SemSeed: A Learning-Based Approach for Creating Realistic Bugs

Jibesh Patra, Michael Pradel
August 25, 2021
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 ✓

```java
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 ✓

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

Unrealistic Buggy 🤔

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

Realistic Buggy 🤔

```java
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

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

Unrealistic Buggy 🚫

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

Realistic Buggy 🗣️

```java
if(hasFailed && process.arch != 'x86')
```
An approach for automatically seeding **realistic** bugs in a semantics-aware way.

By **imitating** bugs occurring in the wild (top-100 GitHub projects).
Overview
Example Seeded Bug

1. Bug Fix to Imitate

2. Bug Seeding Pattern

3. Target Program

4. Bug Seeded Program
Example Seeded Bug

1. Bug Fix to Imitate
   
   ```javascript
   if (process.platform !== 'win32')
   ```

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

   ```javascript
   id1.id2 === lit1
   id1.id2 !== lit2
   ```

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

   3. Target Program

   2. Bug Seeding Pattern

   4. Bug Seeded Program
1. Bug Fix to Imitate

```javascript
if (process.platform !== 'win32')
if (process.platform === 'darwin')
```
Example Seeded Bug

1. Bug Fix to Imitate
   - if (process.platform !== 'win32')
   - if (process.platform === 'darwin')

2. Bug Seeding Pattern
   - id1.id2 === lit1
   - id1.id2 !== lit2

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

4. Bug Seeded Program
Example Seeded Bug

1. Bug Fix to Imitate

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

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

2. Bug Seeding Pattern

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

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

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

3. Target Program

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

4. Bug Seeded Program
Example Seeded Bug

1. Bug Fix to Imitate

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

2. Bug Seeding Pattern

```javascript
id1.id2 === lit1
id1.id2 !== lit2
```

3. Target Program

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

4. Bug Seeded Program
Example Seeded Bug

1. Bug Fix to Imitate
   - if (process.platform !== 'win32')
   - if (process.platform === 'darwin')

2. Bug Seeding Pattern
   - id1.id2 === lit1
   - id1.id2 !== lit2

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

4. Bug Seeded Program
   - hasFailed = item.errCode === -1;
   - if(hasFailed && process.arch === 'x64')
Example Seeded Bug

1. Bug Fix to Imitate
   ```javascript
   if (process.platform !== 'win32')
   if (process.platform === 'darwin')
   ```

2. Bug Seeding Pattern
   ```javascript
   id1.id2 === lit1
   id1.id2 !== lit2
   ```

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

4. Bug Seeded Program
   ```javascript
   hasFailed = item.errCode === -1;
   if (hasFailed && process.arch !== 'x86')
   ```
Example Seeded Bug

1. Bug Fix to Imitate
   - if (process.platform !== 'win32')
   - if (process.platform === 'darwin')

2. Bug Seeding Pattern
   - id1.id2 === lit1
   - id1.id2 !== lit2

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

4. Bug Seeded Program
   - hasFailed = item.errCode === -1;
   - if(hasFailed && process.arch !== 'x86')
Goal: If a bug fix to imitate is *semantic match* of a target.

- **process**: 
- **platform**: 'darwin'

To Imitate

- **process**: 
- **arch**: 'x64'

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

process . platform === 'darwin'

process . arch === 'x64'

To Imitate

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

<table>
<thead>
<tr>
<th>process</th>
<th>.</th>
<th>platform</th>
<th>===</th>
<th>'darwin'</th>
</tr>
</thead>
</table>

To Imitate

If average cosine similarity is less than a threshold $T$

<table>
<thead>
<tr>
<th>process</th>
<th>.</th>
<th>arch</th>
<th>===</th>
<th>'x64'</th>
</tr>
</thead>
</table>
Apply Pattern

1. Bug Fix to imitate
   
   ```javascript
   if (process.platform !== 'win32')
   ```

2. Bug Seeding Pattern
   
   ```javascript
   if (process.platform === 'darwin')
   ```

   ```javascript
   id1.id2 === lit1
   id1.id2 !== lit2)
   ```

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

4. Bug Seeded Program
   
   ```javascript
   hasFailed = item.errCode === -1;
   if(hasFailed && process.arch !== 'x86')
   ```
Apply Pattern

1. Bug Fix to imitate
   - if (process.platform !== 'win32')
   - if (process.platform === 'darwin')

2. Bug Seeding Pattern
   - id1.id2 === lit1
   - id1.id2 !== lit2)

Semantic Matching

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

4. Bug Seeded Program
   - hasFailed = item.errCode === -1;
   - if(hasFailed && process.arch !== 'x86')
Apply Pattern

<table>
<thead>
<tr>
<th>To Imitate (Correct)</th>
<th>process</th>
<th>.</th>
<th>platform</th>
<th>===</th>
<th>'darwin'</th>
</tr>
</thead>
<tbody>
<tr>
<td>Target (Correct)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

| To Imitate (Buggy)  |         |   |          |     |         |
| Target (Buggy)      |         |   |          |     |         |
Apply Pattern

To Imitate (Correct)

- process
- platform ===
- 'darwin'

Target (Correct)

- process
- arch ===
- 'x64'

To Imitate (Buggy)

- process
- platform !=

Target (Buggy)

- process
- arch !=
Apply Pattern

id1 . id2 === lit1

To Imitate (Correct)

process . platform === 'darwin'

Target (Correct)

process . arch === 'x64'

To Imitate (Buggy)

Target (Buggy)

---

7
### Apply Pattern

<table>
<thead>
<tr>
<th>id1</th>
<th>.</th>
<th>id2</th>
<th>==</th>
<th>lit1</th>
</tr>
</thead>
<tbody>
<tr>
<td>process</td>
<td>.</td>
<td>platform</td>
<td>==</td>
<td>'darwin'</td>
</tr>
</tbody>
</table>

#### To Imitate (Correct)

<table>
<thead>
<tr>
<th>id1</th>
<th>.</th>
<th>id2</th>
<th>==</th>
<th>lit1</th>
</tr>
</thead>
<tbody>
<tr>
<td>process</td>
<td>.</td>
<td>arch</td>
<td>==</td>
<td>'x64'</td>
</tr>
</tbody>
</table>

#### Target (Correct)

<table>
<thead>
<tr>
<th>id1</th>
<th>.</th>
<th>id2</th>
<th>==</th>
<th>lit1</th>
</tr>
</thead>
<tbody>
<tr>
<td>process</td>
<td>.</td>
<td>platform</td>
<td>!=</td>
<td>'win32'</td>
</tr>
</tbody>
</table>

#### To Imitate (Buggy)

<table>
<thead>
<tr>
<th>id1</th>
<th>.</th>
<th>id2</th>
<th>==</th>
<th>lit1</th>
</tr>
</thead>
<tbody>
<tr>
<td>process</td>
<td>.</td>
<td>platform</td>
<td>!=</td>
<td>'win32'</td>
</tr>
</tbody>
</table>

#### Target (Buggy)
### Apply Pattern

<table>
<thead>
<tr>
<th></th>
<th>id1</th>
<th>.</th>
<th>id2</th>
<th>===</th>
<th>lit1</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>To Imitate</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>(Correct)</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>process</td>
<td></td>
<td></td>
<td>platform</td>
<td></td>
<td>'darwin'</td>
</tr>
<tr>
<td><strong>Target</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>(Correct)</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>process</td>
<td></td>
<td></td>
<td>arch</td>
<td></td>
<td>'x64'</td>
</tr>
<tr>
<td>id1</td>
<td></td>
<td></td>
<td>id2</td>
<td>!==</td>
<td>lit2</td>
</tr>
<tr>
<td><strong>To Imitate</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>(Buggy)</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>process</td>
<td></td>
<td></td>
<td>platform</td>
<td>!==</td>
<td>'win32'</td>
</tr>
</tbody>
</table>

To Imitate (Correct)

To Imitate (Buggy)

Target (Correct)

Target (Buggy)
### Apply Pattern

#### To Imitate (Correct)
<table>
<thead>
<tr>
<th>id1</th>
<th>.</th>
<th>id2</th>
<th>===</th>
<th>lit1</th>
</tr>
</thead>
<tbody>
<tr>
<td>process</td>
<td>.</td>
<td>platform</td>
<td>===</td>
<td>'darwin'</td>
</tr>
</tbody>
</table>

#### Target (Correct)
<table>
<thead>
<tr>
<th>id1</th>
<th>.</th>
<th>id2</th>
<th>!==</th>
<th>lit2</th>
</tr>
</thead>
<tbody>
<tr>
<td>process</td>
<td>.</td>
<td>arch</td>
<td>!==</td>
<td>'x64'</td>
</tr>
</tbody>
</table>

#### To Imitate (Buggy)
<table>
<thead>
<tr>
<th>id1</th>
<th>.</th>
<th>id2</th>
<th>!==</th>
<th>lit2</th>
</tr>
</thead>
<tbody>
<tr>
<td>process</td>
<td>.</td>
<td>platform</td>
<td>!==</td>
<td>'win32'</td>
</tr>
</tbody>
</table>

#### Target (Buggy)
Apply Pattern

To Imitate (Correct)

Target (Correct)

To Imitate (Buggy)

Target (Buggy)
### Apply Pattern

<table>
<thead>
<tr>
<th></th>
<th>id1</th>
<th></th>
<th>id2</th>
<th>===</th>
<th>lit1</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>To Imitate (Correct)</strong></td>
<td>process</td>
<td></td>
<td>platform</td>
<td>===</td>
<td>'darwin'</td>
</tr>
<tr>
<td><strong>Target (Correct)</strong></td>
<td>process</td>
<td></td>
<td>arch</td>
<td>===</td>
<td>'x64'</td>
</tr>
<tr>
<td></td>
<td>id1</td>
<td></td>
<td>id2</td>
<td>!==</td>
<td>lit2</td>
</tr>
<tr>
<td><strong>To Imitate (Buggy)</strong></td>
<td>process</td>
<td></td>
<td>platform</td>
<td>!==</td>
<td>'win32'</td>
</tr>
<tr>
<td><strong>Target (Buggy)</strong></td>
<td>process</td>
<td></td>
<td>arch</td>
<td>?</td>
<td>?</td>
</tr>
</tbody>
</table>
**Apply Pattern**

<table>
<thead>
<tr>
<th>id1</th>
<th>·</th>
<th>id2</th>
<th>===</th>
<th>lit1</th>
</tr>
</thead>
<tbody>
<tr>
<td>process</td>
<td>·</td>
<td>platform</td>
<td>===</td>
<td>'darwin'</td>
</tr>
</tbody>
</table>

- **To Imitate (Correct)**
  - Target (Correct)
    - id1 | · | id2 | !== | lit2 |
    - process | · | arch | !== | 'x64' |

- **To Imitate (Buggy)**
  - Target (Buggy)
    - process | · | platform | !== | 'win32' |

- Target (Buggy)
  - process | · | arch | ? | ?
Apply Pattern

<table>
<thead>
<tr>
<th>To Imitate (Correct)</th>
<th>Target (Correct)</th>
<th>To Imitate (Buggy)</th>
<th>Target (Buggy)</th>
</tr>
</thead>
<tbody>
<tr>
<td>id1</td>
<td>process</td>
<td>id1</td>
<td>process</td>
</tr>
<tr>
<td>.</td>
<td>.</td>
<td>id2</td>
<td>process</td>
</tr>
<tr>
<td>id2</td>
<td>platform</td>
<td>==</td>
<td>platform</td>
</tr>
<tr>
<td>==</td>
<td>==</td>
<td>lit1</td>
<td>==</td>
</tr>
<tr>
<td>lit1</td>
<td>'darwin'</td>
<td>process</td>
<td>'win32'</td>
</tr>
<tr>
<td>process</td>
<td>arch</td>
<td>process</td>
<td>arch</td>
</tr>
<tr>
<td>arch</td>
<td>==</td>
<td>process</td>
<td>==</td>
</tr>
<tr>
<td>==</td>
<td>lit2</td>
<td>'x64'</td>
<td>?</td>
</tr>
<tr>
<td>?</td>
<td>?</td>
<td>?</td>
<td>?</td>
</tr>
</tbody>
</table>
Analogy Queries for Binding Unbound Token

Embedding Space

- process \cdot platform \equiv 'darwin'

- process \cdot platform \not\equiv 'win32'
Analogy Queries for Binding Unbound Token

Embedding Space

- process
  - platform == 'darwin'
- process
  - platform != 'win32'
- process
  - arch == 'x64'
- process
  - arch != ?
Embedding Space

- process
- platform === 'darwin'
- process platform !== 'win32'
- process arch === 'x64'
- process arch !== ?
Analogy Queries for Binding Unbound Token

Embedding Space

process

platform

process

platform

process

arch

process

arch

process

arch
Analogy Queries for Binding Unbound Token

`platform` \( \neq \) \( 'win32' \)

`arch` \( \neq \) \( ? \)
Analogy Queries for Binding Unbound Token

Embedding Space

process

platform

arch

'win32'

platform === 'darwin'

platform !== 'win32'

arch === 'x64'

arch !== 'x86'

?
Analogy Queries for Binding Unbound Token

Embedding Space

- process  •  platform  ===  'darwin'
- process  •  platform  !==  'win32'
- process  •  arch  ===  'x64'
- process  •  arch  !==  'x86'
Analogy Queries for Binding Unbound Token

Embedding Space

- process
- platform
- arch

<table>
<thead>
<tr>
<th>process</th>
<th>platform</th>
<th>arch</th>
</tr>
</thead>
<tbody>
<tr>
<td>'win32'</td>
<td></td>
<td></td>
</tr>
<tr>
<td>'darwin'</td>
<td></td>
<td></td>
</tr>
<tr>
<td>'x64'</td>
<td></td>
<td></td>
</tr>
<tr>
<td>'win32'</td>
<td></td>
<td></td>
</tr>
<tr>
<td>'darwin'</td>
<td></td>
<td></td>
</tr>
<tr>
<td>'x86'</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Analogy Queries for Binding Unbound Token

Embedding Space

- **process**
  - platform === 'darwin'
  - platform !== 'win32'
  - arch === 'x64'
  - arch !== ?

- **platform**
  - platform === 'darwin'
  - platform !== 'win32'

- **arch**
  - arch === 'x64'
  - arch !== ?
Analogy Queries for Binding Unbound Token

Embedding Space

- \text{process}
- \text{platform}
- \text{arch}

<table>
<thead>
<tr>
<th>process</th>
<th>platform</th>
<th>arch</th>
</tr>
</thead>
<tbody>
<tr>
<td>'win32'</td>
<td>'darwin'</td>
<td>'x64'</td>
</tr>
<tr>
<td>'darwin'</td>
<td>'win32'</td>
<td>'x86'</td>
</tr>
<tr>
<td>'win32'</td>
<td>'win32'</td>
<td>'x64'</td>
</tr>
<tr>
<td>'darwin'</td>
<td>'darwin'</td>
<td>'x86'</td>
</tr>
</tbody>
</table>

- \text{process} \cdot \text{platform} === 'darwin'
- \text{process} \cdot \text{platform} !== 'win32'
- \text{process} \cdot \text{arch} === 'x64'
- \text{process} \cdot \text{arch} !== 'x86'
Analogy Queries for Binding Unbound Token

```
process \cdot platform === 'darwin'
process \cdot platform !== 'win32'
process \cdot arch === 'x64'
process \cdot arch !== 'x86'
```
Results
3600 concrete bug fixes extracted from GitHub
Experimental Setup

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

Remaining 720 as *hold-out bugs* → 53 bugs may be reproduced

Reproduce:
Seeded bug exactly matches the real world bug
Experimental Setup

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

2880 (80%) used as *training bugs* and

Remaining 720 as *hold-out bugs*

*training bugs* $\cap$ *hold-out bugs* $\rightarrow$ 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

Number 'k' of bugs seeded per code location

Number of reproduced bugs

Tokens also from the same file ($T_{file}$)

Tokens from the same function ($T_{fct}$)

SemSeed could reproduce in total 47 bugs
RQ1. Can SemSeed Reproduce Real World Bugs

SemSeed could reproduce in total 47 bugs
DeepBugs[1] is a learning-based bug detector that need large number correct and buggy examples for training.

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=0;` instead of `i=\theta;`

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=0;` instead of `i=0;`

Train DeepBugs using two separate datasets:

- **Artificial**: Default in DeepBugs
- **SemSeed**: generated

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

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

(a) Precision wrong assignments.

(b) Recall wrong assignments.
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
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

<table>
<thead>
<tr>
<th>Correct</th>
<th>Buggy</th>
<th>Nb.</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>id1 : lit1</code></td>
<td><code>id1 : lit2</code></td>
<td>99</td>
</tr>
<tr>
<td><code>lit1 : lit2</code></td>
<td><code>lit1 : lit3</code></td>
<td>71</td>
</tr>
<tr>
<td><code>id1.id2(lit1);</code></td>
<td><code>id1.id2(lit2);</code></td>
<td>40</td>
</tr>
<tr>
<td><code>var id1 = lit1;</code></td>
<td><code>var id1 = lit2;</code></td>
<td>33</td>
</tr>
<tr>
<td><code>id1 : lit1</code></td>
<td><code>id2 : lit1</code></td>
<td>18</td>
</tr>
<tr>
<td><code>id1 = lit1 in id2</code></td>
<td><code>id1 = !!id2.id3</code></td>
<td>1</td>
</tr>
<tr>
<td><code>id1.id2(lit1 + id3).id4);</code></td>
<td><code>id1.id2(lit1 + id3);</code></td>
<td>2</td>
</tr>
<tr>
<td><code>id1.id2(id3[id4.id5]);</code></td>
<td><code>id1.id2(id4.id5)</code></td>
<td>2</td>
</tr>
<tr>
<td><code>var id1 = id2.id3(id4);</code></td>
<td><code>var id1 = id2.id3;</code></td>
<td>1</td>
</tr>
<tr>
<td><code>var id1 = id2.id1;</code></td>
<td><code>var id1 =id2. id3 ;</code></td>
<td>5</td>
</tr>
</tbody>
</table>

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

Match Threshold (minimum similarity required for seeding)

- Syntactic Matching (Least Similar)

Seeded out of all bugs seeded with threshold 0.0
Reproduced out of 47

Area ≈ Effectiveness 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.