Learning a Static Analyzer from Data



Pavol Bielik Veselin Raychev Martin Vechev

Department of Computer Science
ETH Zurich

CAV 2017
July 22-28, Heidelberg

Writing a Static Analyzer

DOOP

Framework for Java Pointer Analysis

17 contributors

TAJS

Static Type Checker for JavaScript

Static Type Checker for JavaScript

~400 contributors

Writing static analyzer is

Writing static analyzer is **hard**
Writing static analyzer is **frustrating**
Writing static analyzer is **time consuming**
Writing static analyzer is **brittle**

Learn more

Example of Unsound Analysis

flow

```
1  /* @flow */
2
3  var data = [7, 2, 9];
4
5  function collect(val, idx, obj) {
6    if (val > this.threshold) {
7
8    }
9  }
10
11 data.threshold;
12 data.forEach(collect);
```

Missed Error (line 6: `this.threshold`)

Error correctly reported (line 11: `data.threshold;`)

Errors | JSON | AST | v0.51.0 ▼

```
11: data.threshold;
         ^ property `threshold`. Property not found in
11: data.threshold;
    ^ Array
```

Can we learn a static analyzer?

(aka its abstract transformers)

This Work: Learn Static Analyzer from Data

Input Dataset

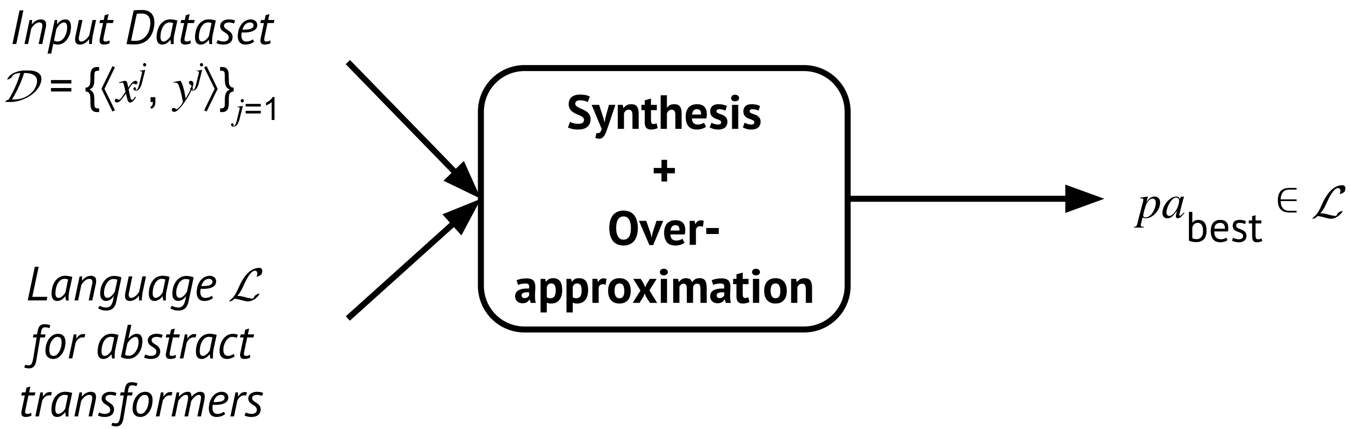
$\mathcal{D} = \{\langle x^j, y^j \rangle\}_{j=1}$

This Work: Learn Static Analyzer from Data

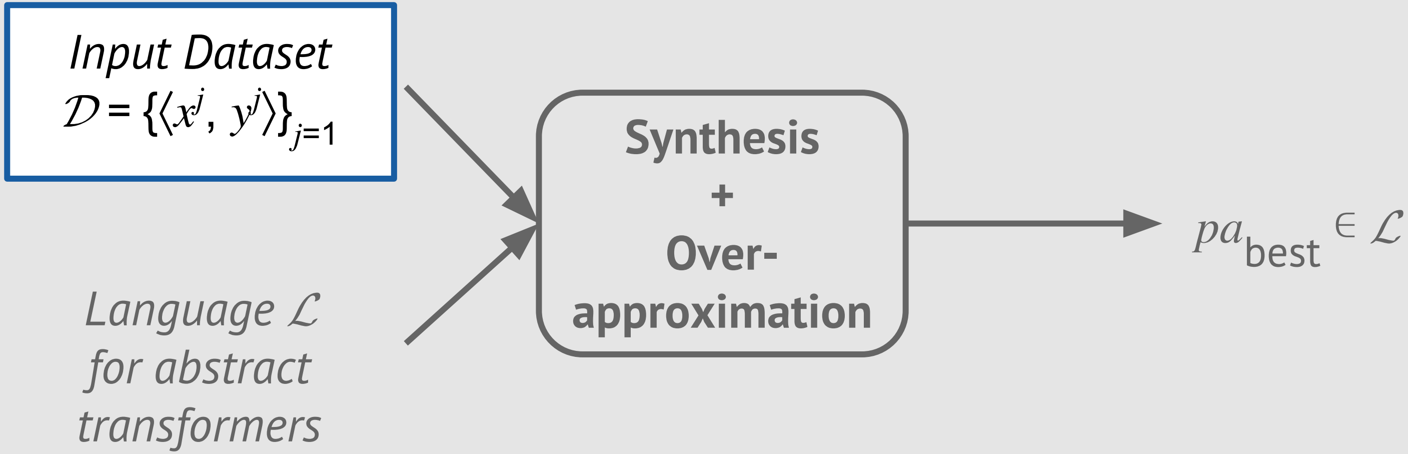
Input Dataset
$\mathcal{D} = \{\langle x^j, y^j \rangle\}_{j=1}$

Language $\mathcal{L}$
for abstract transformers

This Work: Learn Static Analyzer from Data

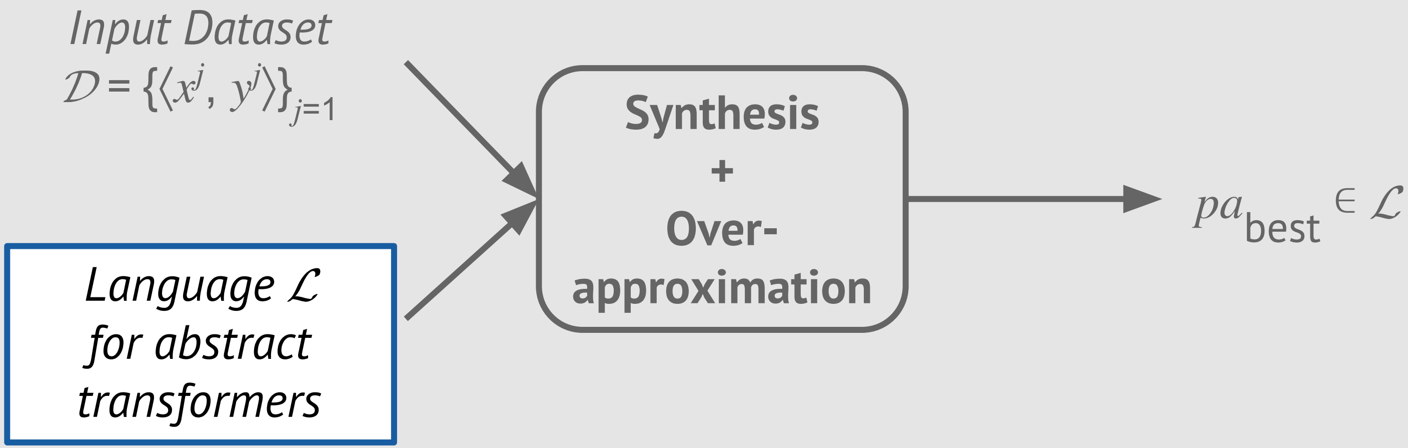


This Work: Learn Static Analyzer from Data



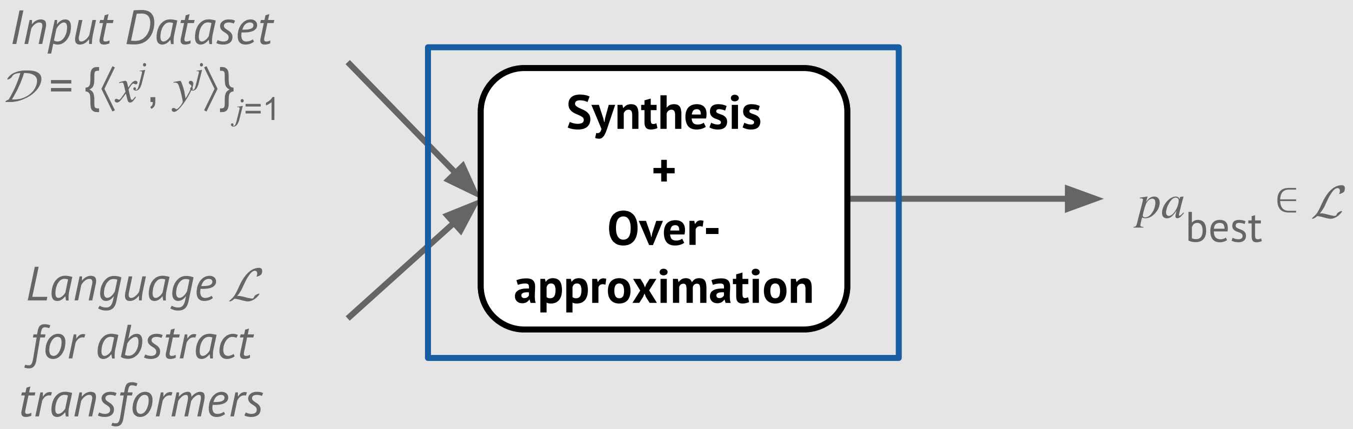
How to obtain suitable dataset?

This Work: Learn Static Analyzer from Data



What is the language over which to learn?
How to allow generating new interesting transformers?

This Work: Learn Static Analyzer from Data



How to design scalable learning over large search spaces?
How to prevent overfitting?

Can we learn a static analyzer?

interpretable and sound

↓

Can we learn a static analyzer?

Problem Formulation

$$pa_{\text{best}} = \underset{pa \in \mathcal{L}}{\text{arg min}}\ cost(\mathcal{D}, pa)$$ ← analysis precision

analysis soundness → $$\text{st. } \forall \langle x, y \rangle \in \mathcal{D} \,.\, \alpha(y) \sqsubseteq pa(x)$$

An Example Transformer Learned

```
Array.prototype.filter ::=
  if caller has one argument then
    points-to global object
  else if 2nd argument is Identifier then
    if 2nd argument is undefined then
      points-to global object
    else
      points-to 2nd argument
  else if 2nd argument is this then
      points-to 2nd argument
  else if 2nd argument is null then
      points-to global object
  else //2nd argument is a primitive value
      points-to new allocation site
```

An Example Transformer Learned

```
Array.prototype.filter ::=
  if caller has one argument then
    points-to global object
  else if 2nd argument is Identifier then
    if 2nd argument is undefined then
      points-to global object
    else
      points-to 2nd argument
  else if 2nd argument is this then
      points-to 2nd argument
  else if 2nd argument is null then
      points-to global object
  else //2nd argument is a primitive value
      points-to new allocation site
```

An Example Transformer Learned

```
Array.prototype.filter ::=
  if caller has one argument then
    points-to global object
  else if 2nd argument is Identifier then
    if 2nd argument is undefined then
      points-to global object
    else
      points-to 2nd argument
  else if 2nd argument is this then
      points-to 2nd argument
  else if 2nd argument is null then
      points-to global object
  else //2nd argument is a primitive value
      points-to new allocation site
```

Let us show the learning on an example analysis (aka points-to analysis)

Dataset: Points-to Analysis

Program

```
function collect(value, idx, obj) {
  if (value >= this.threshold) {
    ...
  }
  ...
}
```

	Abstract Syntax Tree (AST)	***execution reads/writes***
l_1	IfStatement	-
l_2	BinaryExpression	-
l_3	Identifier:value	o_1
l_4	MemberExpression	-
l_5	ThisExpression	o_2
l_6	Property:threshold	o_3

Dataset: Points-to Analysis

Program

```
function collect(value, idx, obj) {
  if (value >= this.threshold) {
    ...
  }
  ...
}
```

	Abstract Syntax Tree (AST)	***execution reads/writes***
l_1	IfStatement	-
l_2	BinaryExpression	-
l_3	Identifier:value	o_1
l_4	MemberExpression	-
l_5	ThisExpression	o_2
l_6	Property:threshold	o_3

$$\langle (AST, l_5), o_2 \rangle$$

$$\mathcal{D} = \{\langle x^j, y^j \rangle\}_{j=1}$$

Language Describing Abstract Transformers

$$l \in \mathcal{L} \coloneqq a \mid \textbf{if } g \textbf{ then } l \textbf{ else } l$$

$a \in$ *Actions* $g \in$ *Guards*

```
function collect(val, idx, obj) {
  if (val >= this.threshold) { ... }
}

var dat = [5, 3, 9];
dat.filter( collect, ctx );
```

Points-to Query

a_1

method name is filter

g_1

has 2nd argument

g_2

Language Describing Abstract Transformers

$$l \in \mathcal{L} \coloneqq a \mid \textbf{if } g \textbf{ then } l \textbf{ else } l$$

$a \in \textit{Actions}$ $g \in \textit{Guards}$

```
function collect(val, idx, obj) {
  if (val >= this.threshold) { ... }
}

var dat = [5, 3, 9];
dat.filter( collect, ctx );
```

Points-to Query

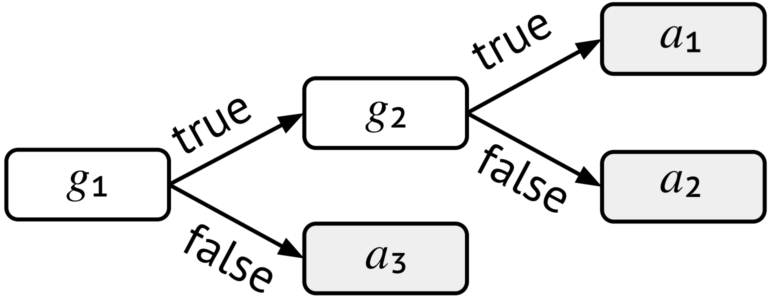
a_1

method name is filter

g_1

has 2nd argument

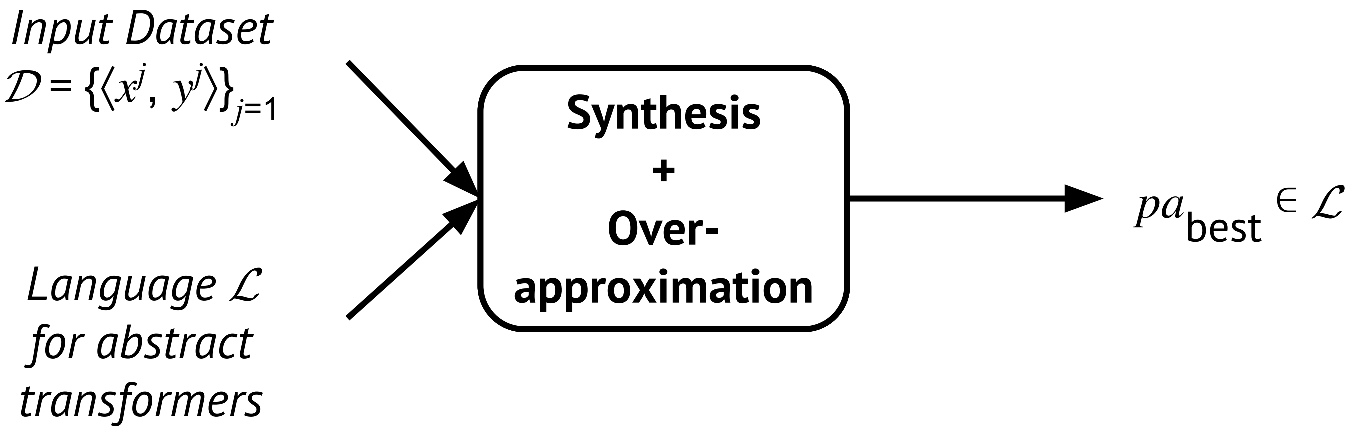
g_2



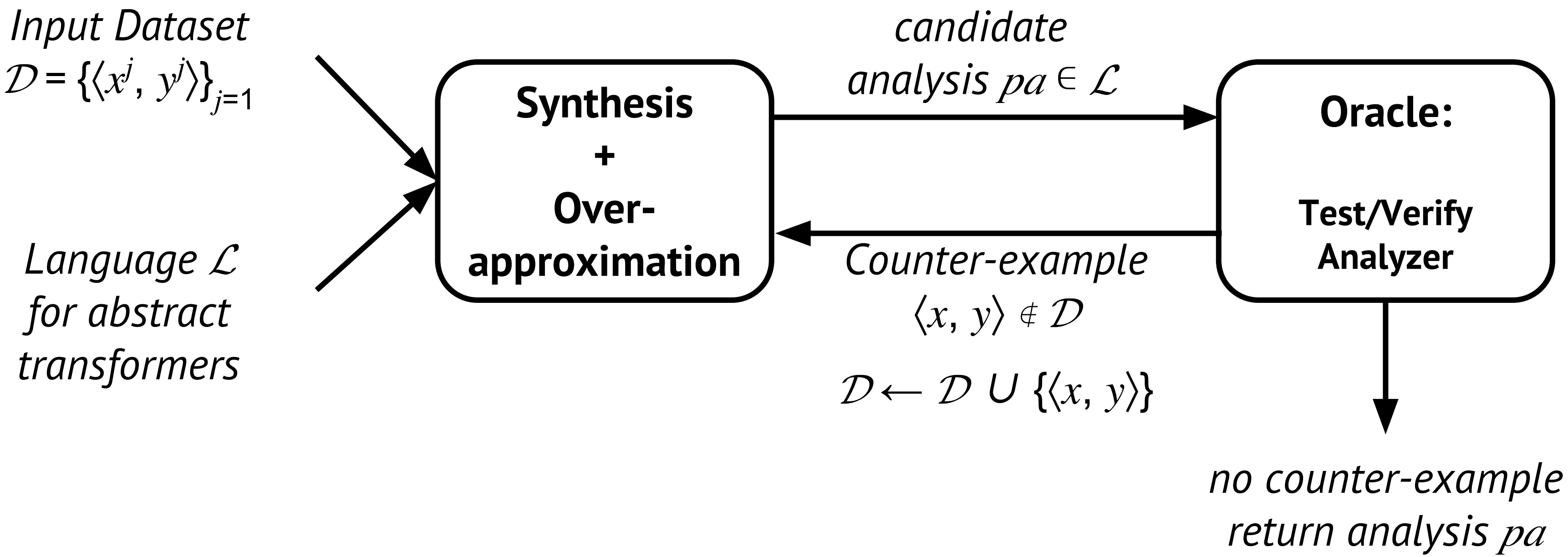
can be represented as decision tree

paths interpreted as abstract transformers

Learning: Decision Trees



Learning: Decision Trees + CEGIS



Learning: Problem Formulation

Problem Formulation

$$pa_{\text{best}} = \arg\min_{pa \in \mathcal{L}} cost(\mathcal{D}, pa)$$

$$\text{st. } \forall \langle x, y \rangle \in \mathcal{D} \,.\, \alpha(y) \sqsubseteq pa(x)$$

guarantees analysis soundness

Cost Function

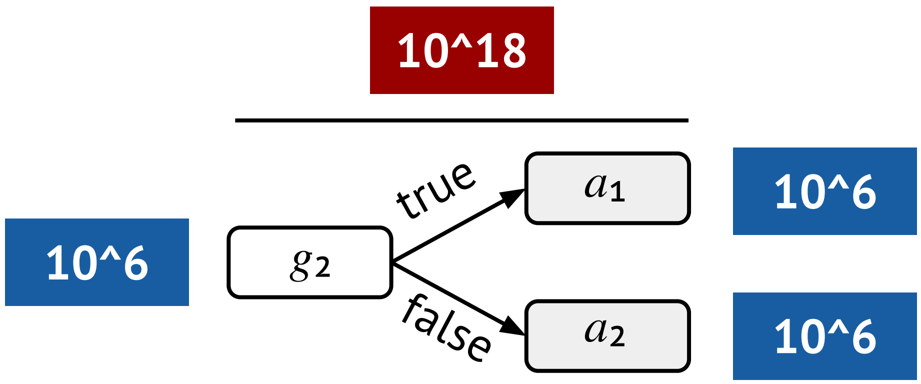
$$r(x, y, pa) = \textbf{if } (y \neq pa(x)) \textbf{ then } 1 \textbf{ else } 0$$

$$cost(\mathcal{D}, pa) = \sum_{\langle x, y \rangle \in \mathcal{D}} r(x, y, pa)$$

prefer analysis with fewer errors

Learning Algorithm

$$l \in \mathcal{L} \coloneqq a \mid \textbf{if } g \textbf{ then } l \textbf{ else } l$$

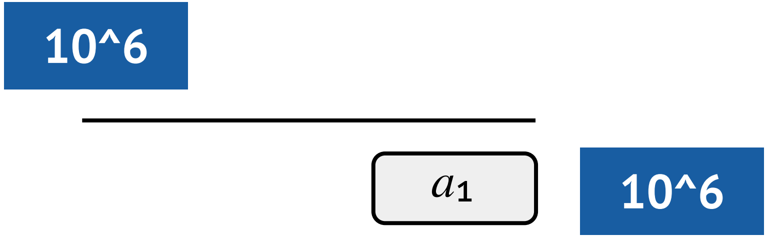


Untractable

Learning Algorithm

$$l \in \mathcal{L} \coloneqq a \mid \textbf{if } g \textbf{ then } l \textbf{ else } l$$

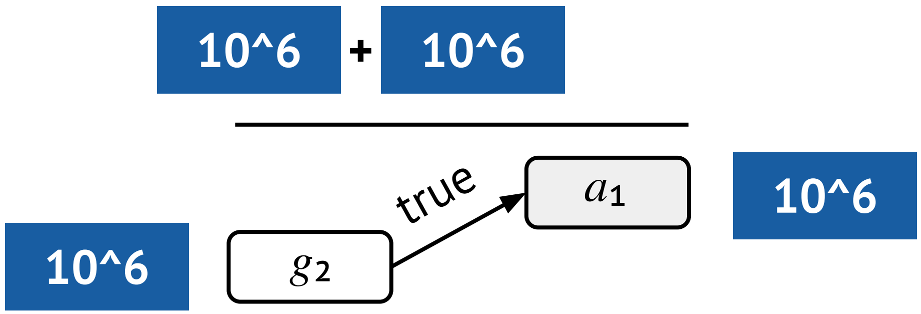
Key Idea: Synthesise Programs in Parts



Learning Algorithm

$$l \in \mathcal{L} \coloneqq a \mid \textbf{if } g \textbf{ then } l \textbf{ else } l$$

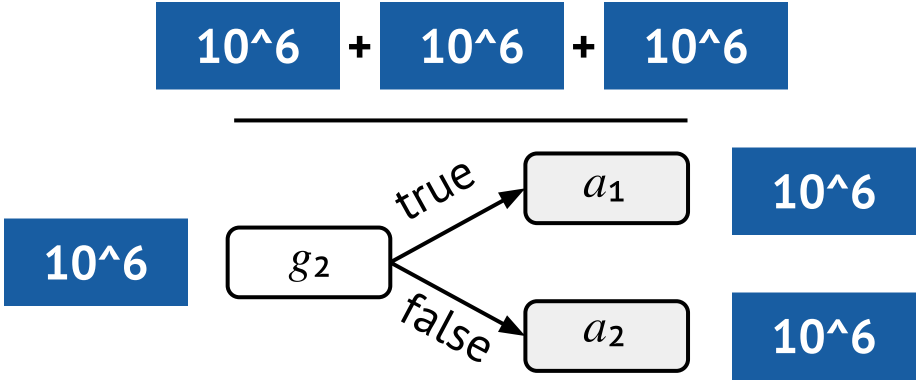
Key Idea: Synthesise Programs in Parts



Learning Algorithm

$$l \in \mathcal{L} \coloneqq a \mid \textbf{if } g \textbf{ then } l \textbf{ else } l$$

Key Idea: Synthesise Programs in Parts



Learning Algorithm

$$a_{\text{best}} = \arg\min_{a \in Actions} cost(\mathcal{D}, a)$$

$cost(\mathcal{D}, a_{\text{best}}) > 0$

a_{best}

refine analysis

$\mathcal{D}$

$cost(\mathcal{D}, a_{\text{best}}) = 0$

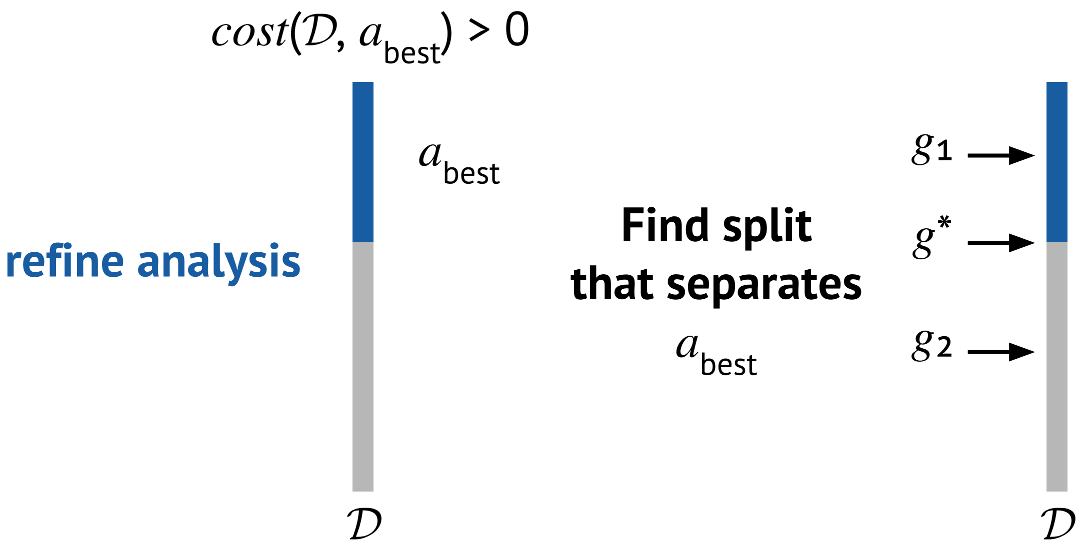
a_{best}

no errors
return a_{best}

$\mathcal{D}$

Learning Algorithm

$$g_{\text{best}} = \underset{g \,\in\, Guards}{\arg\max}\; InfGain(\mathcal{D}, g, a_{\text{best}})$$

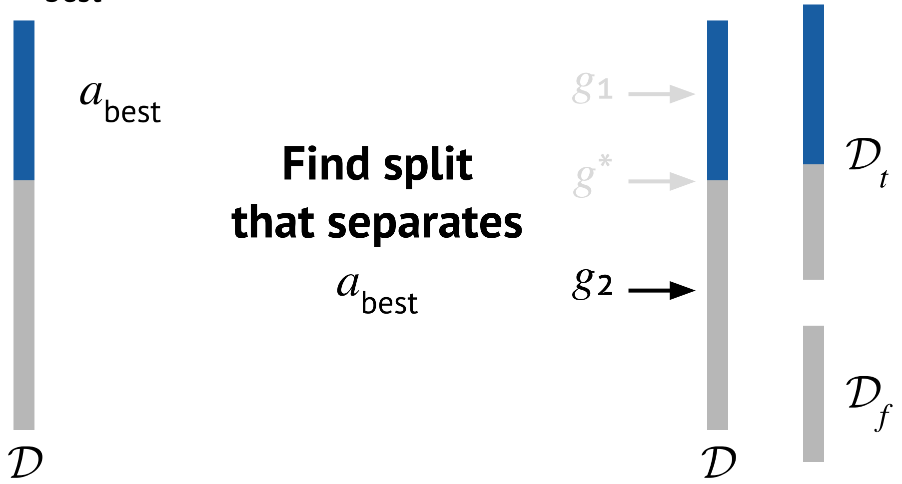


Learning Algorithm

$$g_{\text{best}} = \underset{g \in \textit{Guards}}{\text{arg max}}\ \textit{InfGain}(\mathcal{D}, g, a_{\text{best}})$$

$cost(\mathcal{D}, a_{\text{best}}) > 0$

refine analysis

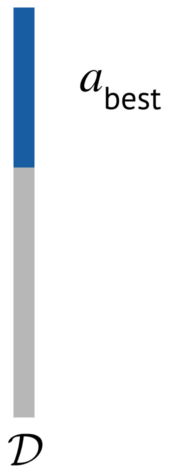


Learning Algorithm

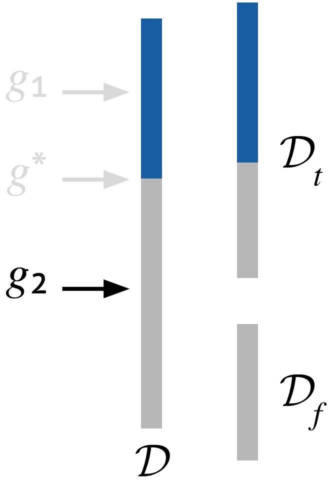
$$g_{\text{best}} = \arg\max_{g \in Guards} InfGain(\mathcal{D}, g, a_{\text{best}})$$

$cost(\mathcal{D}, a_{\text{best}}) > 0$

refine analysis



Find split that separates a_{best}



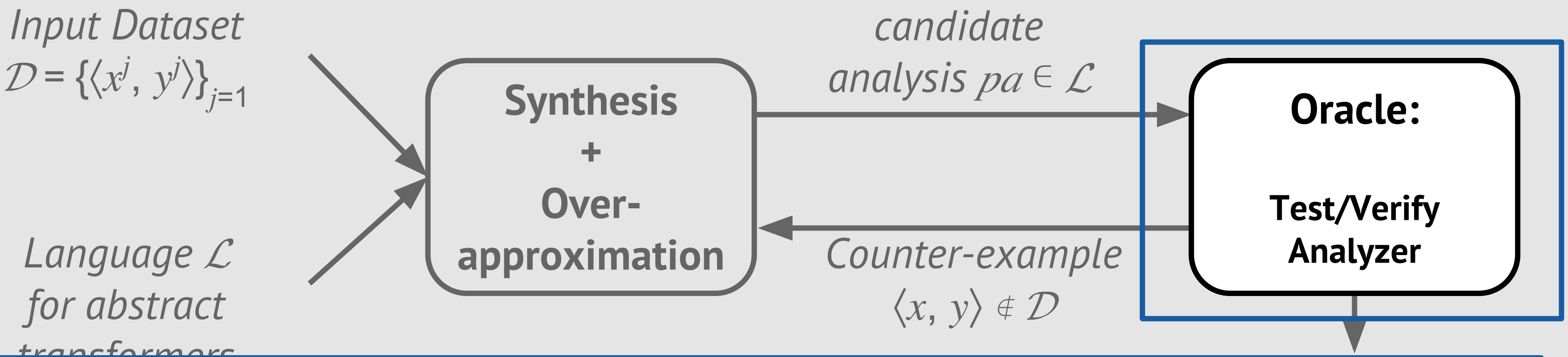
$InfGain(\mathcal{D}, g, a_{\text{best}}) = 0$

no split reduces entropy

↓

$approximate(\mathcal{D})$

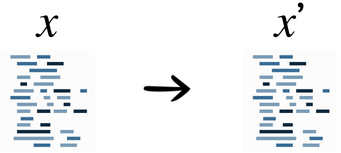
Learning: Decision Trees + CEGIS



How to find complex counter-examples quickly?
How to efficiently explore hard to find corner cases?

Naive Approach: Random Fuzzing

1. Pick a random training example — $\langle x, y \rangle \in \mathcal{D}$
2. Mutate the input randomly — $x \rightarrow x'$



3. Obtain the correct label — Execute x' $\rightarrow$ $\mathcal{D}'$
4. Check for correctness — $\forall \langle x, y \rangle \in \mathcal{D}' \,.\, \alpha(y) \sqsubseteq pa(x)$
5. Repeat

Naive Approach: Random Fuzzing

1. Pick a random training example
2. Mutate the input randomly
3. Obtain the correct label
4. Check for correctness
5. Repeat

Exponential Number of Choices

Slow

When to stop?

The Oracle: Testing an Analyzer

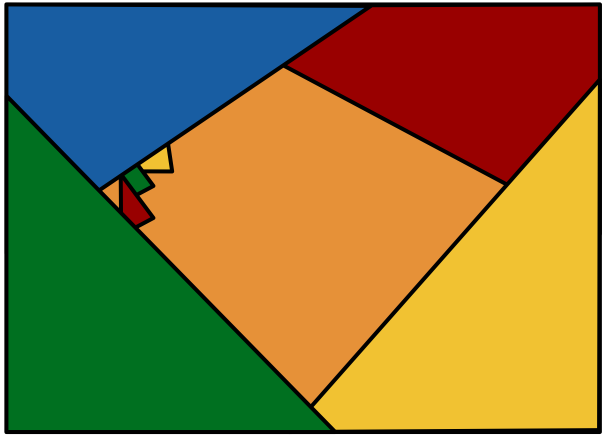
Key Idea: Take advantage of candidate analysis pa

$\mathcal{T}$

How to sample from space of all programs?

The Oracle: Testing an Analyzer

execution path coverage of pa

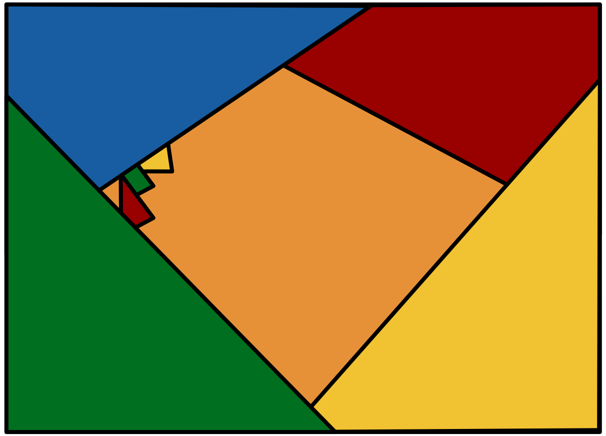


$\mathcal{T}$

The Oracle: Testing an Analyzer

execution path coverage of pa

mutate only parts that affect pa



$\mathcal{T}$

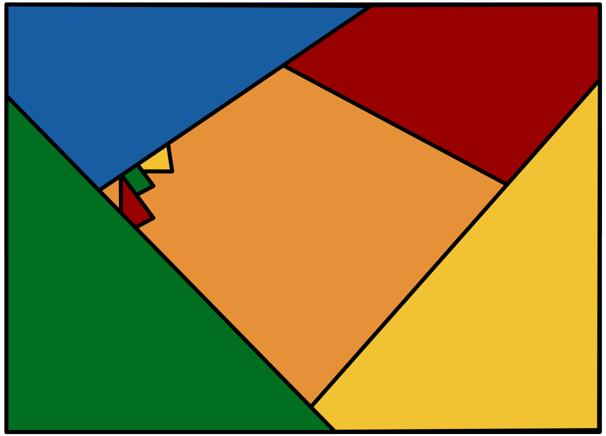
```
fnc collect(val, idx, obj) {
  if (val >= this.threshold){
    ...
  }
}

var dat = [5, 3, 9];
dat.filter(collect, ctx);
```

Query

Locations accessed by the analysis

The Oracle: Testing an Analyzer

execution path coverage of pa



$\mathcal{T}$

mutate only parts that affect pa

```
fnc collect(val, idx, obj) {
  if (val >= this.threshold){
    ...
  }
}
var dat = [5, 3, 9];
dat.filter(collect, ctx);
```

Query

Locations accessed by the analysis

select relevant program mutations

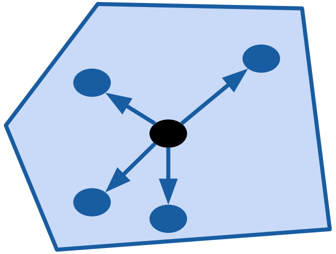
Modification via Equivalence Modulo Abstraction (EMA)

Modification via Global Jumps

The Oracle: Testing an Analyzer

Modifications via Equivalence Modulo Abstraction (EMA)

Semantic preserving mutations



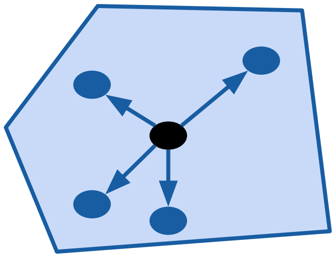
Adding dead code
Renaming variables
Renaming user defined functions
Side-effect free expressions

$\mathcal{T}$

The Oracle: Testing an Analyzer

Modifications via Equivalence Modulo Abstraction (EMA)

Semantic preserving mutations



Adding dead code
Renaming variables
Renaming user defined functions
Side-effect free expressions

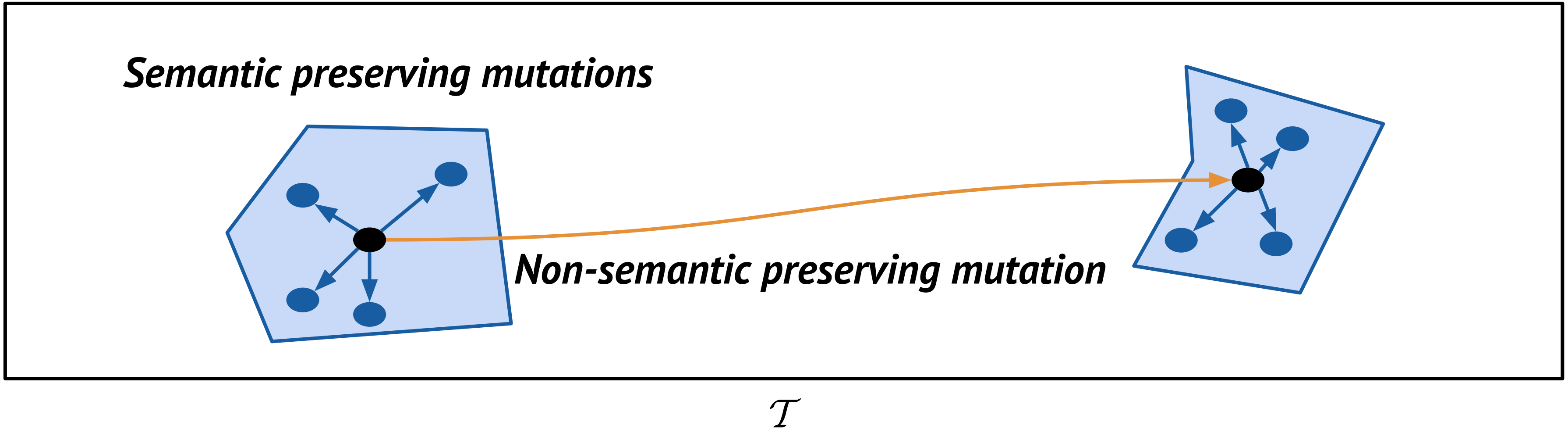
labels y can be reused

$\mathcal{T}$

The Oracle: Testing an Analyzer

Modifications via Equivalence Modulo Abstraction (EMA)

Modifications via Global Jumps



Evaluation

ECMAScript (ECMA-262) Conformance Suite

15 675

Programs

Points-to Analysis

```
function collect(val, idx, obj) {
  if (val >= this.threshold) { ... }
}

var dat = [5, 3, 9];
dat.filter( collect, ctx );
```

Allocation Site Analysis

```
var obj = {a: 7};
var arr = [1, 2, 3, 4];
if (arr.slice(0, 2) == ... )
var n = new Number(7);
var obj2 = new Object(obj);
try { ... } catch (err) { ... }
```

Approach Instantiation for Points-to Analysis

Input Dataset
$\mathcal{D} = \{\langle x^j, y^j \rangle\}_{j=1}$

$y \leftarrow$ **concrete object id**

Language $\mathcal{L}$ for abstract transformers

$a \in Actions$ ::= ε | `Move; a`
`Move`$_{\texttt{Core}}$ ::= `Up, Left, Right, DownFirst, DownLast, Top`

$g \in Guards$ ::= ε | `Move; g` | `Write; g`
`WriteOp` ::= `WriteValue, WriteType, WritePos, HasLeftSibling, HasRightSibling, HasChild`

Synthesis

$(\mathcal{H}, \sqsubseteq)$, $\boldsymbol{\alpha}$, $\boldsymbol{\gamma}$

Oracle

semantic preserving mutations
Adding dead code
Renaming variables
Renaming user defined functions
Side-effect free expressions

non-semantic preserving mutations
Add method arguments
Add method parameters
Change program constants

Learned Points-to Analysis

Function Name	*Dataset Size*	*Analysis Size*	*Counter-examples Found*
`Function.prototype`			
`call()`	26	97(18)	372
`apply()`	6	54(10)	182
`Array.prototype`			
`map()`	315	36(6)	64
`some()`	229	36(6)	82
`forEach()`	604	35(5)	177
`every()`	338	36(6)	31
`filter()`	408	38(6)	76
`find()`	53	36(6)	73
`findIndex()`	51	28(7)	96
`Array`			
`from()`	32	57(7)	160
`JSON`			
`stringify()`	18	9(2)	55

rules missed by Facebook Flow

Average Learning Time 14 minutes (4 min synthesis, 10 min oracle)

Learned Allocation Site Analysis

134 721
training dataset size

905
counter-examples found

99
refinement iterations

3 hours
Synthesis time

7 hours
time to find counter-examples

Overview of Learned Analysis

```
if HasPrevNodeValue then ⊤
elif WriteType == CallExpression then
  if Up WriteType == ExpressionStatement then
    ⊤       // return value not assigned
  else ...
elif WriteType == ArrayAccess then ...
elif WriteType == ObjectExp|ArrayExp|RegExp then
  NewAlloc // implicit constructors
elif WriteType == NewExpression then
  ...       // explicit constructor
elif Up WriteType == AssignmentExpression
  if left hand side of the assignment then NoAlloc
  ...
```

Learning a Static Analyzer from Data

Learns practical abstract transformers missed by existing state-of-the-art analyzers