Metric	Count	POG	POG+F	POG+FB	POG+FBC
# Consistent	47,385	35,370	43,358	45,404	45,522
# Functions	54,368	54,368	54,368	54,368	54,368
Consistency	87.16%	65.06%	79.75%	83.51%	83.73%
# Functions	55,518	55,518	55,518	55,518	55,518
# Duplicated	171,5034	274,252	273,252	273,678	273,684
Accuracy	3.28%	20.24%	20.32%	20.29%	20.29%

Email: ryan040@korea.ac.kr