SemSeed: A Learning-Based Approach for Creating Realistic Bugs

Jibesh Patra, Michael Pradel

August 25, 2021

SOLA SoftwareLab
University of Stuttgart
Germany

Why Seed Bugs?

Common wisdom suggests that bugs are harmful but why *seed* them?

Needs large amount of **realistic** bugs:

- Evaluate bug detectors
- Evaluate mutation testing
- Train learning-based bug detectors

What are the Current Options?

- Manually written bug benchmarks
  - Difficult to create a large dataset

What are the Current Options?

- Manually written bug benchmarks
  - Difficult to create a large dataset
- Heuristics based syntactic transformations
  - Unrealistic

What are the Current Options?

- Manually written bug benchmarks
  - Difficult to create a large dataset
- Heuristics based syntactic transformations
  - Unrealistic

Correct ✔

```
if(hasFailed && process.arch === 'x64')
```

What are the Current Options?

- Manually written bug benchmarks
  - Difficult to create a large dataset
- Heuristics based syntactic transformations
  - Unrealistic

Correct ✔

```
if(hasFailed && process.arch === 'x64')
```

Unrealistic Buggy 

```
if(hasFailed && process.arch !== 'first')
```

What are the Current Options?

- Manually written bug benchmarks
  - Difficult to create a large dataset
- Heuristics based syntactic transformations
  - Unrealistic

Correct ✔

```
if(hasFailed && process.arch === 'x64')
```

Unrealistic Buggy 

```
if(hasFailed && process.arch !== 'first')
```

Realistic Buggy 

```
if(hasFailed && process.arch !==  'x86' )
```

What are the Current Options?

- Manually written bug benchmarks
  - Difficult to create a large dataset
- Heuristics based syntactic transformations
  - Unrealistic

Correct ✔

```
if(hasFailed && process.arch === 'x64')
```

Unrealistic Buggy 

```
if(hasFailed && process.arch !== 'first')
```

Realistic Buggy 

```
if(hasFailed && process.arch !==  'x86' )
```

How to get such bugs?

SemSeed

- An approach for automatically seeding **realistic** bugs in a semantics-aware way.
- By **imitating** bugs occuring in the wild (top-100 GitHub projects).

Overview

Example Seeded Bug

3. Target Program

1. Bug Fix to Imitate

2. Bug Seeding Pattern

4. Bug Seeded Program

Example Seeded Bug

```
if (process.platform !== 'win32')
```

```
if (process.platform === 'darwin')
```

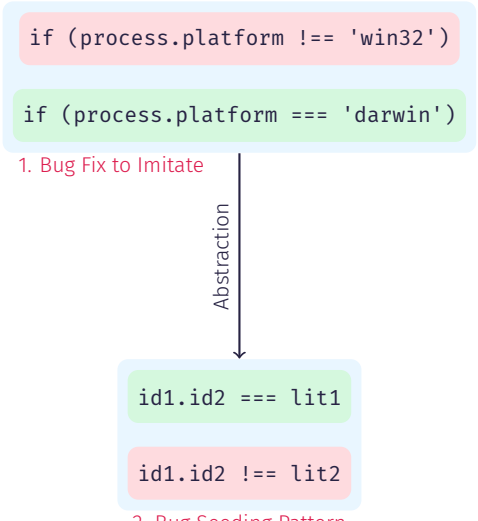
1. Bug Fix to Imitate

2. Bug Seeding Pattern

3. Target Program

4. Bug Seeded Program

Example Seeded Bug



3. Target Program

4. Bug Seeded Program

Example Seeded Bug

```
if (process.platform !== 'win32')
```

```
if (process.platform === 'darwin')
```

1. Bug Fix to Imitate


```
id1.id2 === lit1
```

```
id1.id2 !== lit2
```

2. Bug Seeding Pattern

```
...
hasFailed = item.errCode === -1;
if(hasFailed && process.arch === 'x64')
...
```

3. Target Program

4. Bug Seeded Program

Example Seeded Bug

```
if (process.platform !== 'win32')
```

```
if (process.platform === 'darwin')
```

1. Bug Fix to Imitate

Abstraction

```
id1.id2 === lit1
```

```
id1.id2 !== lit2
```

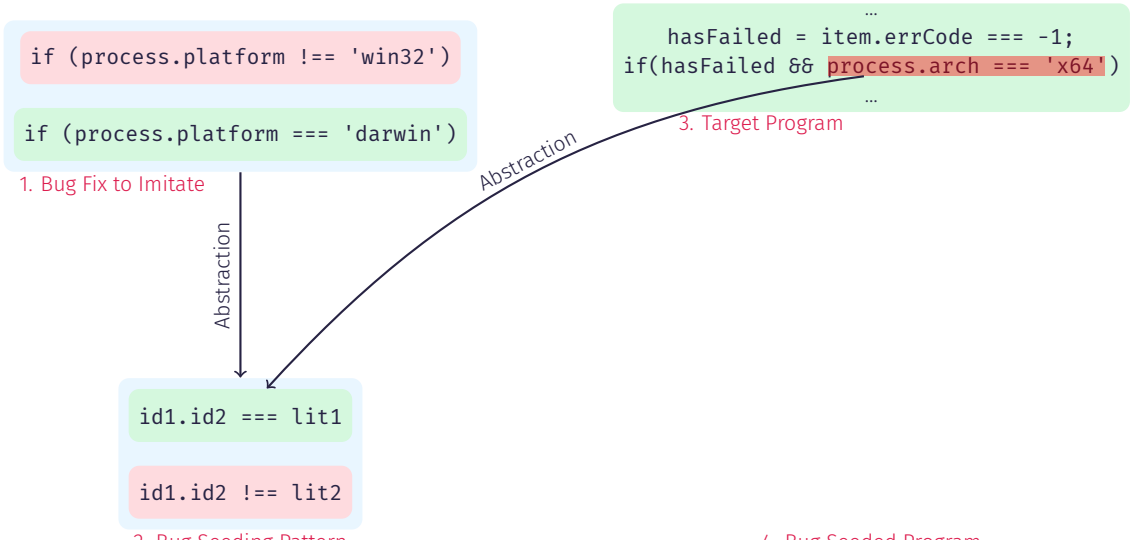
2. Bug Seeding Pattern

```
...
    hasFailed = item.errCode === -1;
if(hasFailed && process.arch === 'x64')
...
```

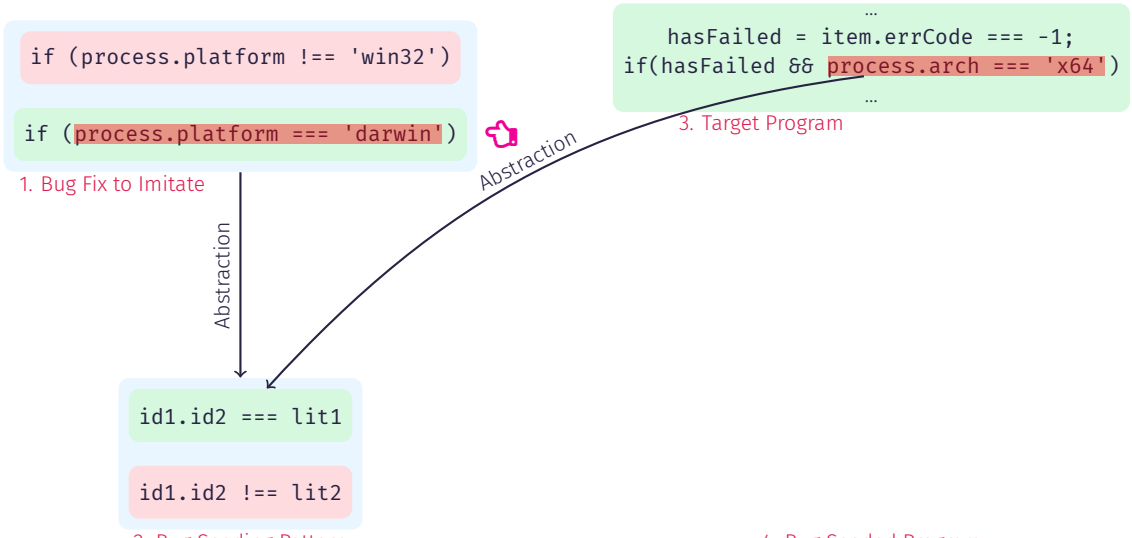
3. Target Program

4. Bug Seeded Program

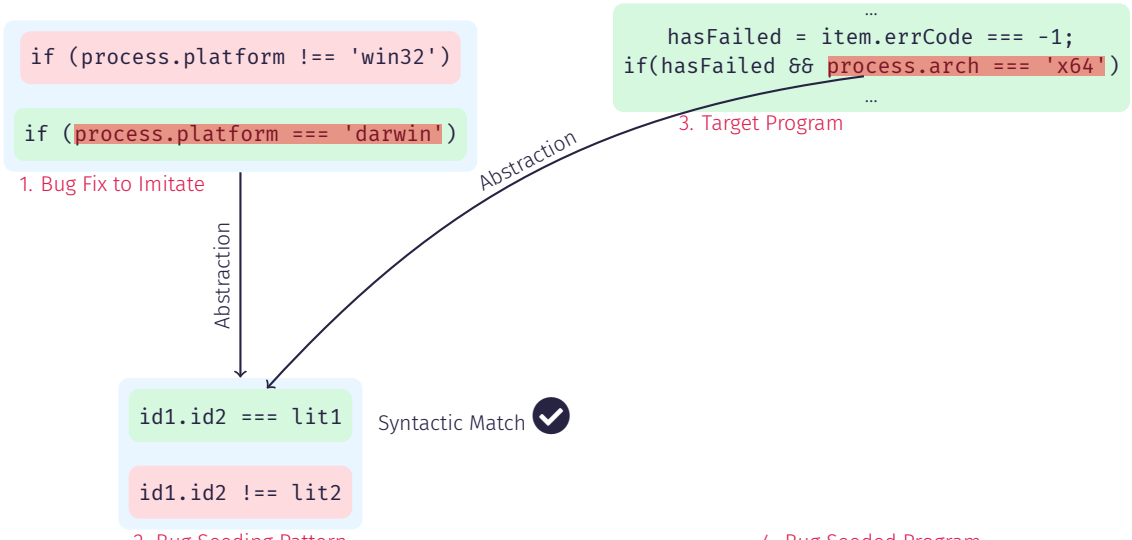
Example Seeded Bug



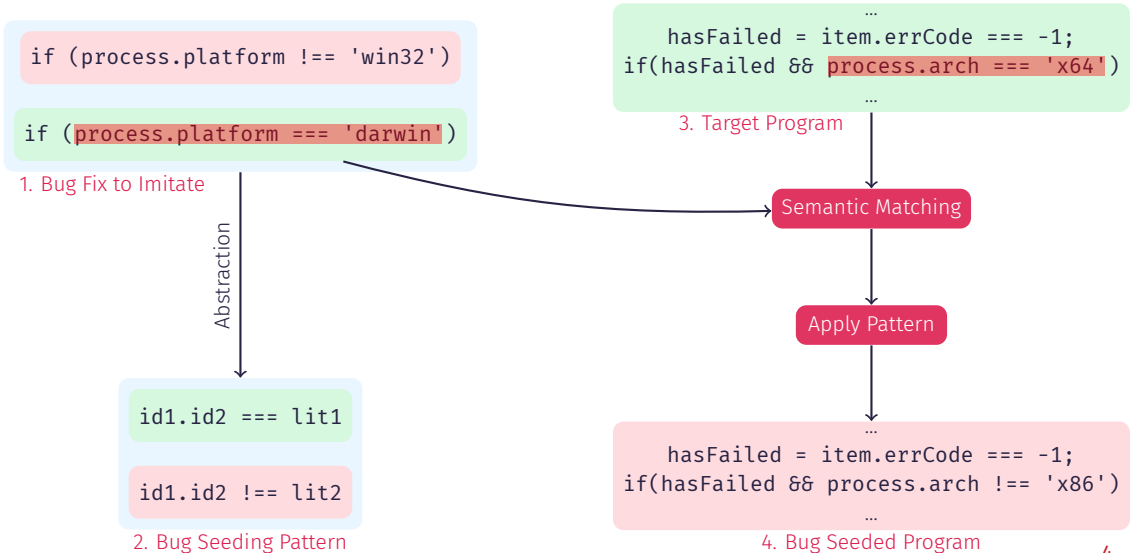
Example Seeded Bug



Example Seeded Bug



Example Seeded Bug



Semantic Matching

Goal: If a bug fix to imitate is *semantic match* of a target.

```
process . platform === 'darwin'
```

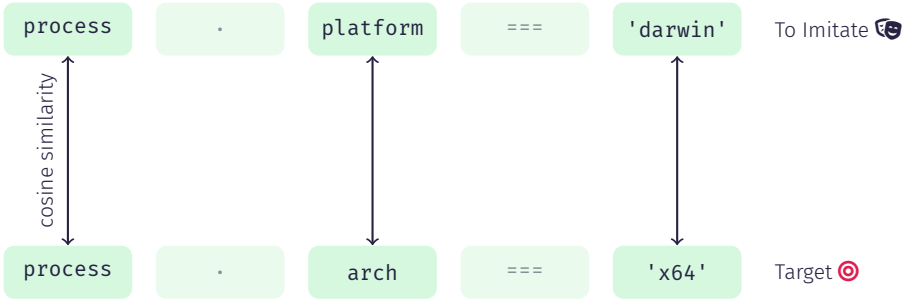
To Imitate

```
process . arch === 'x64'
```

Target

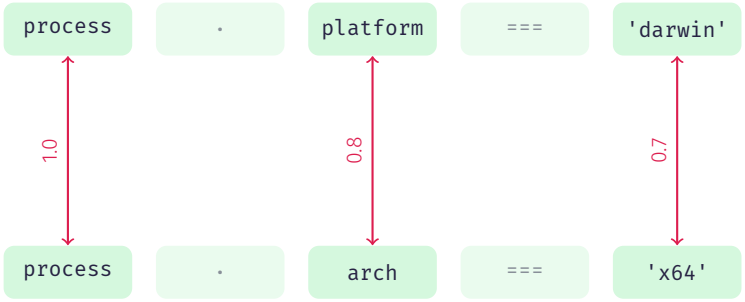
Semantic Matching

Goal: If a bug fix to imitate is *semantic match* of a target.



Semantic Matching

Goal: If a bug fix to imitate is *semantic match* of a target.



Apply Pattern

```
if (process.platform !== 'win32')

if (process.platform === 'darwin')
```

1. Bug Fix to imitate

```
...
hasFailed = item.errCode === -1;
if(hasFailed && process.arch === 'x64')
...
```

3. Target Program

Semantic Matching

Apply Pattern

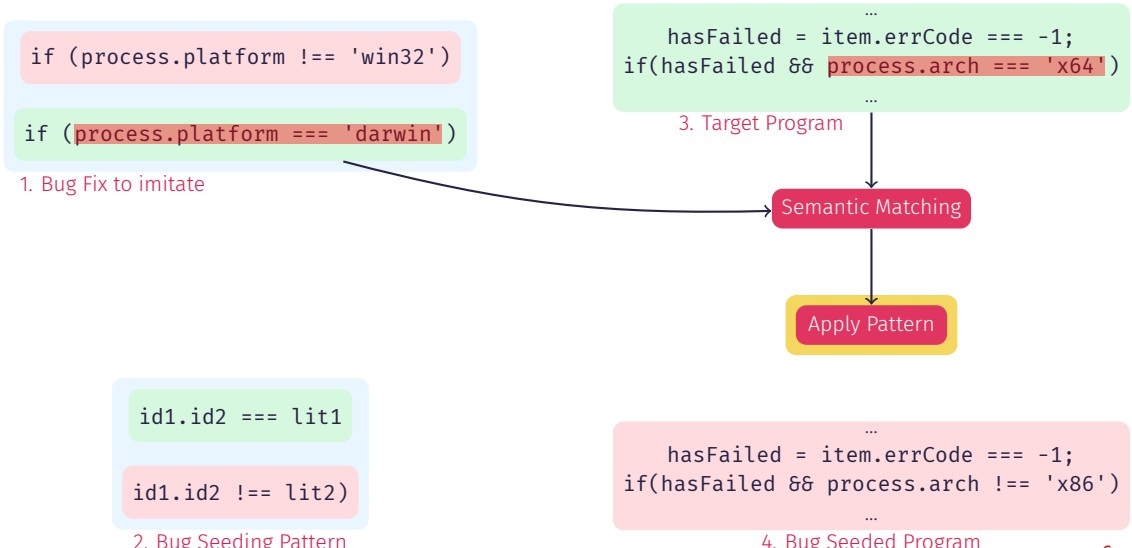
```
id1.id2 === lit1

id1.id2 !== lit2)
```

2. Bug Seeding Pattern

```
...
hasFailed = item.errCode === -1;
if(hasFailed && process.arch !== 'x86')
...
```

4. Bug Seeded Program

Apply Pattern



Apply Pattern

🎭 To Imitate (Correct)

```
process . platform === 'darwin'
```

🎯 Target (Correct)

🎭 To Imitate (Buggy)

🎯 Target (Buggy)

Apply Pattern

🎭 To Imitate (Correct)	`process`	`.`	`platform`	`===`	`'darwin'`
🎯 Target (Correct)	`process`	`.`	`arch`	`===`	`'x64'`

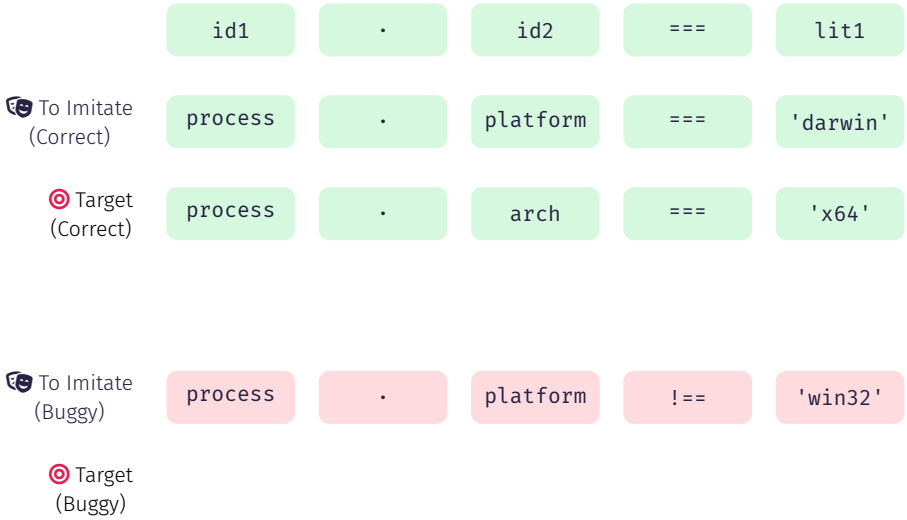
🎭 To Imitate (Buggy)

🎯 Target (Buggy)

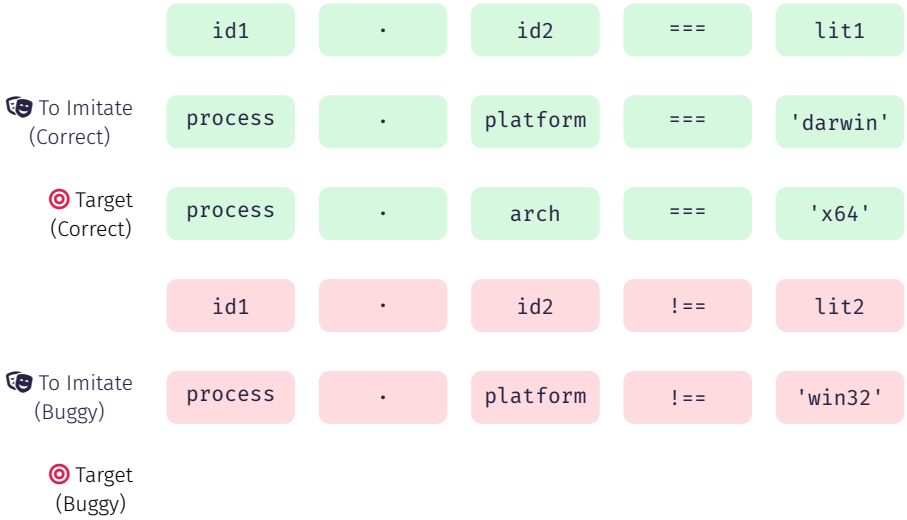
Apply Pattern

	`id1`	`.`	`id2`	`===`	`lit1`
🎭 To Imitate (Correct)	`process`	`.`	`platform`	`===`	`'darwin'`
🎯 Target (Correct)	`process`	`.`	`arch`	`===`	`'x64'`
🎭 To Imitate (Buggy)					
🎯 Target (Buggy)					

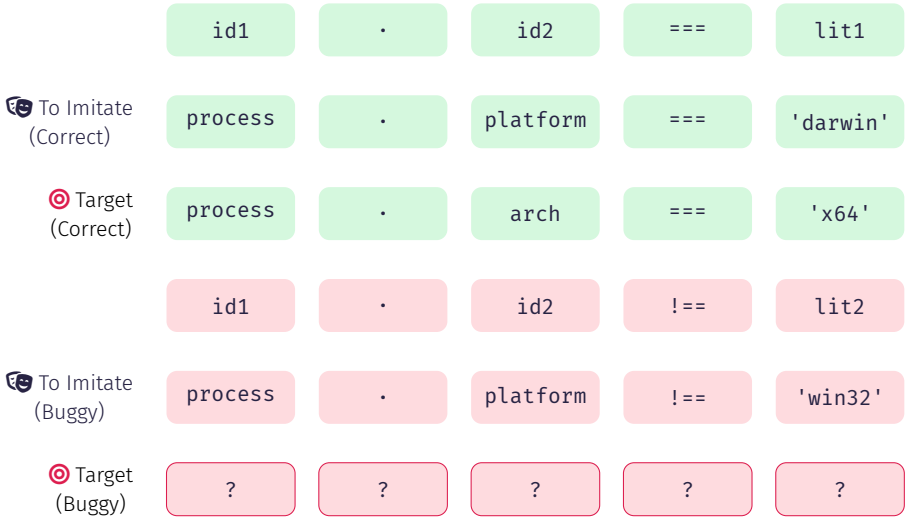
Apply Pattern



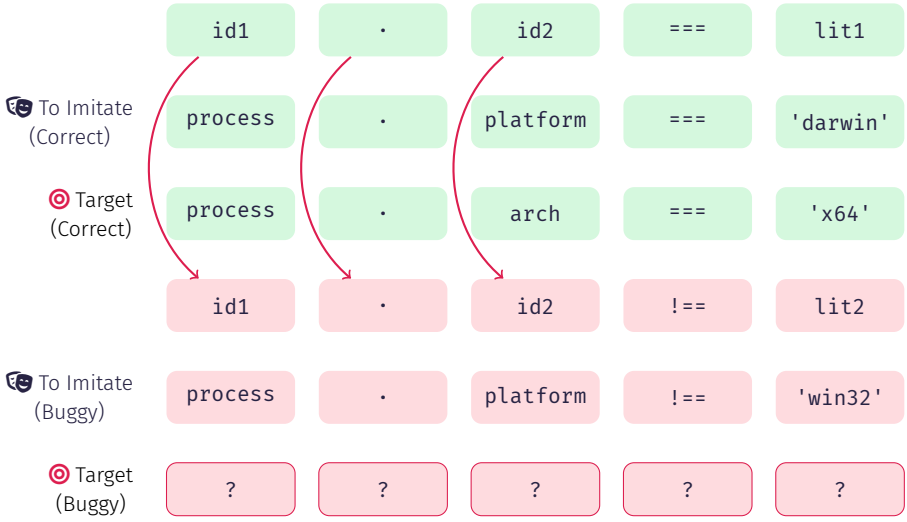
Apply Pattern



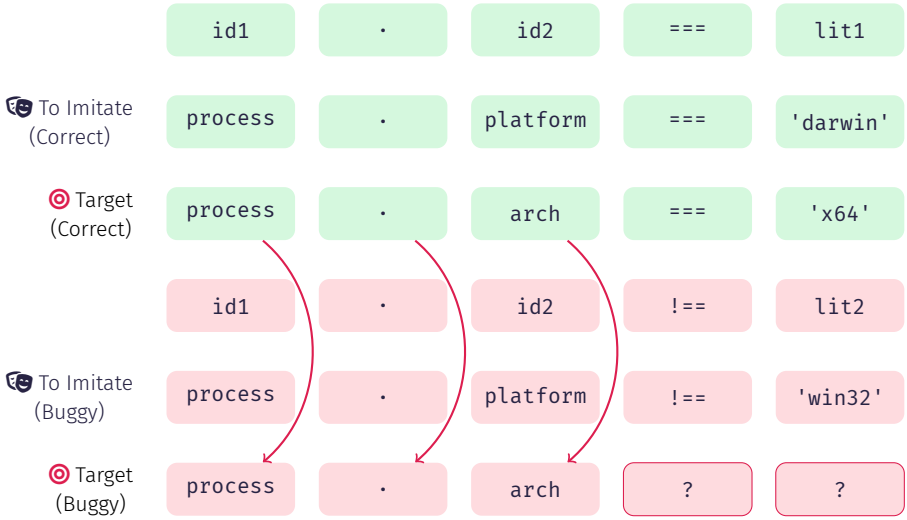
Apply Pattern



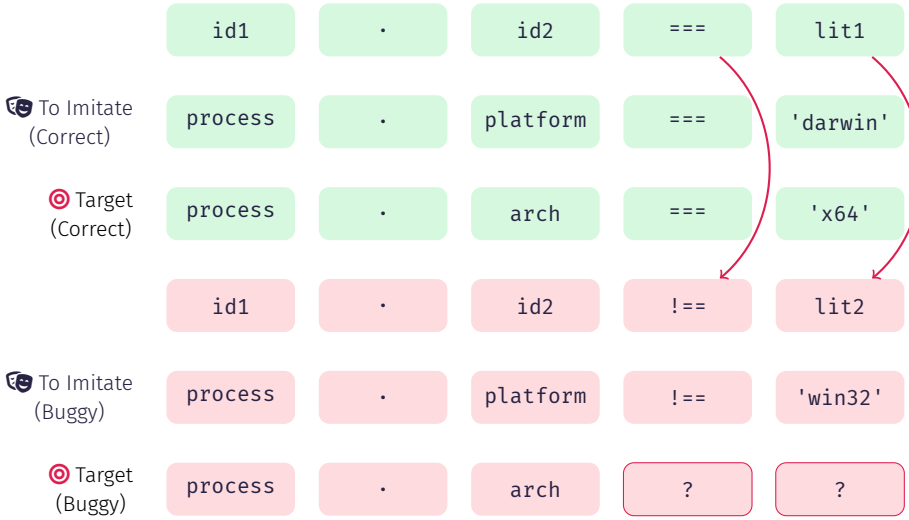
Apply Pattern



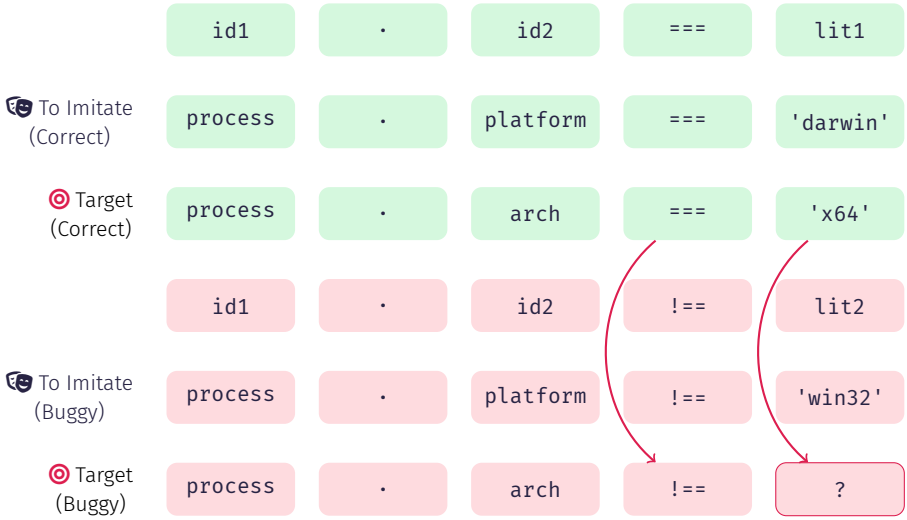
Apply Pattern



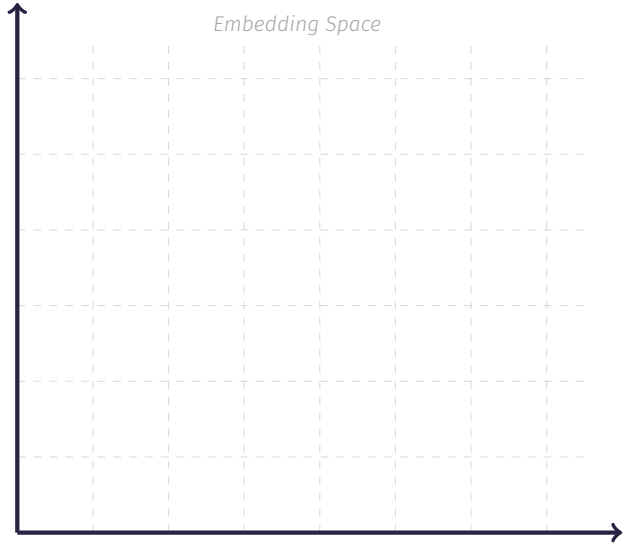
Apply Pattern



Apply Pattern

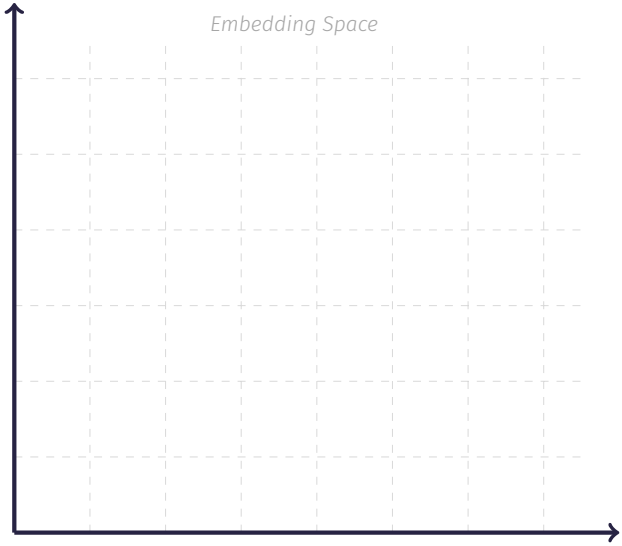


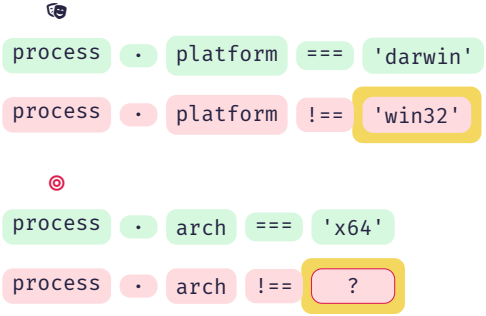
Analogy Queries for Binding Unbound Token



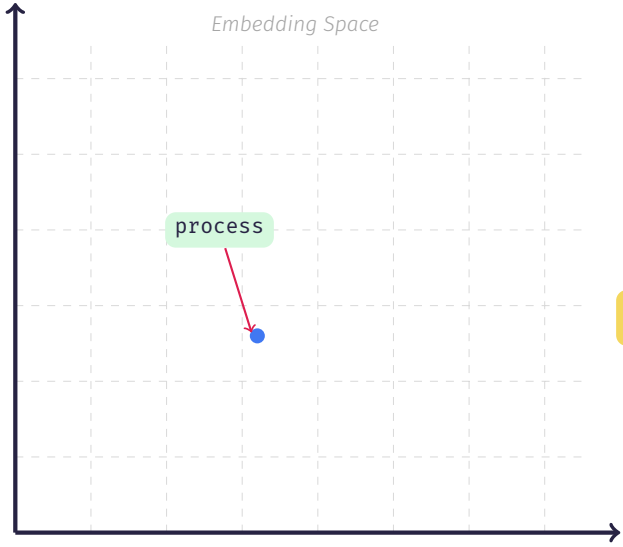

```
process . platform === 'darwin'
process . platform !== 'win32'
```

Analogy Queries for Binding Unbound Token



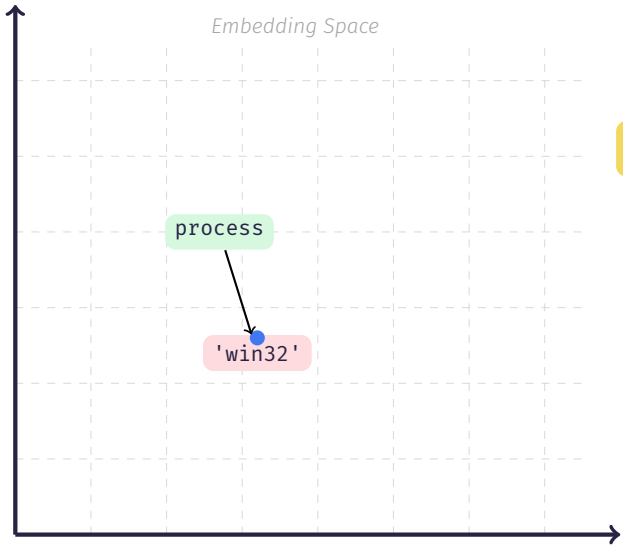


Analogy Queries for Binding Unbound Token




```
process . platform === 'darwin'
process . platform !== 'win32'

process . arch === 'x64'
process . arch !== ?
```

Analogy Queries for Binding Unbound Token



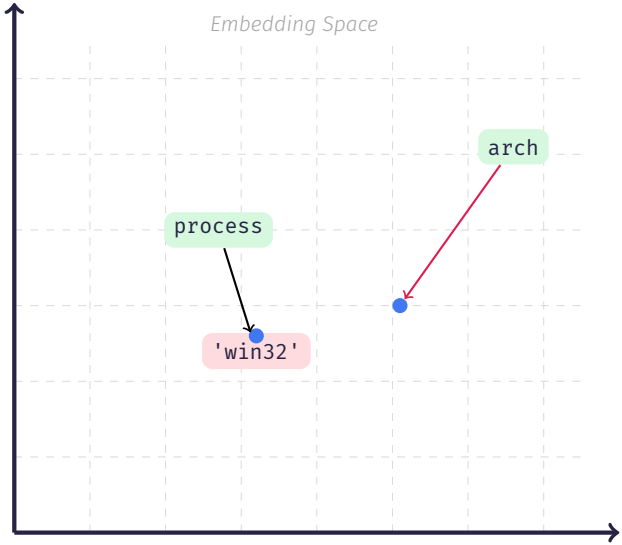
process · platform === 'darwin'

process · platform !== 'win32'

process · arch === 'x64'

process · arch !== ?

Analogy Queries for Binding Unbound Token



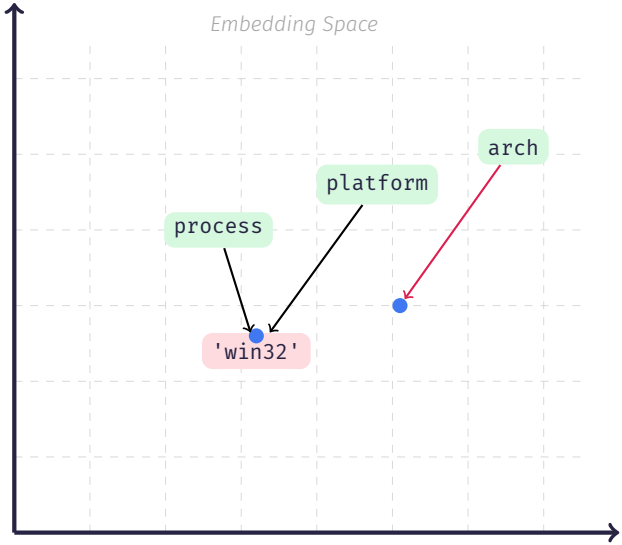
process . platform === 'darwin'

process . platform !== 'win32'

process . arch === 'x64'

process . arch !== ?

Analogy Queries for Binding Unbound Token



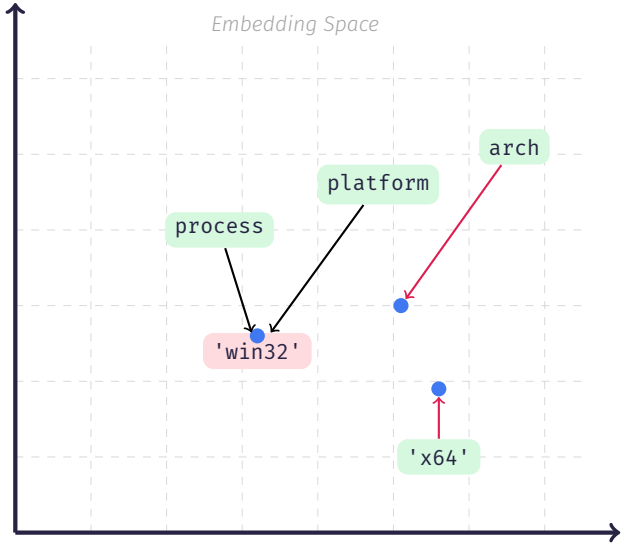
process · platform === 'darwin'

process · platform !== 'win32'

process · arch === 'x64'

process · arch !== ?

Analogy Queries for Binding Unbound Token



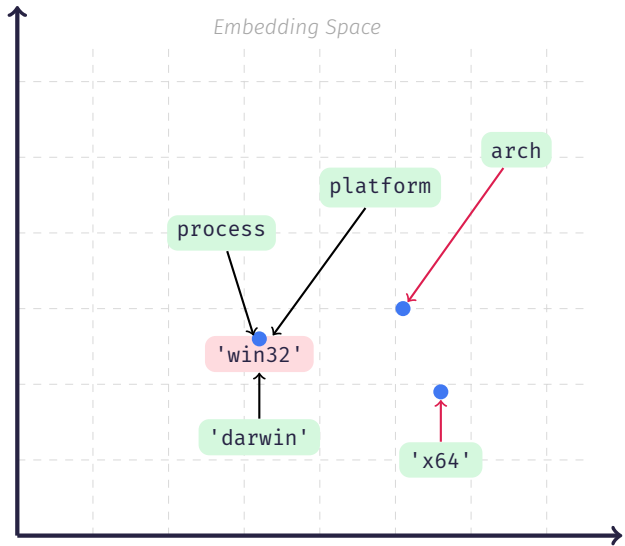
process · platform === 'darwin'

process · platform !== 'win32'

process · arch === 'x64'

process · arch !== ?

Analogy Queries for Binding Unbound Token



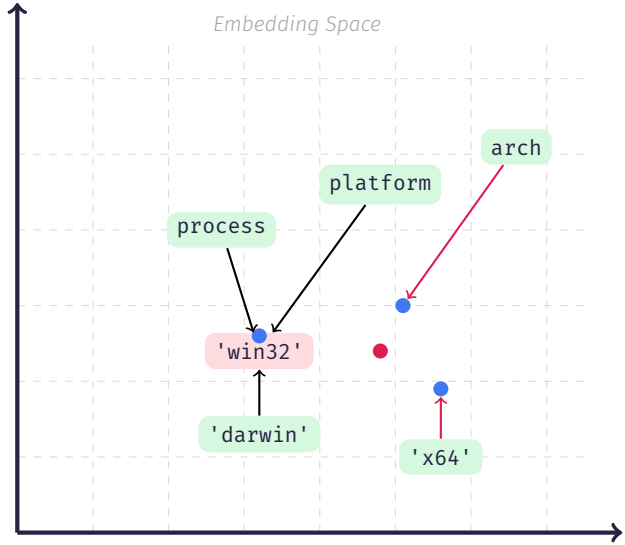
process . platform === 'darwin'

process . platform !== 'win32'

process . arch === 'x64'

process . arch !== ?

Analogy Queries for Binding Unbound Token



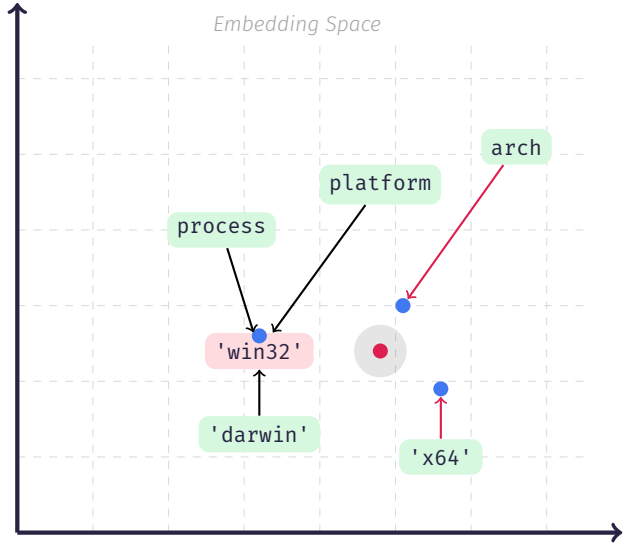
process · platform === 'darwin'

process · platform !== 'win32'

process · arch === 'x64'

process · arch !== ?

Analogy Queries for Binding Unbound Token



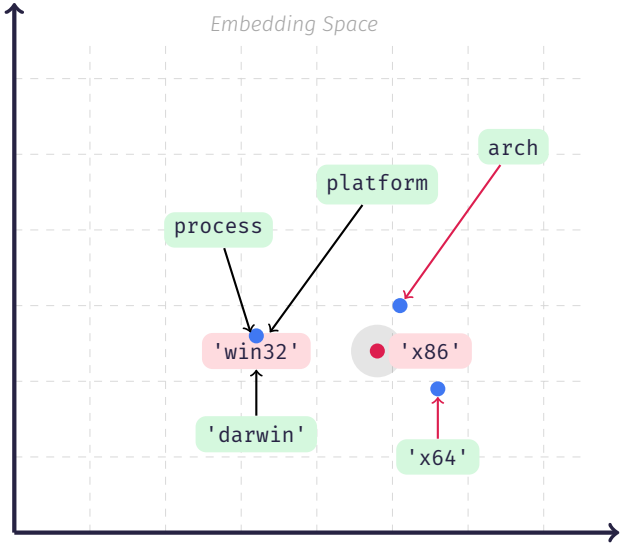
process · platform === 'darwin'

process · platform !== 'win32'

process · arch === 'x64'

process · arch !== ?

Analogy Queries for Binding Unbound Token



`process` · `platform` `===` `'darwin'`

`process` · `platform` `!==` `'win32'`

`process` · `arch` `===` `'x64'`

`process` · `arch` `!==` `'x86'`

Results

Experimental Setup

- 3600 concrete bug fixes extracted from GitHub

Experimental Setup

- 3600 concrete bug fixes extracted from GitHub
- 2880 (80%) used as *training bugs* and

Experimental Setup

- **3600** concrete bug fixes extracted from GitHub
- **2880** (80%) used as *training bugs* and
- Remaining **720** as *hold-out bugs*

Experimental Setup

- **3600** concrete bug fixes extracted from GitHub
- **2880** (80%) used as *training bugs* and
- Remaining **720** as *hold-out bugs*
- *training bugs* ∩ *hold-out bugs* → **53** bugs may be reproduced

Experimental Setup

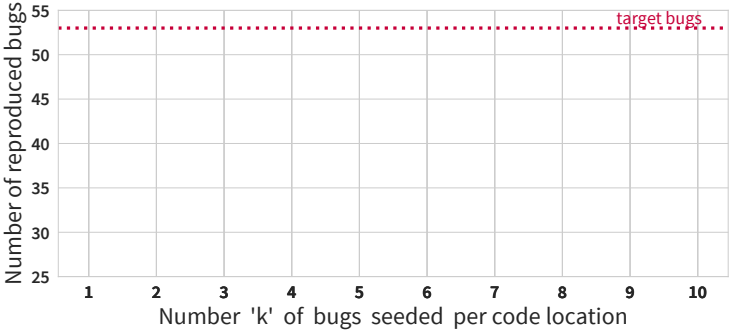
- **3600** concrete bug fixes extracted from GitHub
- **2880** (80%) used as *training bugs* and
- Remaining **720** as *hold-out bugs*
- *training bugs* ∩ *hold-out bugs* → **53** bugs may be reproduced
- *Reproduce:* Seeded bug exactly matches the real world bug

Primary Research Questions

RQ1. Can SemSeed reproduce real world bugs?

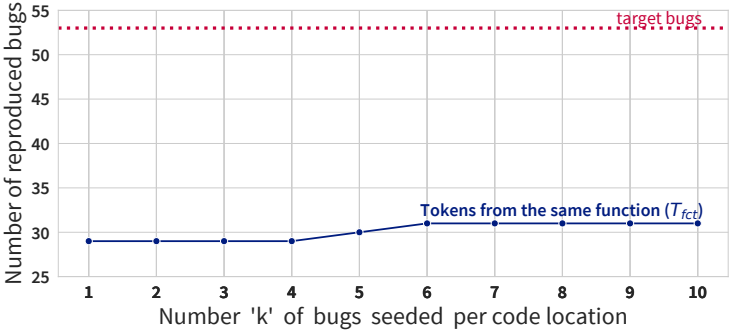
RQ2. How effective is SemSeed in training a learning-based bug detector?

RQ1. Can SemSeed Reproduce Real World Bugs



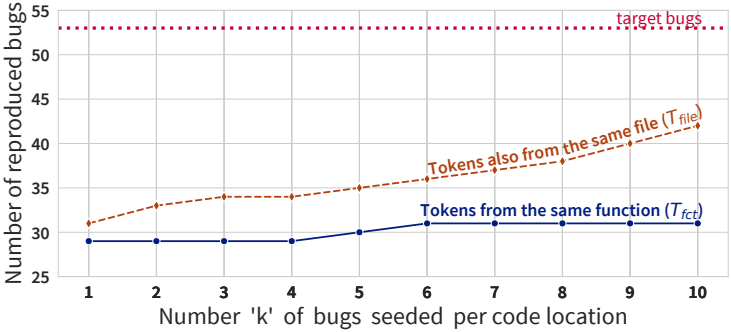
SemSeed could reproduce in total **47** bugs

RQ1. Can SemSeed Reproduce Real World Bugs



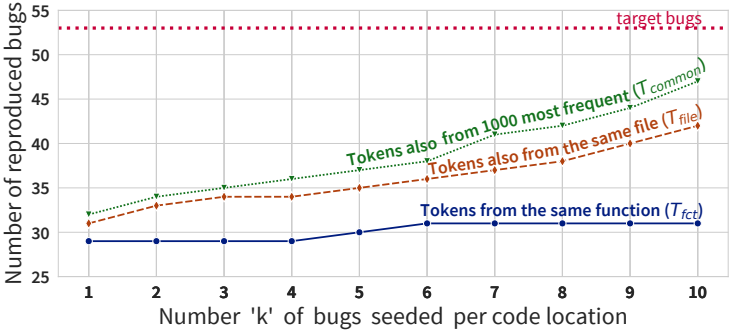
SemSeed could reproduce in total **47** bugs

RQ1. Can SemSeed Reproduce Real World Bugs



SemSeed could reproduce in total **47** bugs

RQ1. Can SemSeed Reproduce Real World Bugs



SemSeed could reproduce in total **47** bugs

RQ2. Effectiveness in Training a Learning-Based Bug Detector —— DeepBugs

- DeepBugs[1] is a learning-based bug detector that need large number correct and buggy examples for training.

[1] Michael Pradel and Koushik Sen, DeepBugs: A learning approach to name-based bug detection. OOPSLA (2018)

RQ2. Effectiveness in Training a Learning-Based Bug Detector —— DeepBugs

- DeepBugs[1] is a learning-based bug detector that need large number correct and buggy examples for training.
- Wrong assignment bugs:
  - → A developer writes `i=o;` instead of `i=0;`

[1] Michael Pradel and Koushik Sen, DeepBugs: A learning approach to name-based bug detection. OOPSLA (2018)

RQ2. Effectiveness in Training a Learning-Based Bug Detector —— DeepBugs

- DeepBugs[1] is a learning-based bug detector that need large number correct and buggy examples for training.
- Wrong assignment bugs:
  - → A developer writes `i=o;` instead of `i=0;`
- Train DeepBugs using two separate datasets:
  - → **Artificial**: Default in DeepBugs
  - → **SemSeed** generated

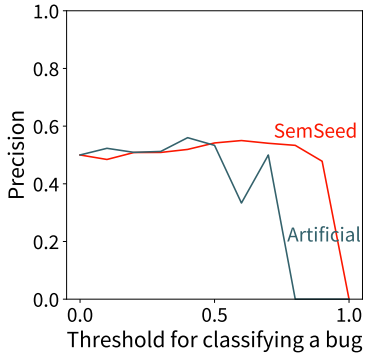
[1] Michael Pradel and Koushik Sen, DeepBugs: A learning approach to name-based bug detection. OOPSLA (2018)

RQ2. Effectiveness in Training a Learning-Based Bug Detector —— DeepBugs

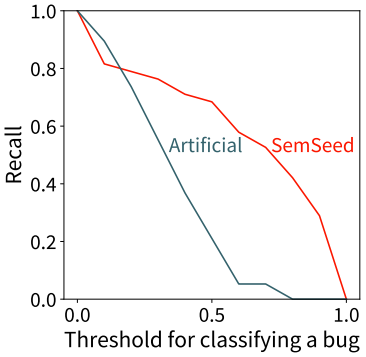
- DeepBugs[1] is a learning-based bug detector that need large number correct and buggy examples for training.
- Wrong assignment bugs:
  - → A developer writes `i=o;` instead of `i=0;`
- Train DeepBugs using two separate datasets:
  - → **Artificial**: Default in DeepBugs
  - → **SemSeed** generated
- Evaluate DeepBugs on the capability of finding real bugs

[1] Michael Pradel and Koushik Sen, DeepBugs: A learning approach to name-based bug detection. OOPSLA (2018)

Effectiveness in Training a Learning-Based Bug Finder —— DeepBugs



(a) Precision wrong assignments.



(b) Recall wrong assignments.

Other Findings

- **62%** of all bug seeding patterns contains at least one unbound token.
- Average bug seeding time **0.01** seconds.
- Seeded bugs goes beyond the traditional mutation operators.

Main Takeaways

Conclusions

- Many applications **need** large amount of realistic bugs.
- The current approaches are either **not scalable** or produce **unrealistic** bugs.
- SemSeed uses real bug fixes as **patterns**.
- Able to seed large number of **realistic** bugs quickly.
- The seeded bugs are **useful** for training learning-based bug detectors.

https://github.com/sola-st/SemSeed

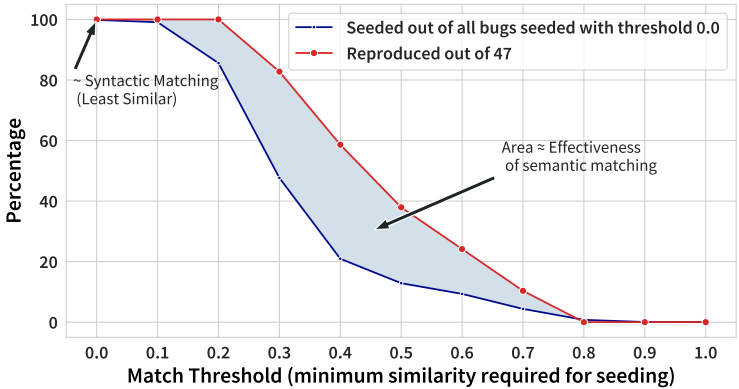
Thank You

Bug Seeding Patterns

Correct	Buggy	Nb.
`id1 : lit1`	`id1 : lit2`	99
`lit1 : lit2`	`lit1 : lit3`	71
`id1.id2(lit1);`	`id1.id2( lit2 );`	40
`var id1 = lit1;`	`var id1 = lit2 ;`	33
`id1 : lit1`	`id2 : lit1`	18
`id1 = lit1 in id2`	`id1 = !!id2. id3`	1
`id1.id2(lit1 + id3).id4);`	`id1.id2(lit1 + id3);`	2
`id1.id2(id3[id4.id5]);`	`id1.id2(id4.id5)`	2
`var id1 = id2.id3(id4);`	`var id1 = id2.id3;`	1
`var id1 = id2.id1;`	`var id1=id2. id3 ;`	5

Five most **frequent** and five **randomly** selected bug seeding patterns. Unbound tokens are highlighted

Usefulness of Semantic Matching



Comparison with Mutandis Mutation Operators

- Also present in SemSeed patterns:
  - Swap function parameters.
  - Change the variable type by converting `x = number` to `x = string`
- **Not** present in SemSeed patterns:
  - Swap consecutive nested `for/while`.
  - Removing the integer base argument 10 from `parseInt('09/11/08', 10)`.

Example of Rearrangement Bug

🎭 Bug to imitate

```
if (speed && typeof speed === "object"){
```

🎯 Seeded Bug

```
if (style && typeof style === 'object'){
```

🎭

```
if (typeof speed === "object"){
```

🎯

```
if (typeof style === 'object'){
```

Example of Unbound Token Bug

🎭 Bug to imitate

```
catalog. complete .getReleaseVersion
```

🎯 Seeded Bug

```
parent. stderr .on('data',
  function(){ });
```

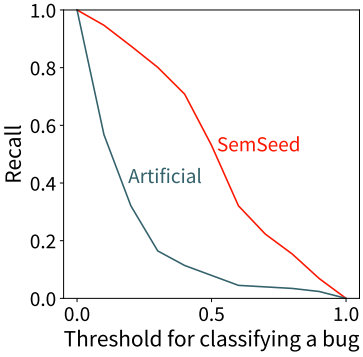
🎭

```
catalog. official .getReleaseVersion
```

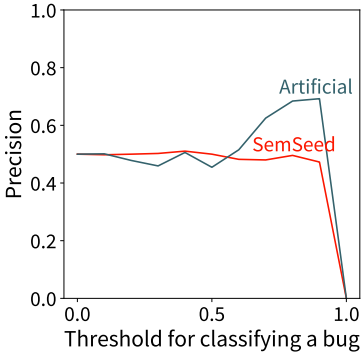
🎯

```
parent. stdout .on('data',
  function(){ });
```

Effectiveness in Training a Learning-Based Bug Finder —— DeepBugs



(a) Precision wrong assignments.



(b) Recall wrong assignments.