UnitCon: Synthesizing Targeted Unit Tests for Java Runtime Exceptions

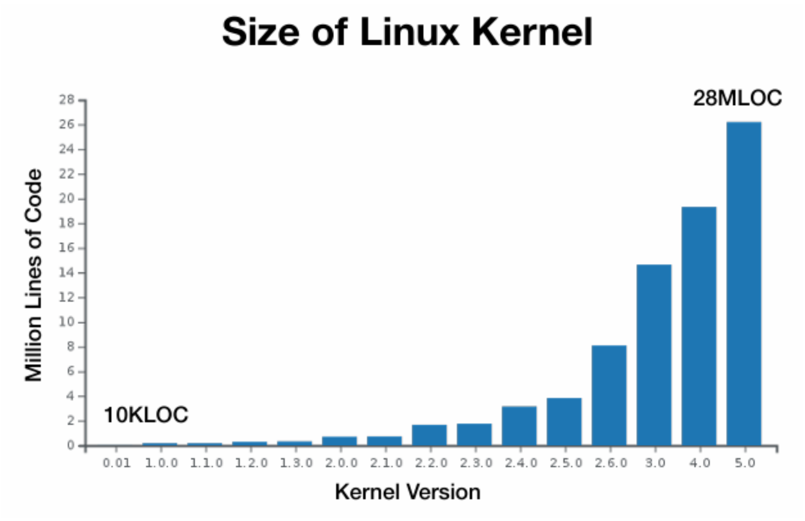
Sujin Jang, Yeonhee Ryou, Heewon Lee, Kihong Heo

KAIST Programming Systems Laboratory

Motivation

Motivation

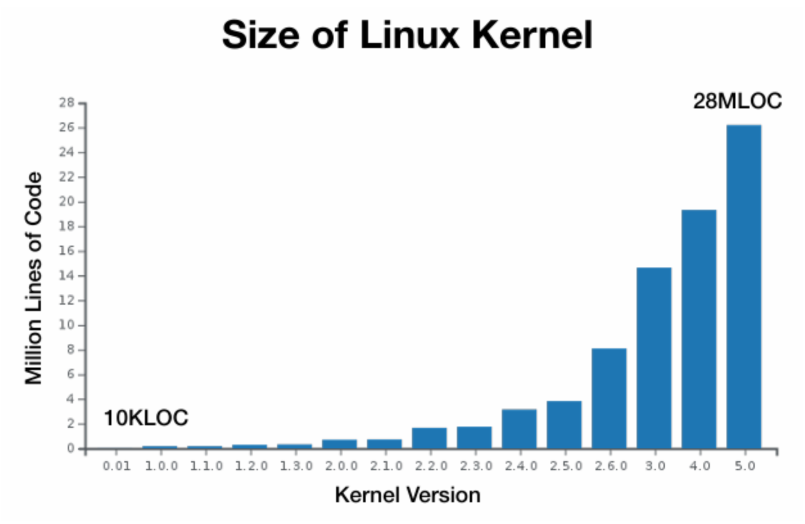
- Software is getting larger, with frequent changes.



Program	Commits (2024.3)	Active Developers (2024.3)
Linux Kernel	864	59
LLVM	3,525	56
V8	800	31
OpenSSL	80	17
Elastic Search	514	49

Motivation

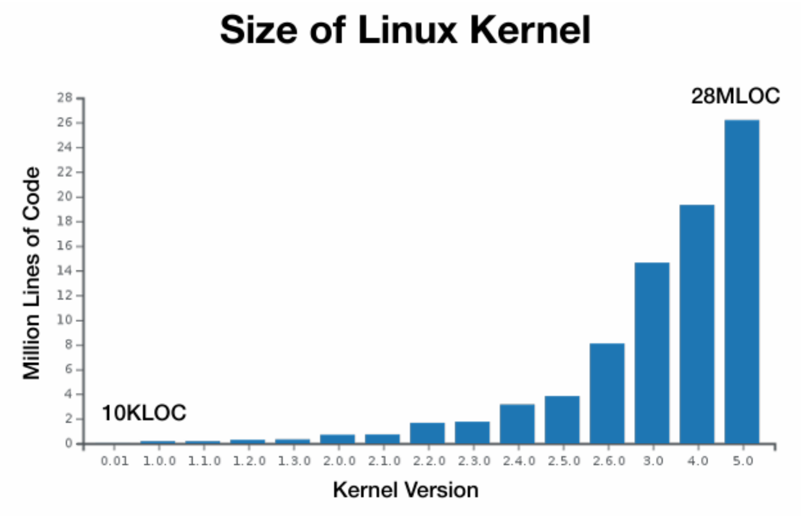
- Software is getting larger, with frequent changes.
- Increasing need for **targeted unit testing**.



Program	Commits (2024.3)	Active Developers (2024.3)
Linux Kernel	864	59
LLVM	3,525	56
V8	800	31
OpenSSL	80	17
Elastic Search	514	49

Motivation

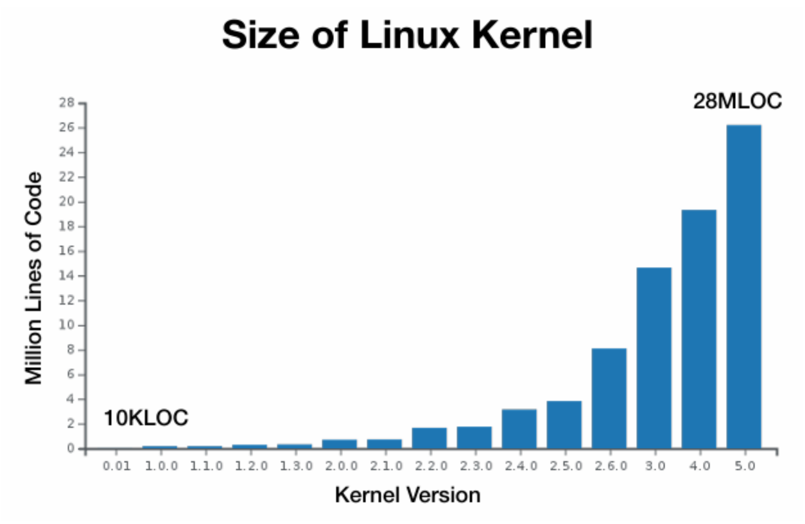
- Software is getting larger, with frequent changes.
- Increasing need for **targeted unit testing**.
  - Aims to reveal a bug at a given specific location.



Program	Commits (2024.3)	Active Developers (2024.3)
Linux Kernel	864	59
LLVM	3,525	56
V8	800	31
OpenSSL	80	17
Elastic Search	514	49

Motivation

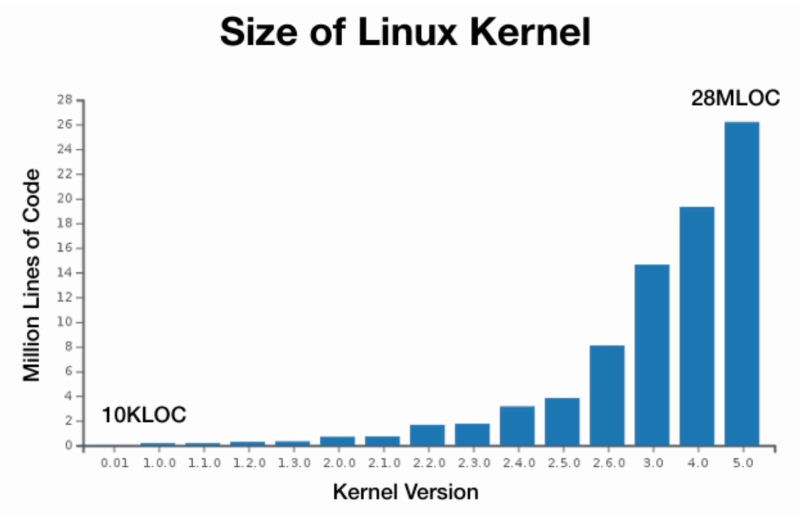
- Software is getting larger, with frequent changes.
- Increasing need for **targeted unit testing**.
  - Aims to reveal a bug at a given specific location.
  - e.g., continuous integration, static analysis alarm inspection.



Program	Commits (2024.3)	Active Developers (2024.3)
Linux Kernel	864	59
LLVM	3,525	56
V8	800	31
OpenSSL	80	17
Elastic Search	514	49

Motivation

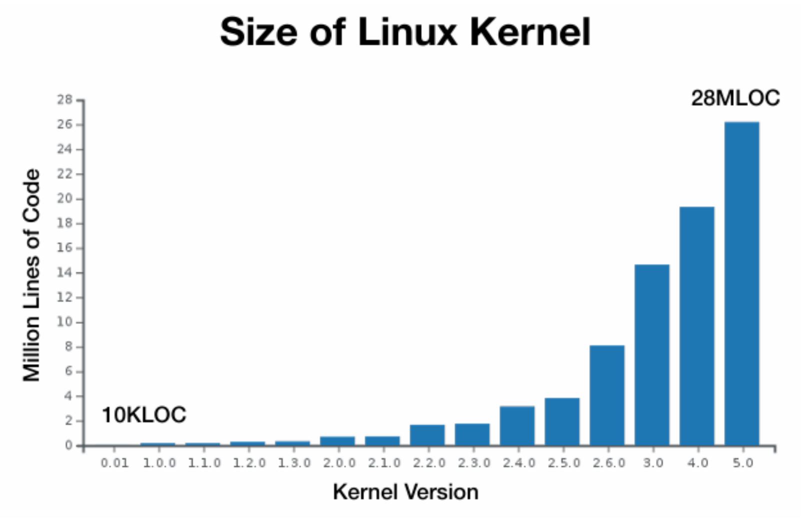
- Software is getting larger, with frequent changes.
- Increasing need for **targeted unit testing**.
    - Aims to reveal a bug at a given specific location.
    - e.g., continuous integration, static analysis alarm inspection.
- Conventional test case generation techniques are not effective for targeted testing.



Program	Commits (2024.3)	Active Developers (2024.3)
Linux Kernel	864	59
LLVM	3,525	56
V8	800	31
OpenSSL	80	17
Elastic Search	514	49

Motivation

- Software is getting larger, with frequent changes.
- Increasing need for **targeted unit testing**.
  - Aims to reveal a bug at a given specific location.
  - e.g., continuous integration, static analysis alarm inspection.
- Conventional test case generation techniques are not effective for targeted testing.
  - Aim to generate regression tests by maximizing code coverage.



Program	Commits (2024.3)	Active Developers (2024.3)
Linux Kernel	864	59
LLVM	3,525	56
V8	800	31
OpenSSL	80	17
Elastic Search	514	49

Problem

Problem

- **Exponential growth of partial test cases** makes simple synthesis ineffective.

Problem

- **Exponential growth of partial test cases** makes simple synthesis ineffective.

* JSqlParser (13K LOC)

```
public void test() {
  Adapter recv = new Adapter();
  recv.M(ID);
  Select select = new Select();
  recv.visit(select);
}
```

Problem

- **Exponential growth of partial test cases** makes simple synthesis ineffective.

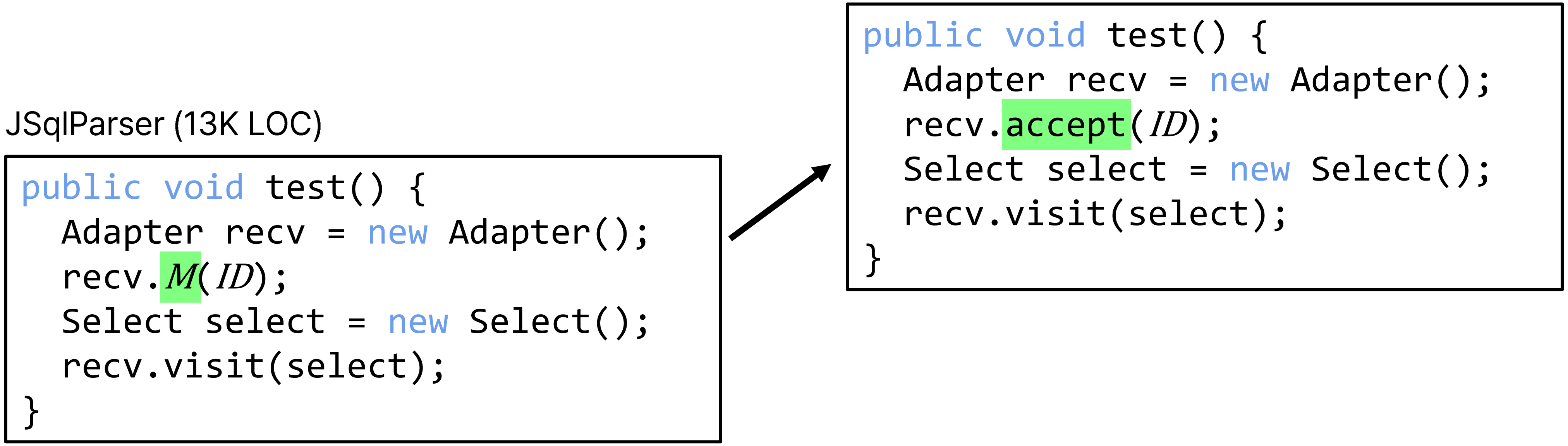
* JSqlParser (13K LOC)

```
public void test() {
  Adapter recv = new Adapter();
  recv.M(ID);
  Select select = new Select();
  recv.visit(select);
}
```

Problem

- **Exponential growth of partial test cases** makes simple synthesis ineffective.

* JSqlParser (13K LOC)



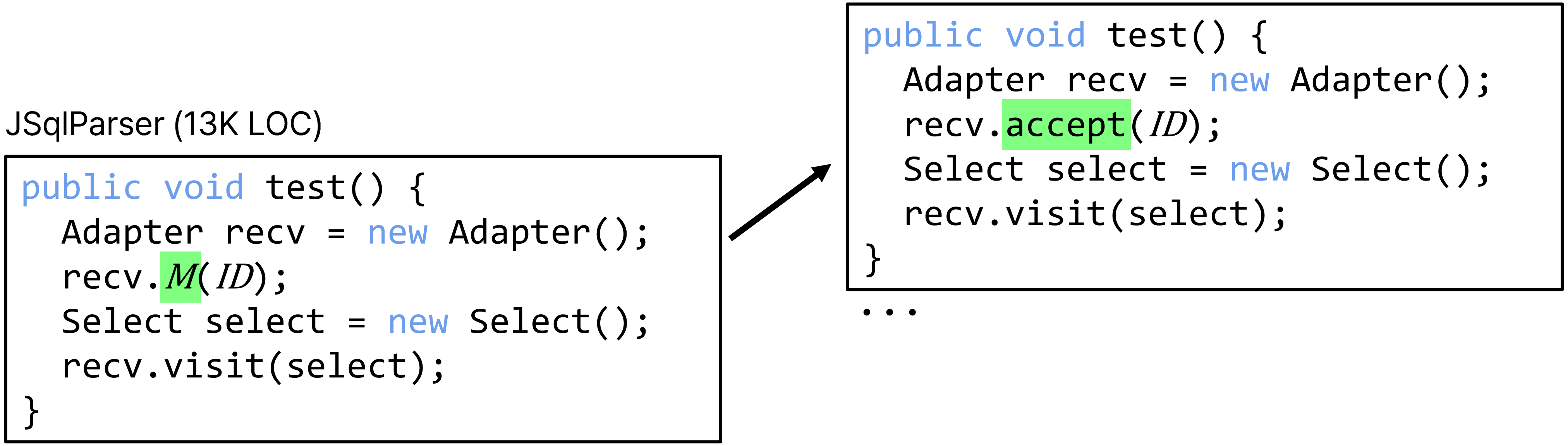
```
public void test() {
  Adapter recv = new Adapter();
  recv.M(ID);
  Select select = new Select();
  recv.visit(select);
}
```

```
public void test() {
  Adapter recv = new Adapter();
  recv.accept(ID);
  Select select = new Select();
  recv.visit(select);
}
```

Problem

- **Exponential growth of partial test cases** makes simple synthesis ineffective.

* JSqlParser (13K LOC)



```
public void test() {
  Adapter recv = new Adapter();
  recv.M(ID);
  Select select = new Select();
  recv.visit(select);
}
```

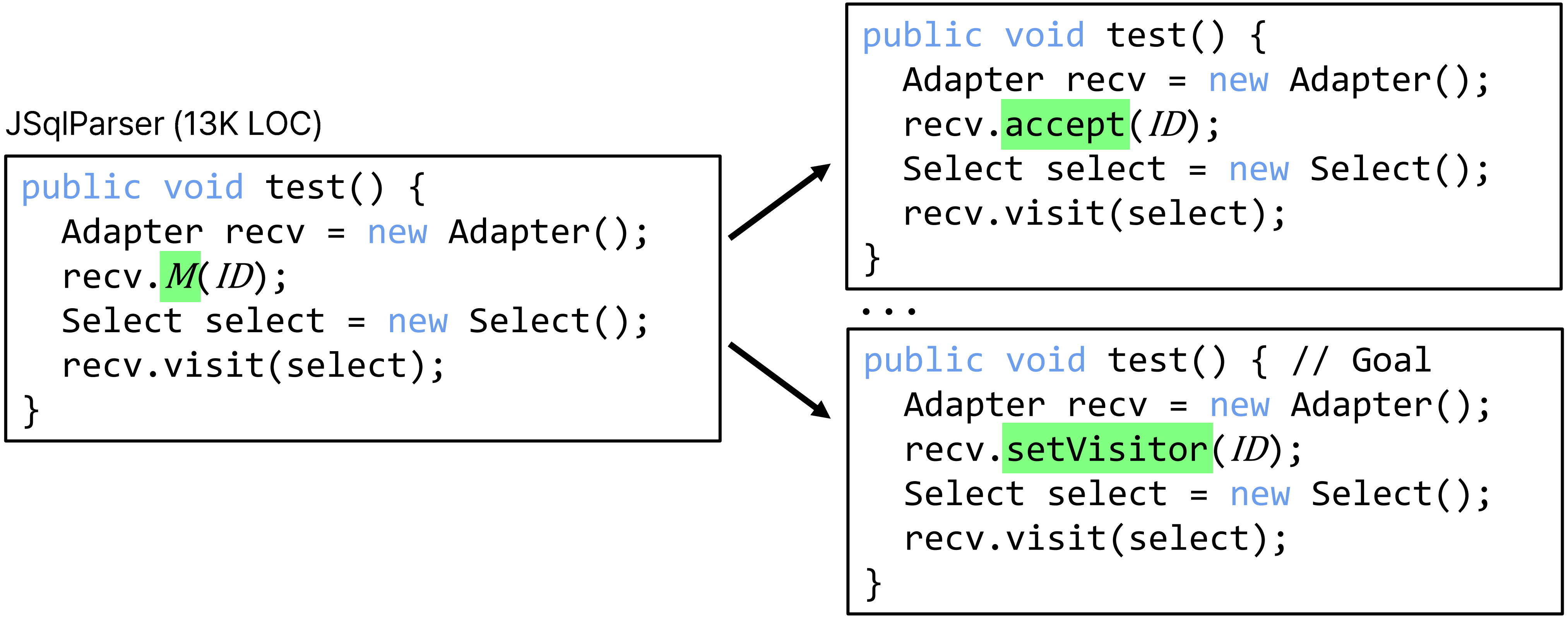
```
public void test() {
  Adapter recv = new Adapter();
  recv.accept(ID);
  Select select = new Select();
  recv.visit(select);
}
```

...

Problem

- **Exponential growth of partial test cases** makes simple synthesis ineffective.

* JSqlParser (13K LOC)



```
public void test() {
  Adapter recv = new Adapter();
  recv.M(ID);
  Select select = new Select();
  recv.visit(select);
}
```

```
public void test() {
  Adapter recv = new Adapter();
  recv.accept(ID);
  Select select = new Select();
  recv.visit(select);
}
```

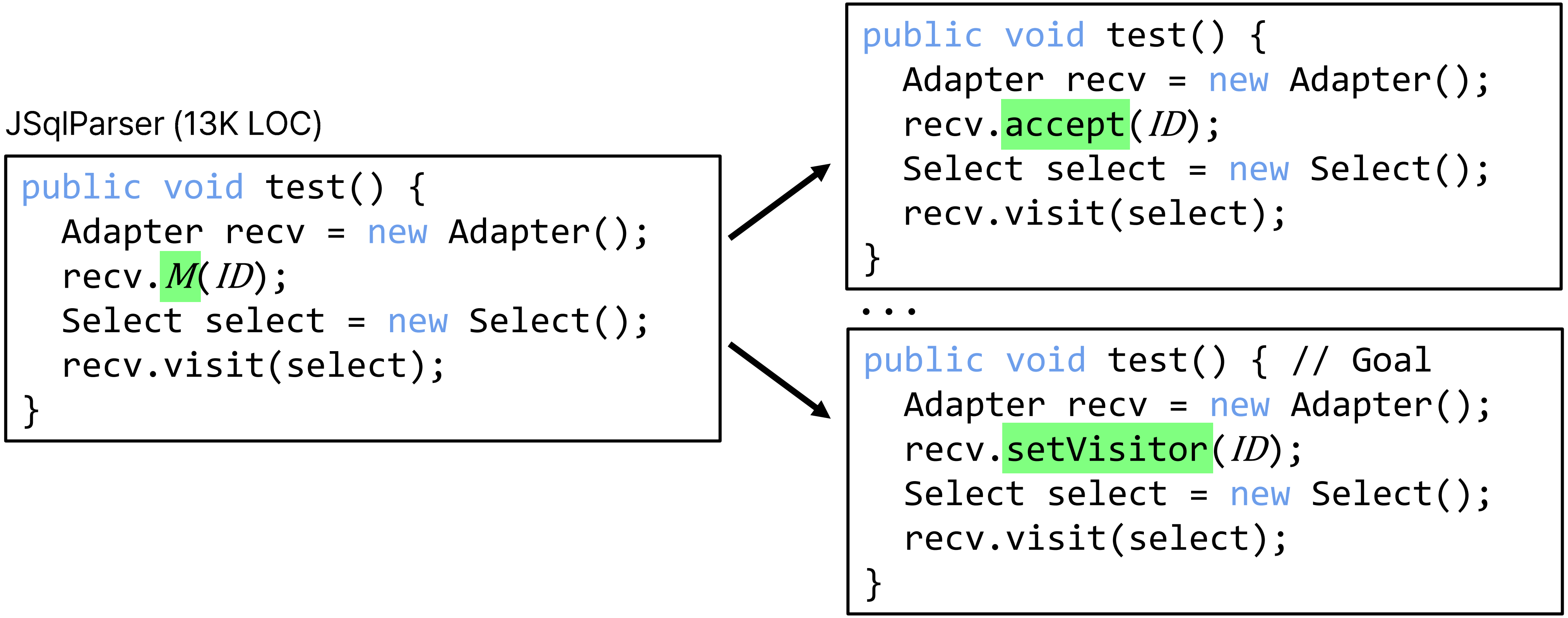
...

```
public void test() { // Goal
  Adapter recv = new Adapter();
  recv.setVisitor(ID);
  Select select = new Select();
  recv.visit(select);
}
```

Problem

- **Exponential growth of partial test cases** makes simple synthesis ineffective.
    - Only **one** step can generate thousands of partial test cases.

* JSqlParser (13K LOC)



```
public void test() {
  Adapter recv = new Adapter();
  recv.M(ID);
  Select select = new Select();
  recv.visit(select);
}
```

```
public void test() {
  Adapter recv = new Adapter();
  recv.accept(ID);
  Select select = new Select();
  recv.visit(select);
}
```

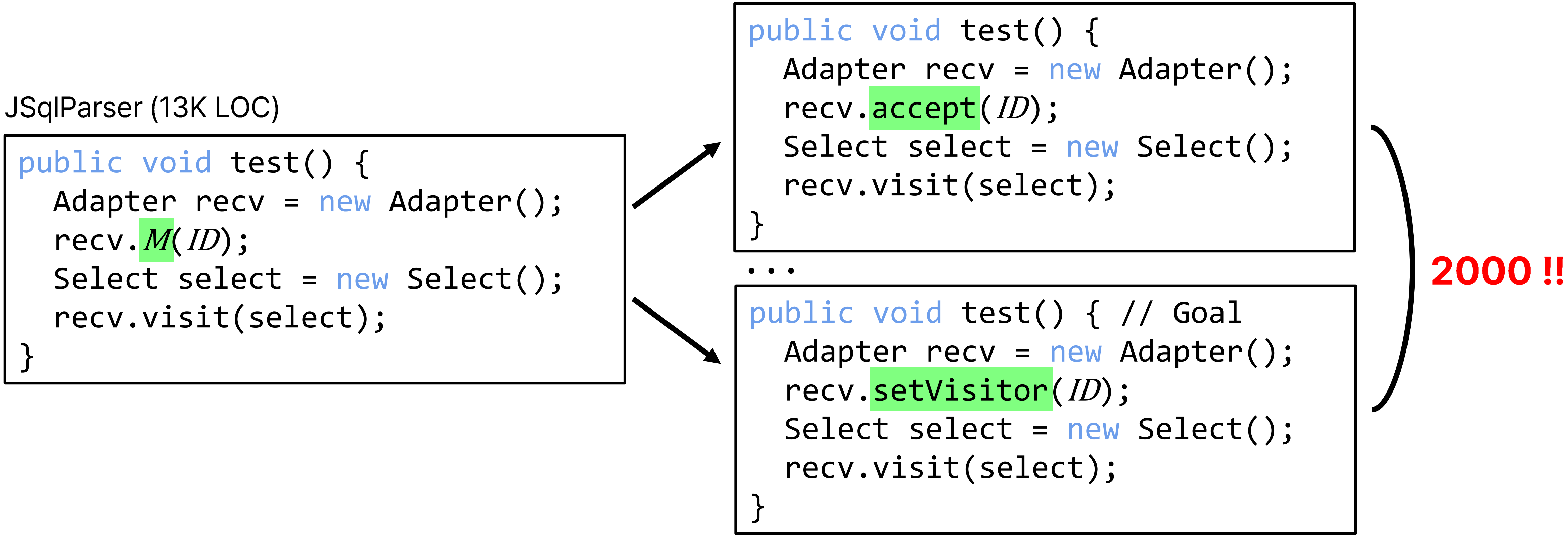
...

```
public void test() { // Goal
  Adapter recv = new Adapter();
  recv.setVisitor(ID);
  Select select = new Select();
  recv.visit(select);
}
```

Problem

- **Exponential growth of partial test cases** makes simple synthesis ineffective.
  - Only **one** step can generate thousands of partial test cases.

* JSqlParser (13K LOC)



```
public void test() {
  Adapter recv = new Adapter();
  recv.M(ID);
  Select select = new Select();
  recv.visit(select);
}
```

```
public void test() {
  Adapter recv = new Adapter();
  recv.accept(ID);
  Select select = new Select();
  recv.visit(select);
}
```

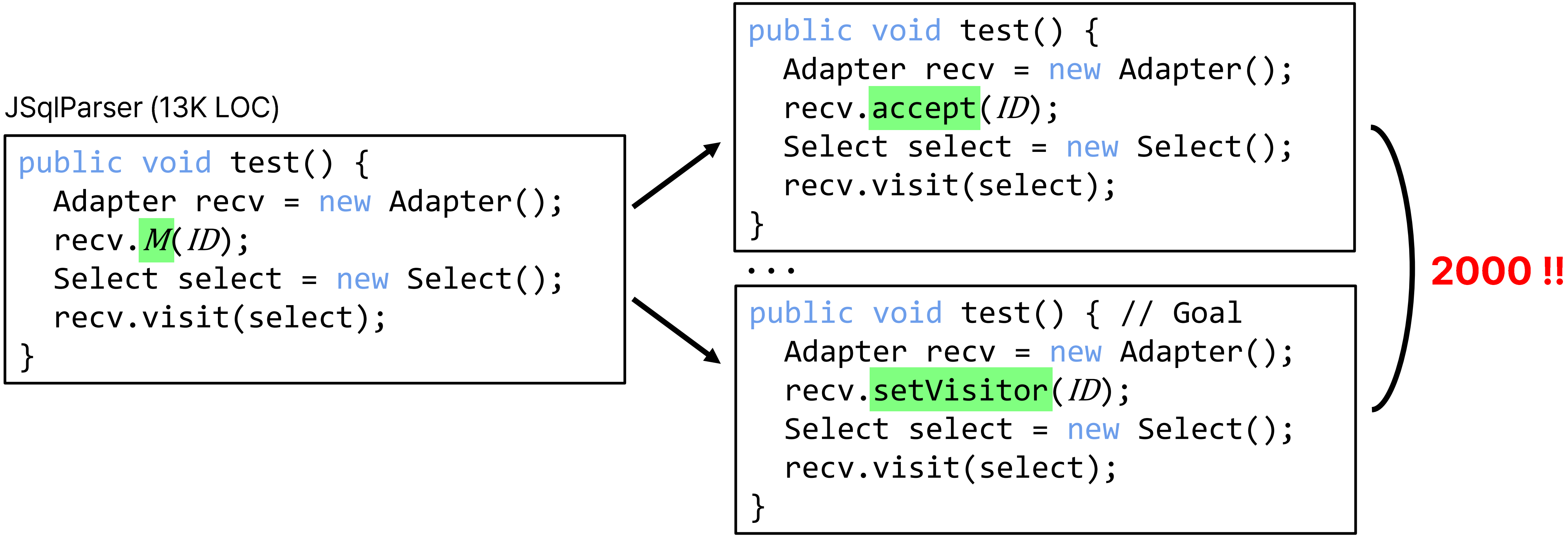
...

```
public void test() { // Goal
  Adapter recv = new Adapter();
  recv.setVisitor(ID);
  Select select = new Select();
  recv.visit(select);
}
```

Problem

- **Exponential growth of partial test cases** makes simple synthesis ineffective.
  - Only **one** step can generate thousands of partial test cases.
  - Other tools (Randoop, EvoSuite) found the bug in **only 0 to 1** out of 10 runs.

* JSqlParser (13K LOC)



```
public void test() {
  Adapter recv = new Adapter();
  recv.M(ID);
  Select select = new Select();
  recv.visit(select);
}
```

```
public void test() {
  Adapter recv = new Adapter();
  recv.accept(ID);
  Select select = new Select();
  recv.visit(select);
}
```

...

```
public void test() { // Goal
  Adapter recv = new Adapter();
  recv.setVisitor(ID);
  Select select = new Select();
  recv.visit(select);
}
```

2000 !!

Our Solution

Our Solution

Program with
a specific location

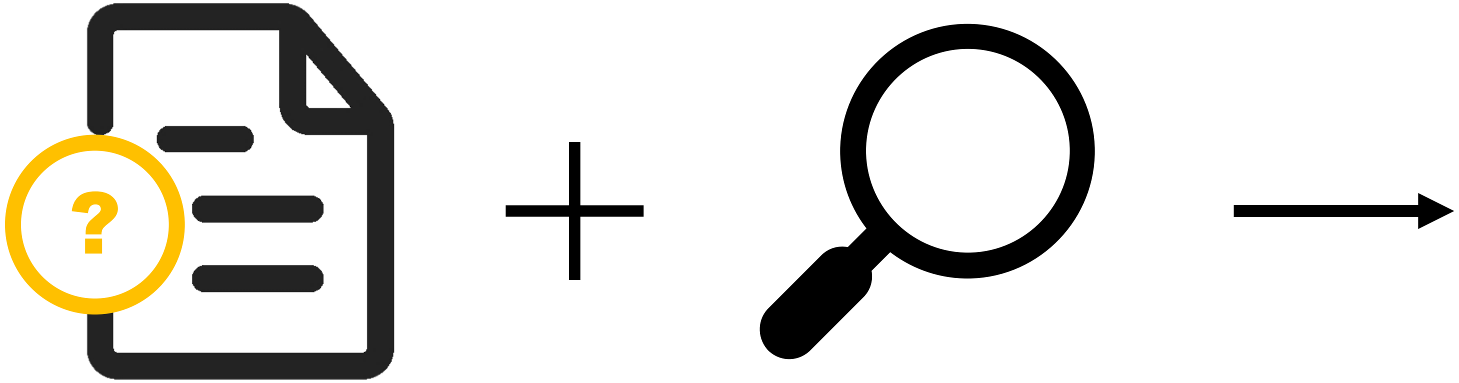
Our Solution

```
public void test() {
  Adapter recv = new Adapter();
  Visitor visitor = new Visitor();
  recv.setVisitor(visitor);
  Select select = new Select();
  recv.visit(select); // NPE
}
```

Program with a specific location

Targeted Unit Test

Our Solution



```
public void test() {
  Adapter recv = new Adapter();
  Visitor visitor = new Visitor();
  recv.setVisitor(visitor);
  Select select = new Select();
  recv.visit(select); // NPE
}
```

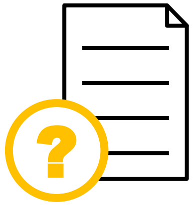
Program with a specific location

Program Synthesis Guided by Static Analyzer

Targeted Unit Test

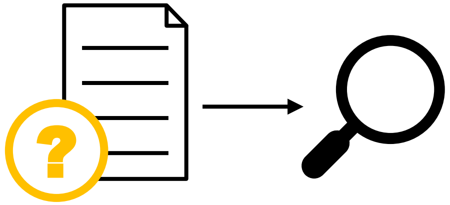
UnitCon System

UnitCon System



Program with
a target location

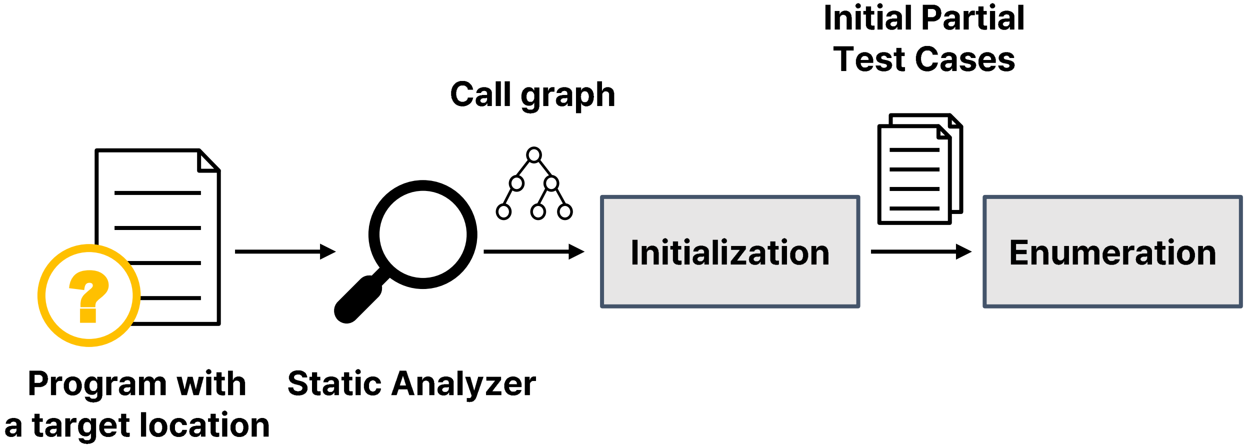
UnitCon System



Program with
a target location

Static Analyzer

UnitCon System



Initial Partial Test Cases

Call graph

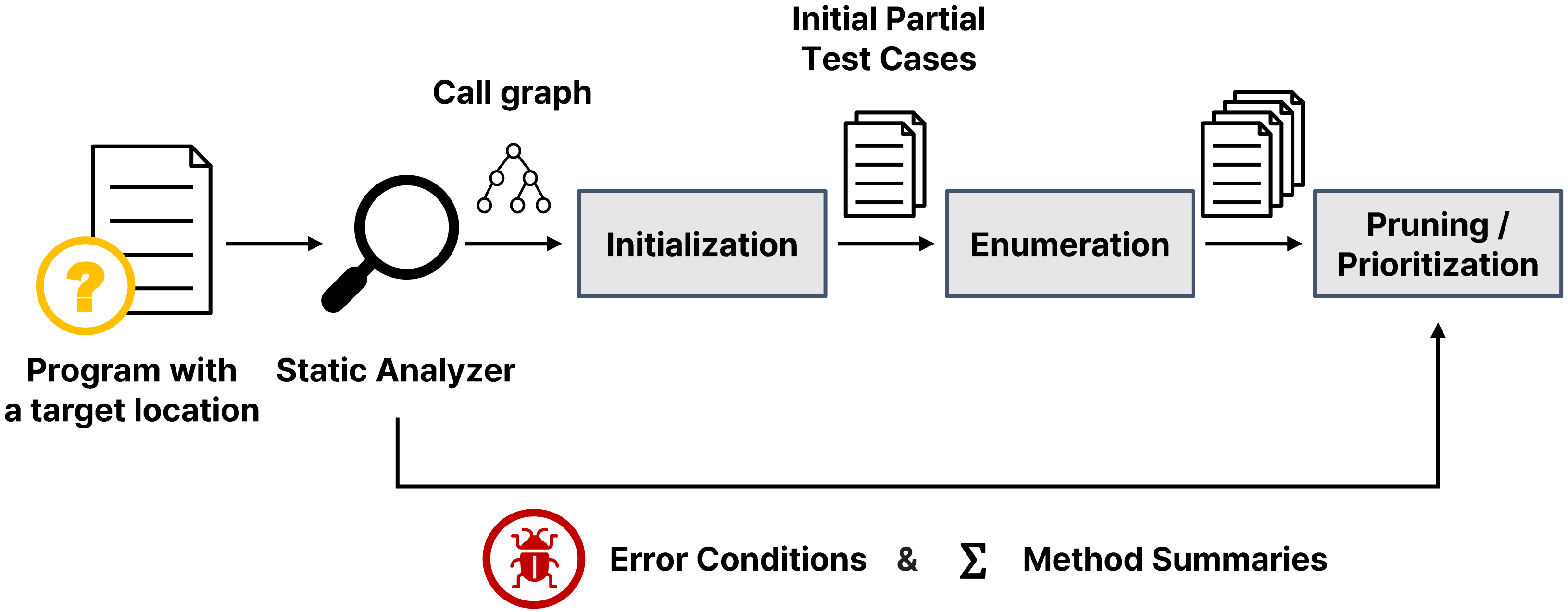
Initialization

Enumeration

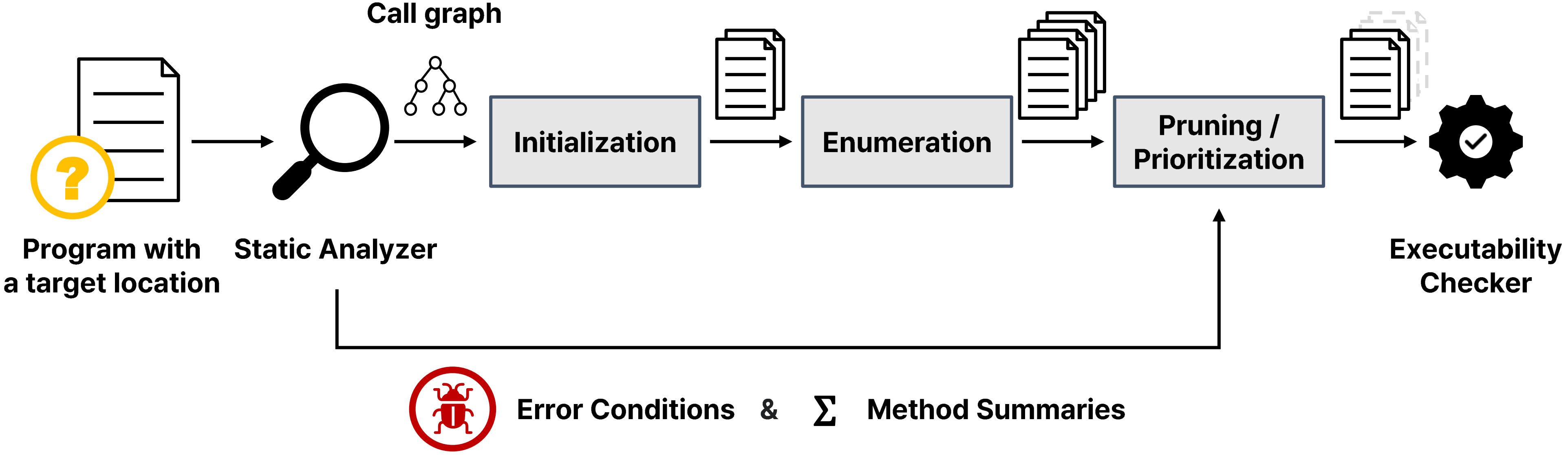
Program with a target location

Static Analyzer

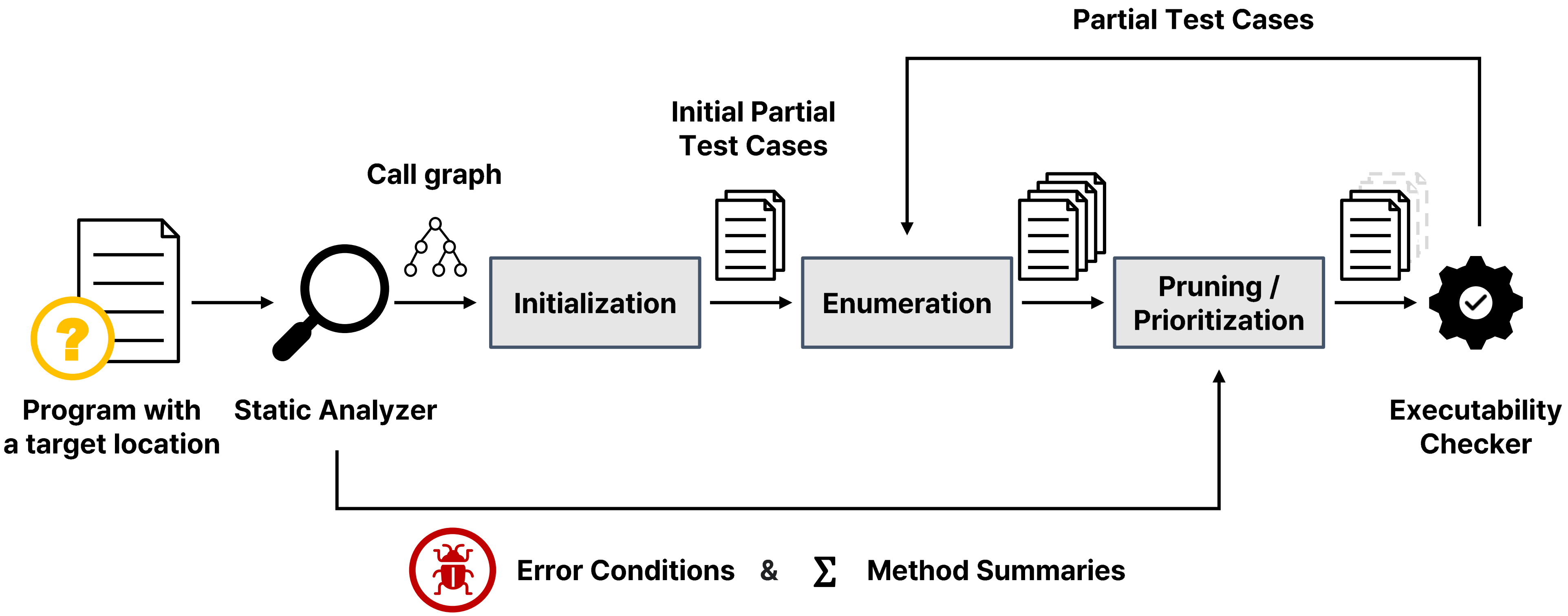
UnitCon System



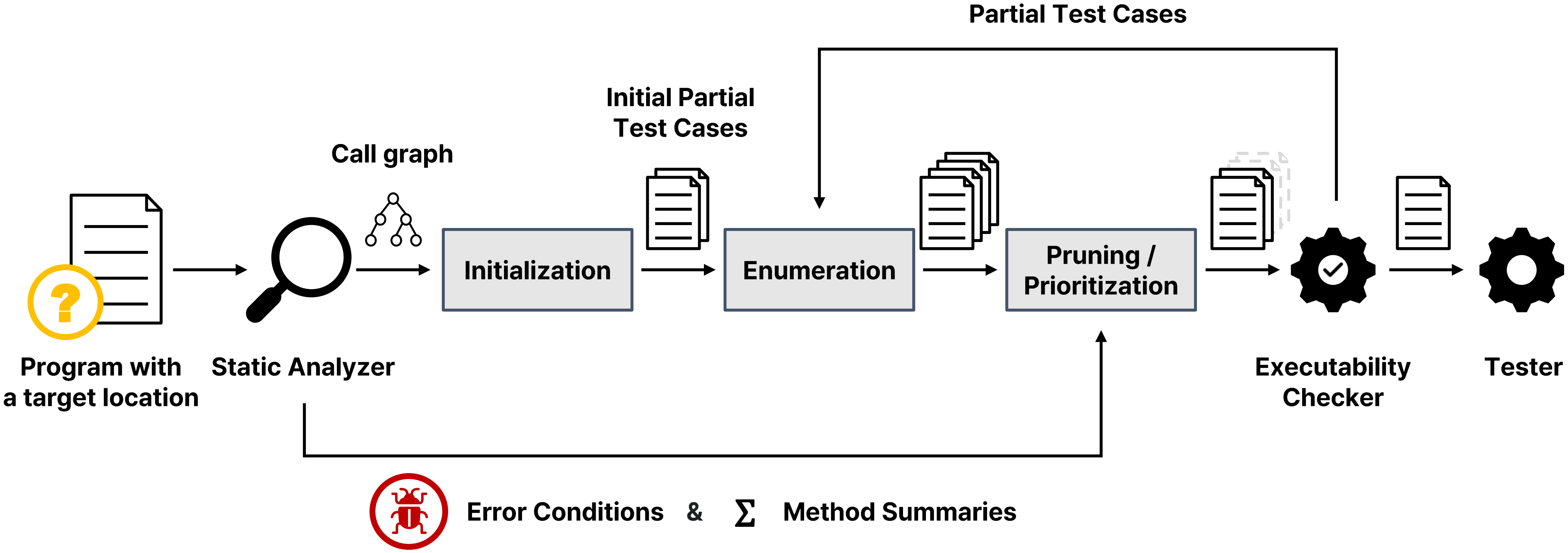
UnitCon System



UnitCon System



UnitCon System



Partial Test Cases

Initial Partial Test Cases

Call graph

Program with a target location

Static Analyzer

Initialization

Enumeration

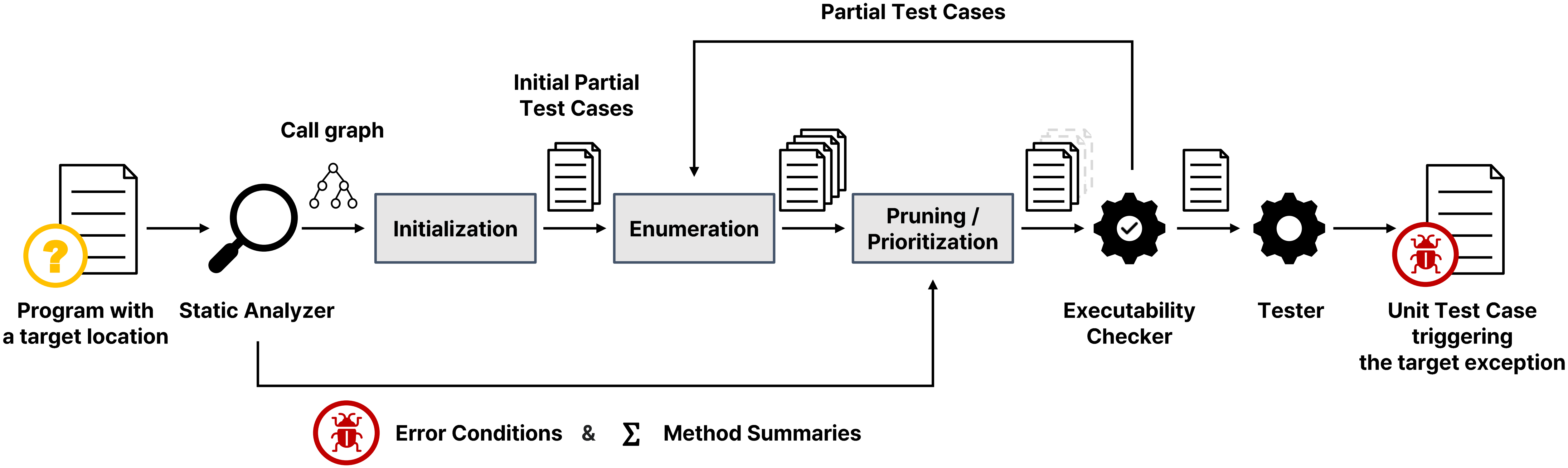
Pruning / Prioritization

Executability Checker

Tester

Error Conditions & Σ Method Summaries

UnitCon System


Partial Test Cases
Initial Partial Test Cases
Call graph
Initialization
Enumeration
Pruning / Prioritization
Program with a target location
Static Analyzer
Executability Checker
Tester
Unit Test Case triggering the target exception
Error Conditions & ∑ Method Summaries

Contributions

Contributions

First targeted
unit test synthesizer

Contributions

First targeted
unit test synthesizer

Up to **3.6 x** better performance
compared to baselines

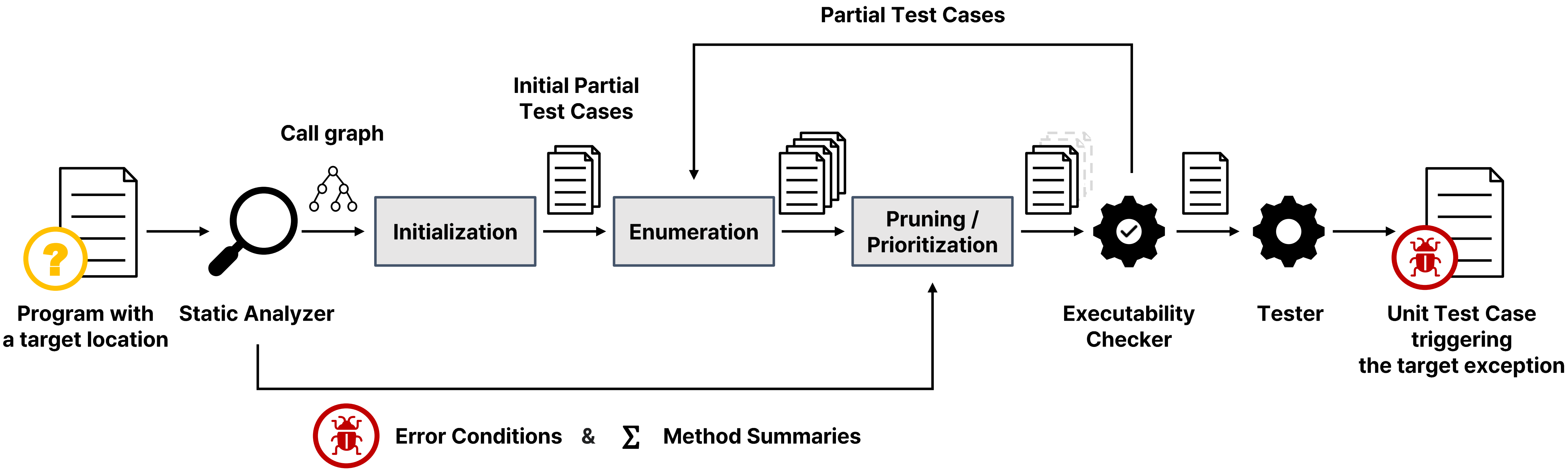
Contributions

First targeted
unit test synthesizer

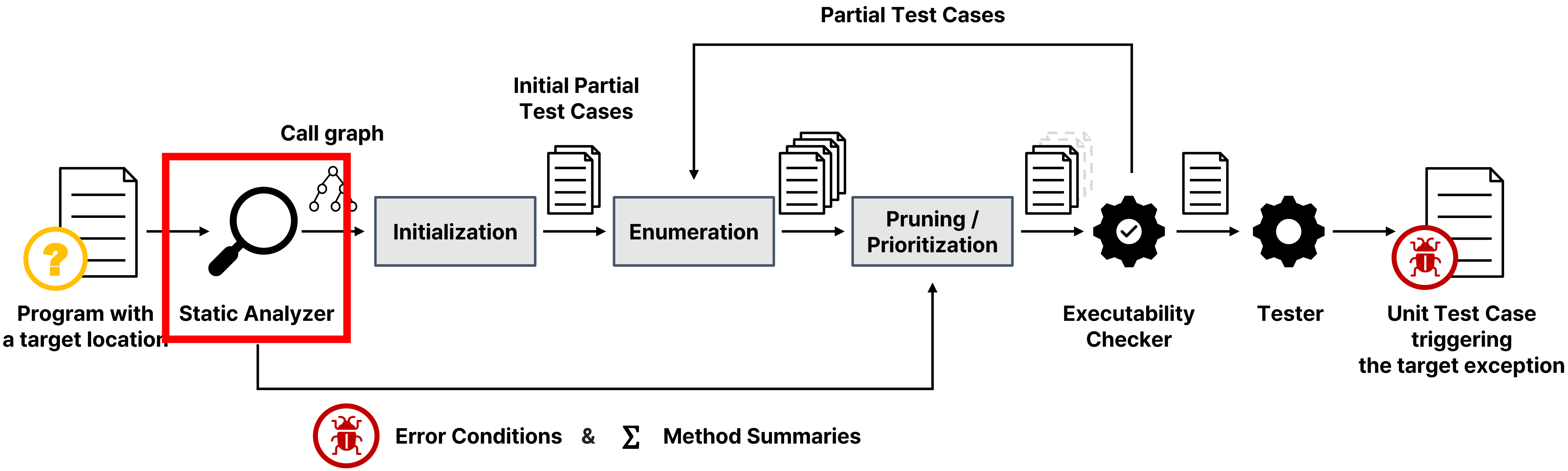
Up to **3.6 x** better performance
compared to baselines

Found **21** new bugs
in open-source programs

UnitCon System



UnitCon System


Partial Test Cases
Initial Partial Test Cases
Call graph
Program with a target location
Static Analyzer
Initialization
Enumeration
Pruning / Prioritization
Executability Checker
Tester
Unit Test Case triggering the target exception
Error Conditions & ∑ Method Summaries

Example

```java
public class Adapter {
  private Visitor visitor;

  public void setVisitor(Visitor visitor) {
    this.visitor = visitor;
  }

  public void visit(Select select) {
    if (visitor != null) {
      ItemsList itemsList = select.getItemsList();
      for (Item item: itemsList) { // Target location
        ...
}

public class Select {
  private List <Item> itemsList;

  public List <Item> getItemsList() { return itemsList; }
}

public class Merge {
  private Select usingSelect;

  public Select getUsingSelect() { return usingSelect; }
}
```

Example

```java
public class Adapter {
  private Visitor visitor;

  public void setVisitor(Visitor visitor) {
    this.visitor = visitor;
  }

  public void visit(Select select) {
    if (visitor != null) {
      ItemsList itemsList = select.getItemsList();
      for (Item item: itemsList) { // Target location
        ...
}

public class Select {
  private List <Item> itemsList;

  public List <Item> getItemsList() { return itemsList; }
}

public class Merge {
  private Select usingSelect;

  public Select getUsingSelect() { return usingSelect; }
}
```

Example

```java
public class Adapter {
  private Visitor visitor;

  public void setVisitor(Visitor visitor) {
    this.visitor = visitor;
  }

  public void visit(Select select) {
    if (visitor != null) {
      ItemsList itemsList = select.getItemsList();
      for (Item item: itemsList) { // Target location
        ...
}

public class Select {
  private List <Item> itemsList;

  public List <Item> getItemsList() { return itemsList; }
}

public class Merge {
  private Select usingSelect;

  public Select getUsingSelect() { return usingSelect; }
}
```

Example

```
public class Adapter {
  private Visitor visitor;

  public void setVisitor(Visitor visitor) {
    this.visitor = visitor;
  }

  public void visit(Select select) {
    if (visitor != null) {
      ItemsList itemsList = select.getItemsList();
      for (Item item: itemsList) { // Target location
        ...
}

public class Select {
  private List <Item> itemsList;

  public List <Item> getItemsList() { return itemsList; }
}

public class Merge {
  private Select usingSelect;

  public Select getUsingSelect() { return usingSelect; }
}
```

Example

```
public class Adapter {
  private Visitor visitor;

  public void setVisitor(Visitor visitor) {
    this.visitor = visitor;
  }

  public void visit(Select select) {
    if (visitor != null) {
      ItemsList itemsList = select.getItemsList();
      for (Item item: itemsList) { // Target location
        ...
}

public class Select {
  private List <Item> itemsList;

  public List <Item> getItemsList() { return itemsList; }
}

public class Merge {
  private Select usingSelect;

  public Select getUsingSelect() { return usingSelect; }
}
```

Example

```java
public class Adapter {
  private Visitor visitor;

  public void setVisitor(Visitor visitor) {
    this.visitor = visitor;
  }

  public void visit(Select select) {
    if (visitor != null) {
      ItemsList itemsList = select.getItemsList();
      for (Item item: itemsList) { // Target location
        ...
}

public class Select {
  private List <Item> itemsList;

  public List <Item> getItemsList() { return itemsList; }
}

public class Merge {
  private Select usingSelect;

  public Select getUsingSelect() { return usingSelect; }
}
```

Error Method

```
Adapter.visit(Select)
```

Example

```java
public class Adapter {
  private Visitor visitor;

  public void setVisitor(Visitor visitor) {
    this.visitor = visitor;
  }

  public void visit(Select select) {
    if (visitor != null) {
      ItemsList itemsList = select.getItemsList();
      for (Item item: itemsList) { // Target location
        ...
}

public class Select {
  private List <Item> itemsList;

  public List <Item> getItemsList() { return itemsList; }
}

public class Merge {
  private Select usingSelect;

  public Select getUsingSelect() { return usingSelect; }
}
```

Error Method

```
Adapter.visit(Select)
```

Error Conditions

Example

```java
public class Adapter {
  private Visitor visitor;

  public void setVisitor(Visitor visitor) {
    this.visitor = visitor;
  }

  public void visit(Select select) {
    if (visitor != null) {
      ItemsList itemsList = select.getItemsList();
      for (Item item: itemsList) { // Target location
        ...
}

public class Select {
  private List <Item> itemsList;

  public List <Item> getItemsList() { return itemsList; }
}

public class Merge {
  private Select usingSelect;

  public Select getUsingSelect() { return usingSelect; }
}
```

Error Method

```
Adapter.visit(Select)
```

Error Conditions

Example

```java
public class Adapter {
  private Visitor visitor;

  public void setVisitor(Visitor visitor) {
    this.visitor = visitor;
  }

  public void visit(Select select) {
    if (visitor != null) {
      ItemsList itemsList = select.getItemsList();
      for (Item item: itemsList) { // Target location
        ...
}

public class Select {
  private List <Item> itemsList;

  public List <Item> getItemsList() { return itemsList; }
}

public class Merge {
  private Select usingSelect;

  public Select getUsingSelect() { return usingSelect; }
}
```

```
select.getItemsList() == null
```

Error Method

```
Adapter.visit(Select)
```

Error Conditions

Example

```
public class Adapter {
  private Visitor visitor;

  public void setVisitor(Visitor visitor) {
    this.visitor = visitor;
  }

  public void visit(Select select) {
    if (visitor != null) {
      ItemsList itemsList = select.getItemsList();
      for (Item item: itemsList) { // Target location
        ...
}

public class Select {
  private List <Item> itemsList;

  public List <Item> getItemsList() { return itemsList; }
}

public class Merge {
  private Select usingSelect;

  public Select getUsingSelect() { return usingSelect; }
}
```

```
select.getItemsList() == null
```

Error Method

```
Adapter.visit(Select)
```

Error Conditions

Example

```java
public class Adapter {
  private Visitor visitor;

  public void setVisitor(Visitor visitor) {
    this.visitor = visitor;
  }

  public void visit(Select select) {
    if (visitor != null) {
      ItemsList itemsList = select.getItemsList();
      for (Item item: itemsList) { // Target location
        ...
}

public class Select {
  private List <Item> itemsList;

  public List <Item> getItemsList() { return itemsList; }
}

public class
  private Select usingSelect;

  public Select getUsingSelect() { return usingSelect; }
}
```

`select.getItemsList() == null`

field `itemsList` of `select == null`

Error Method

`Adapter.visit(Select)`

Error Conditions

Example

```
public class Adapter {
  private Visitor visitor;

  public void setVisitor(Visitor visitor) {
    this.visitor = visitor;
  }

  public void visit(Select select) {
    if (visitor != null) {
      ItemsList itemsList = select.getItemsList();
      for (Item item: itemsList) { // Target location
        ...
}

public class Select {
  private List <Item> itemsList;

  public List <Item> getItemsList() { return itemsList; }
}

public class
  private Select usingSelect;

  public Select getUsingSelect() { return usingSelect; }
}
```

select.getItemsList() == null

field itemsList of select == null

Error Method

Adapter.visit(Select)

Error Conditions

Object	Condition
select.itemsList	== null

Example

```
public class Adapter {
  private Visitor visitor;

  public void setVisitor(Visitor visitor) {
    this.visitor = visitor;
  }

  public void visit(Select select) {
    if (visitor != null) {
      ItemsList itemsList = select.getItemsList();
      for (Item item: itemsList) { // Target location
        ...
}

public class Select {
  private List <Item> itemsList;

  public List <Item> getItemsList() { return itemsList; }
}

public class Merge {
  private Select usingSelect;

  public Select getUsingSelect() { return usingSelect; }
}
```

Error Method

```
Adapter.visit(Select)
```

Error Conditions

Object	Condition
select.itemsList	== null

Example

```java
public class Adapter {
  private Visitor visitor;

  public void setVisitor(Visitor visitor) {
    this.visitor = visitor;
  }

  public void visit(Select select) {
    if (visitor != null) {
      ItemsList itemsList = select.getItemsList();
      for (Item item: itemsList) { // Target location
        ...
}

public class Select {
  private List <Item> itemsList;

  public List <Item> getItemsList() { return itemsList; }
}

public class Merge {
  private Select usingSelect;

  public Select getUsingSelect() { return usingSelect; }
}
```

Error Method

`Adapter.visit(Select)`

Error Conditions

Object	Condition
select.itemsList	== null

Example

```java
public class Adapter {
  private Visitor visitor;

  public void setVisitor(Visitor visitor) {
    this.visitor = visitor;
  }

  public void visit(Select select) {
    if (visitor != null) {
      ItemsList itemsList = select.getItemsList();
      for (Item item: itemsList) { // Target location
        ...
}

public class Select {
  private List <Item> itemsList;

  public List <Item> getItemsList() { return itemsList; }
}

public class Merge {
  private Select usingSelect;

  public Select getUsingSelect() { return usingSelect; }
}
```

Error Method

```
Adapter.visit(Select)
```

Error Conditions

Object	Condition
select.itemsList	== null
this.visitor	!= null

Example

```java
public class Adapter {
  private Visitor visitor;

  public void setVisitor(Visitor visitor) {
    this.visitor = visitor;
  }

  public void visit(Select select) {
    if (visitor != null) {
      ItemsList itemsList = select.getItemsList();
      for (Item item: itemsList) { // Target location
        ...
}

public class Select {
  private List <Item> itemsList;

  public List <Item> getItemsList() { return itemsList; }
}

public class Merge {
  private Select usingSelect;

  public Select getUsingSelect() { return usingSelect; }
}
```

Error Method

```
Adapter.visit(Select)
```

Error Conditions

Object	Condition
`select.itemsList`	`== null`
`this.visitor`	`!= null`

Example

```java
public class Adapter {
  private Visitor visitor;

  public void setVisitor(Visitor visitor) {
    this.visitor = visitor;
  }

  public void visit(Select select) {
    if (visitor != null) {
      ItemsList itemsList = select.getItemsList();
      for (Item item: itemsList) { // Target location
        ...
}

public class Select {
  private List <Item> itemsList;

  public List <Item> getItemsList() { return itemsList; }
}

public class Merge {
  private Select usingSelect;

  public Select getUsingSelect() { return usingSelect; }
}
```

Error Method

`Adapter.visit(Select)`

Error Conditions

Object	Condition
select.itemsList	== null
this.visitor	!= null
select	!= null

Example

```java
public class Adapter {
  private Visitor visitor;

  public void setVisitor(Visitor visitor) {
    this.visitor = visitor;
  }

  public void visit(Select select) {
    if (visitor != null) {
      ItemsList itemsList = select.getItemsList();
      for (Item item: itemsList) { // Target location
        ...
}

public class Select {
  private List <Item> itemsList;

  public List <Item> getItemsList() { return itemsList; }
}

public class Merge {
  private Select usingSelect;

  public Select getUsingSelect() { return usingSelect; }
}
```

Error Method

```
Adapter.visit(Select)
```

Error Conditions

Object	Condition
select.itemsList	== null
this.visitor	!= null
select	!= null

Example

```java
public class Adapter {
  private Visitor visitor;

  public void setVisitor(Visitor visitor) {
    this.visitor = visitor;
  }

  public void visit(Select select) {
    if (visitor != null) {
      ItemsList itemsList = select.getItemsList();
      for (Item item: itemsList) { // Target location
        ...
}

public class Select {
  private List <Item> itemsList;

  public List <Item> getItemsList() { return itemsList; }
}

public class Merge {
  private Select usingSelect;

  public Select getUsingSelect() { return usingSelect; }
}
```

Error Method

`Adapter.visit(Select)`

Error Conditions

Object	Condition
select.itemsList	== null
this.visitor	!= null
select	!= null

Example

```
public class Adapter {
  private Visitor visitor;

  public void setVisitor(Visitor visitor) {
    this.visitor = visitor;
  }

  public void visit(Select select) {
    if (visitor != null) {
      ItemsList itemsList = select.getItemsList();
      for (Item item: itemsList) { // Target location
        ...
}

public class Select {
  private List <Item> itemsList;

  public List <Item> getItemsList() { return itemsList; }
}

public class Merge {
  private Select usingSelect;

  public Select getUsingSelect() { return usingSelect; }
}
```

Error Method

`Adapter.visit(Select)`

Error Conditions

Object	Condition
select.itemsList	== null
this.visitor	!= null
select	!= null

Method Summaries

Method	Memory

Example

```java
public class Adapter {
  private Visitor visitor;

  public void setVisitor(Visitor visitor) {
    this.visitor = visitor;
  }

  public void visit(Select select) {
    if (visitor != null) {
      ItemsList itemsList = select.getItemsList();
      for (Item item: itemsList) { // Target location
        ...
}

public class Select {
  private List <Item> itemsList;

  public List <Item> getItemsList() { return itemsList; }
}

public class Merge {
  private Select usingSelect;

  public Select getUsingSelect() { return usingSelect; }
}
```

Error Method

`Adapter.visit(Select)`

Error Conditions

Object	Condition
select.itemsList	== null
this.visitor	!= null
select	!= null

Method Summaries

Method	Memory
Merge	{ usingSelect ↦ null }

Example

```java
public class Adapter {
  private Visitor visitor;

  public void setVisitor(Visitor visitor) {
    this.visitor = visitor;
  }

  public void visit(Select select) {
    if (visitor != null) {
      ItemsList itemsList = select.getItemsList();
      for (Item item: itemsList) { // Target location
        ...
}

public class Select {
  private List <Item> itemsList;

  public List <Item> getItemsList() { return itemsList; }
}

public class Merge {
  private Select usingSelect;

  public Select getUsingSelect() { return usingSelect; }
}
```

Error Method

`Adapter.visit(Select)`

Error Conditions

Object	Condition
select.itemsList	== null
this.visitor	!= null
select	!= null

Method Summaries

Method	Memory
Merge	{ usingSelect ↦ null }
getUsingSelect	{ ret ↦ usingSelect }

Example

```java
public class Adapter {
  private Visitor visitor;

  public void setVisitor(Visitor visitor) {
    this.visitor = visitor;
  }

  public void visit(Select select) {
    if (visitor != null) {
      ItemsList itemsList = select.getItemsList();
      for (Item item: itemsList) { // Target location
        ...
}

public class Select {
  private List <Item> itemsList;

  public List <Item> getItemsList() { return itemsList; }
}

public class Merge {
  private Select usingSelect;

  public Select getUsingSelect() { return usingSelect; }
}
```

Error Method

`Adapter.visit(Select)`

Error Conditions

Object	Condition
select.itemsList	== null
this.visitor	!= null
select	!= null

Method Summaries

Method	Memory
Merge	{ usingSelect ↦ null }
getUsingSelect	{ ret ↦ usingSelect }
...	...

Example

```java
public class Adapter {
  private Visitor visitor;

  public void setVisitor(Visitor visitor) {
    this.visitor = visitor;
  }

  public void visit(Select select) {
    if (visitor != null) {
      ItemsList itemsList = select.getItemsList();
      for (Item item: itemsList) { // Target location
        ...
}

public class Select {
  private List <Item> itemsList;

  public List <Item> getItemsList() { return itemsList; }
}

public class Merge {
  private Select usingSelect;

  public Select getUsingSelect() { return usingSelect; }
}
```

Error Method

`Adapter.visit(Select)`

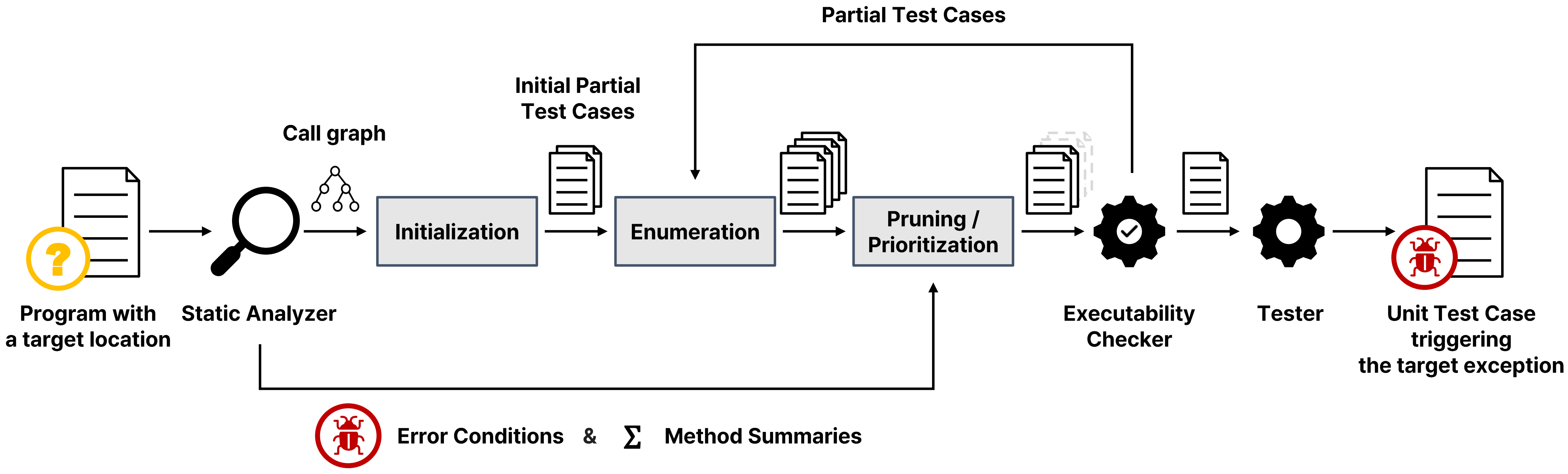
Error Conditions

Object	Condition
select.itemsList	== null
this.visitor	!= null
select	!= null

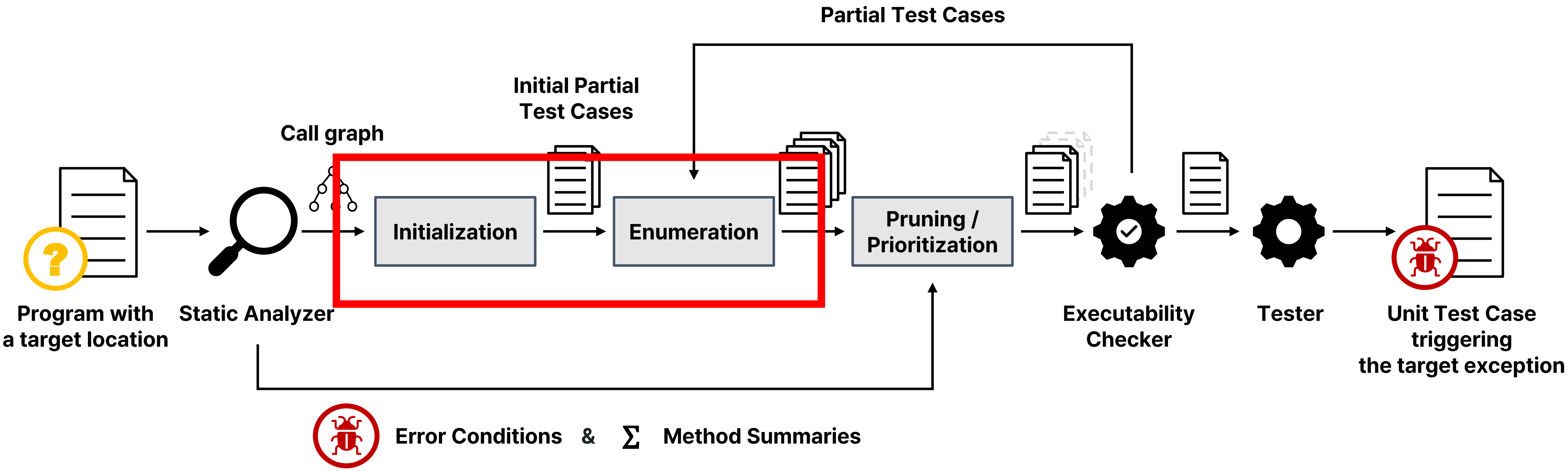
Method Summaries

Method	Memory
Merge	{ usingSelect ↦ null }
getUsingSelect	{ ret ↦ usingSelect }
...	...

UnitCon System



UnitCon System



Top-down enumerative search

Top-down enumerative search

- Expand test cases top-down based on defined rules.

Top-down enumerative search

- Expand test cases top-down based on defined rules.

Rule form: `Before` → `After`

Top-down enumerative search

- Expand test cases top-down based on defined rules.

Rule form: `Before` → `After`

Error Method

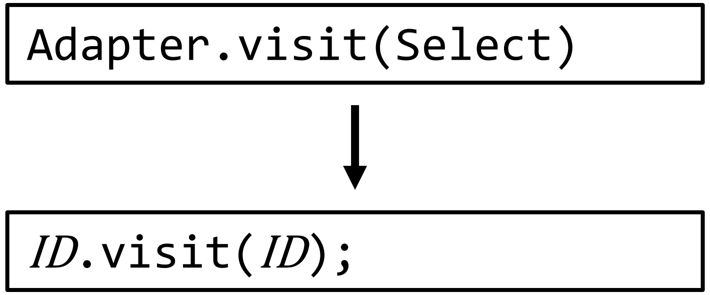
```
Adapter.visit(Select)
```

Top-down enumerative search

- Expand test cases top-down based on defined rules.

Rule form: `Before` → `After`

Error Method



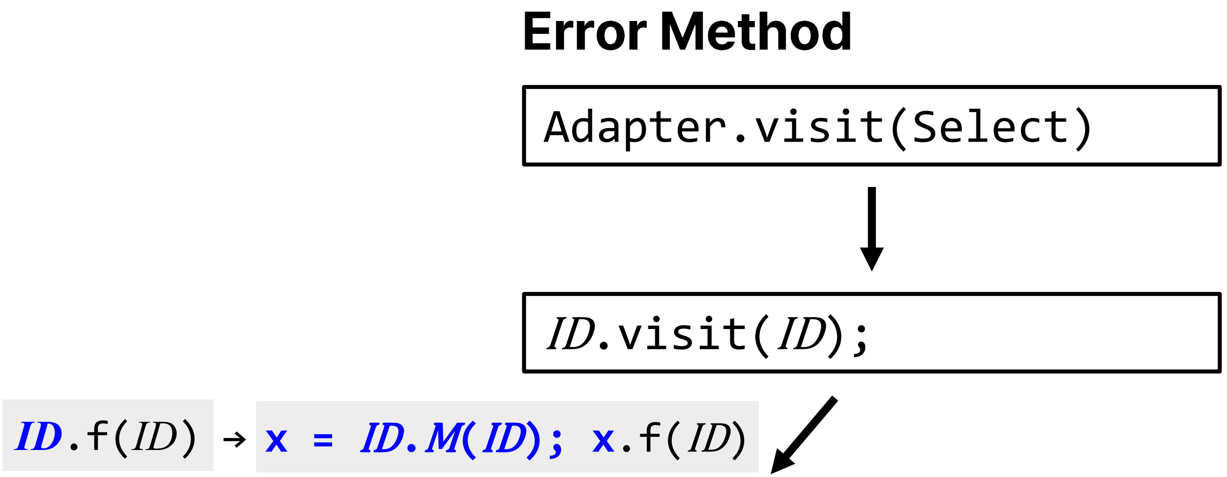
```
Adapter.visit(Select)
```

ID.visit(*ID*);

Top-down enumerative search

- Expand test cases top-down based on defined rules.

Rule form: `Before` → `After`



Error Method

`Adapter.visit(Select)`

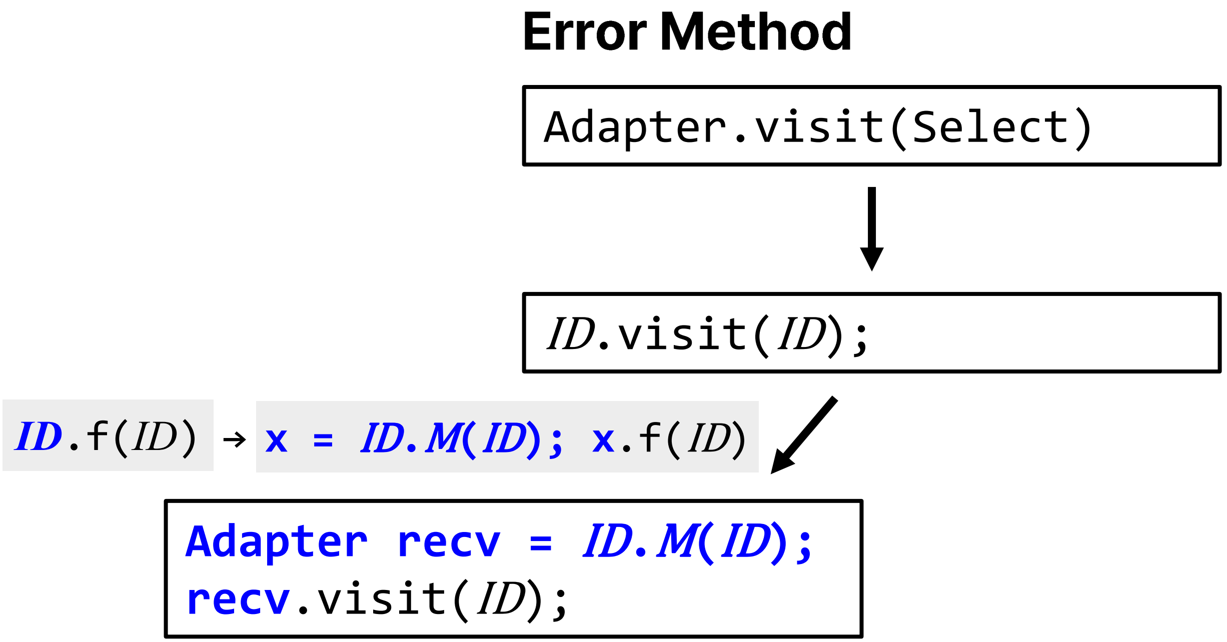
ID`.visit(`*ID*`);`

ID`.f(`*ID*`)` → `x = `*ID*`.`*M*`(`*ID*`); x.f(`*ID*`)`

Top-down enumerative search

- Expand test cases top-down based on defined rules.

Rule form: `Before` → `After`



Error Method

```
Adapter.visit(Select)
```

```
ID.visit(ID);
```

`ID.f(ID)` → `x = ID.M(ID); x.f(ID)`

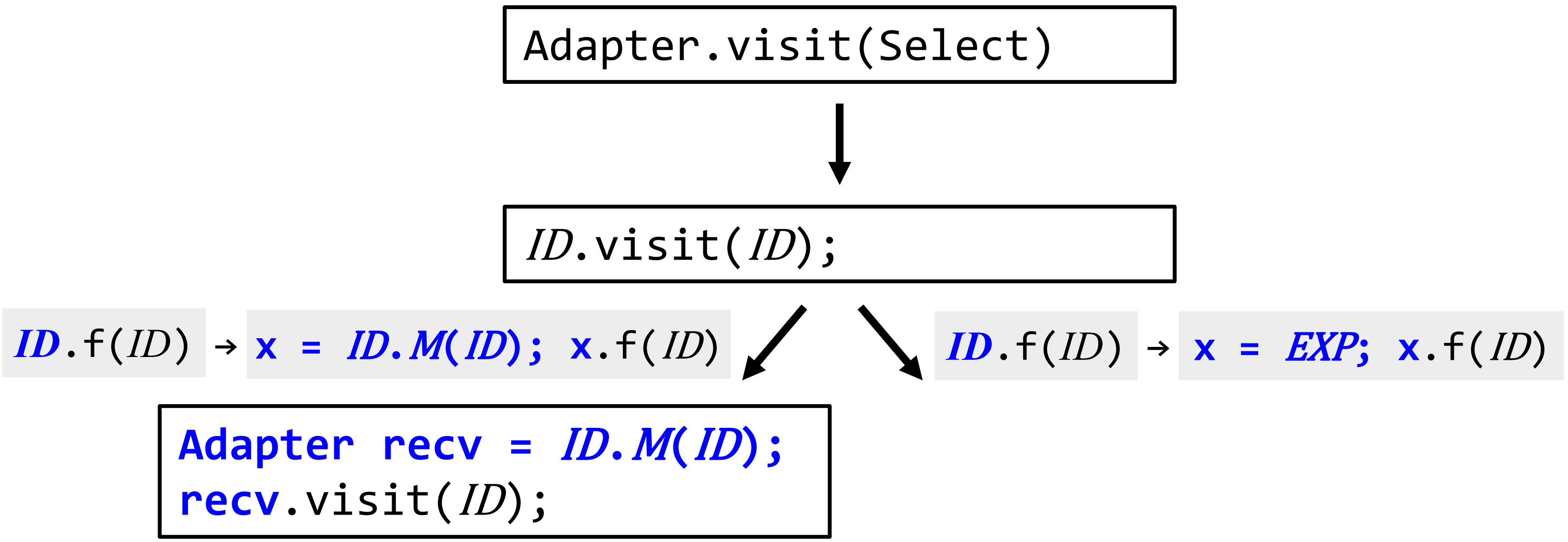
```
Adapter recv = ID.M(ID);
recv.visit(ID);
```

Top-down enumerative search

- Expand test cases top-down based on defined rules.

Rule form: `Before` → `After`

Error Method



`Adapter.visit(Select)`

`ID``.visit(`*`ID`*`);`

`ID``.f(`*`ID`*`)` → `x = `*`ID.M(ID)`*`; x.f(`*`ID`*`)`

`ID``.f(`*`ID`*`)` → `x = `*`EXP`*`; x.f(`*`ID`*`)`

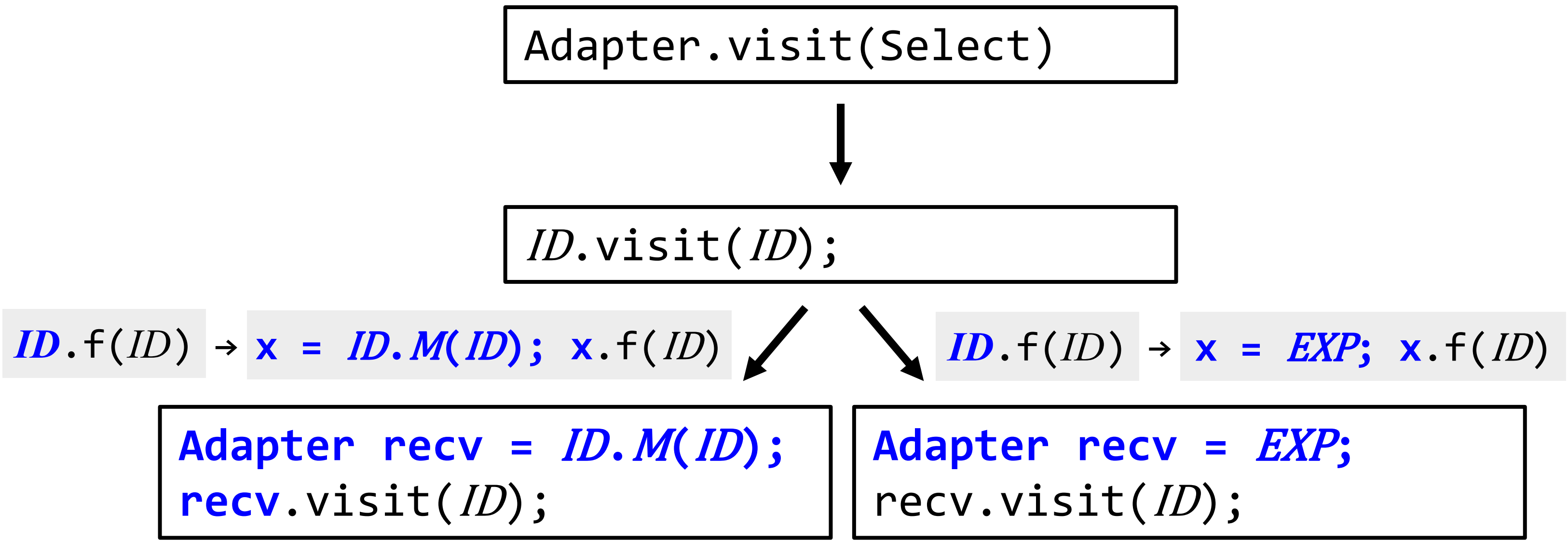
```
Adapter recv = ID.M(ID);
recv.visit(ID);
```

Top-down enumerative search

- Expand test cases top-down based on defined rules.

Rule form: `Before` → `After`

Error Method



```
Adapter.visit(Select)
```

```
ID.visit(ID);
```

`ID.f(ID)` → `x = ID.M(ID); x.f(ID)`

`ID.f(ID)` → `x = EXP; x.f(ID)`

```
Adapter recv = ID.M(ID);
recv.visit(ID);
```

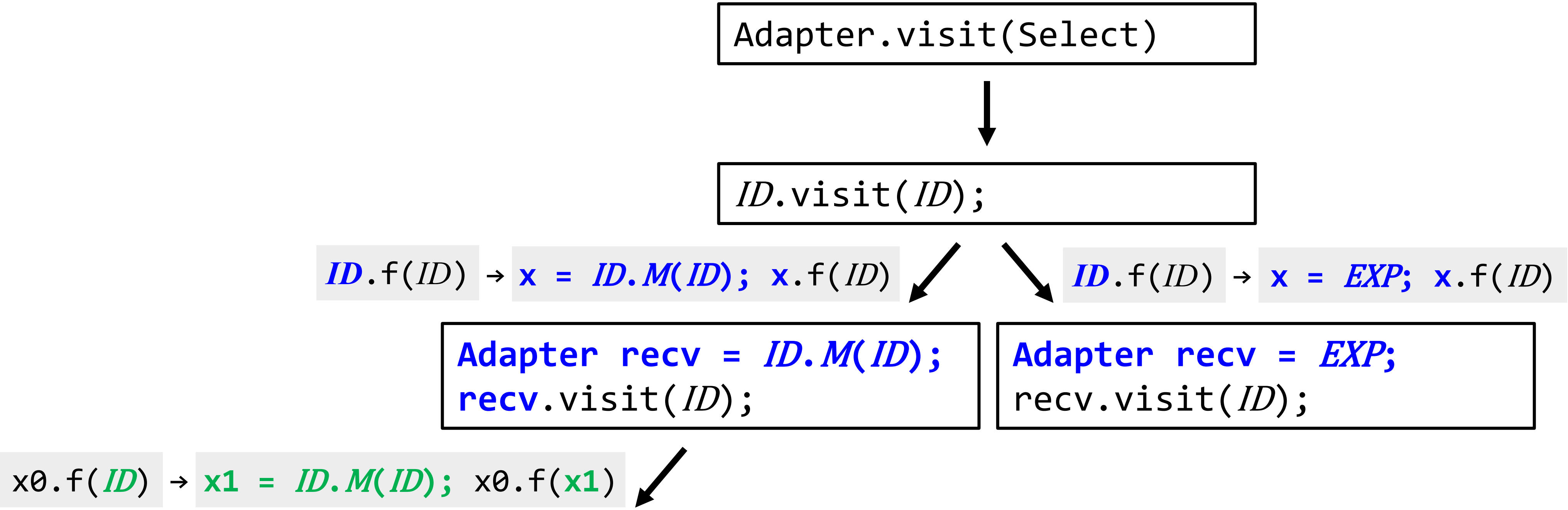
```
Adapter recv = EXP;
recv.visit(ID);
```

Top-down enumerative search

- Expand test cases top-down based on defined rules.

Rule form: **Before** → **After**

Error Method



Adapter.visit(Select)

ID.visit(*ID*);

ID.f(*ID*) → **x = *ID*.*M*(*ID*); x**.f(*ID*)

ID.f(*ID*) → **x = *EXP*; x**.f(*ID*)

Adapter recv = *ID*.*M*(*ID*);
recv.visit(*ID*);

Adapter recv = *EXP*;
recv.visit(*ID*);

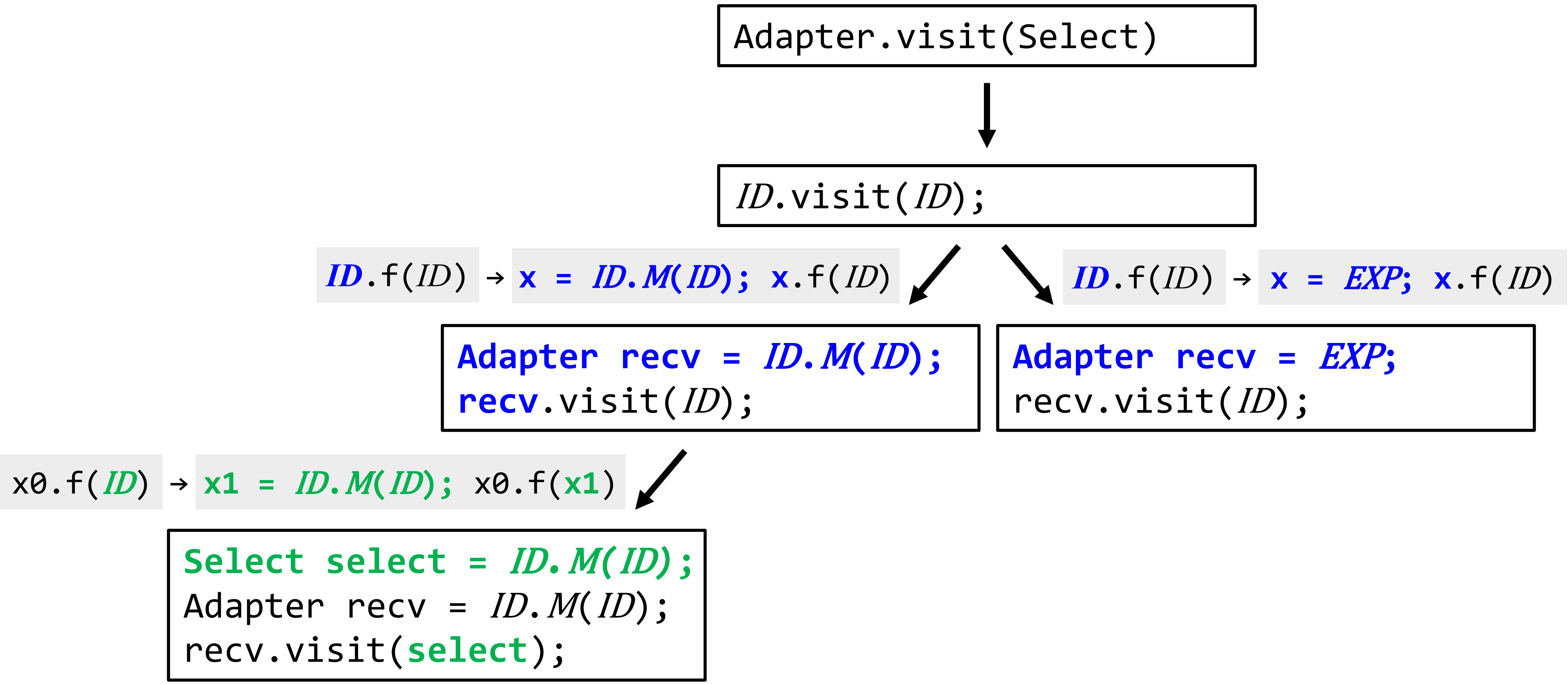
x0.f(***ID***) → **x1 = *ID*.*M*(*ID*);** x0.f(**x1**)

Top-down enumerative search

- Expand test cases top-down based on defined rules.

Rule form: `Before` → `After`

Error Method



```
Adapter.visit(Select)
```

```
ID.visit(ID);
```

`ID.f(ID)` → `x = ID.M(ID); x.f(ID)`

```
Adapter recv = ID.M(ID);
recv.visit(ID);
```

`ID.f(ID)` → `x = EXP; x.f(ID)`

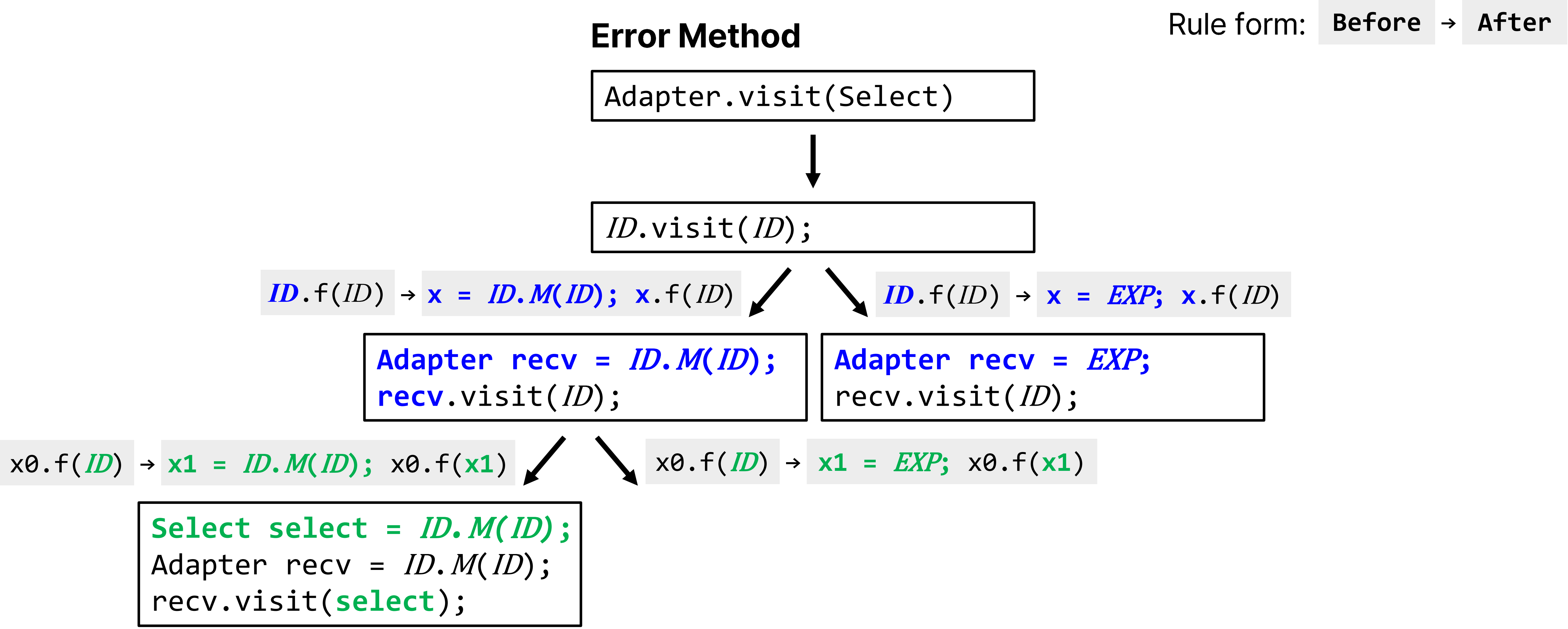
```
Adapter recv = EXP;
recv.visit(ID);
```

`x0.f(ID)` → `x1 = ID.M(ID); x0.f(x1)`

```
Select select = ID.M(ID);
Adapter recv = ID.M(ID);
recv.visit(select);
```

Top-down enumerative search

- Expand test cases top-down based on defined rules.



Rule form: `Before` → `After`

Error Method

```
Adapter.visit(Select)
```

```
ID.visit(ID);
```

`ID.f(ID)` → `x = ID.M(ID); x.f(ID)`

`ID.f(ID)` → `x = EXP; x.f(ID)`

```
Adapter recv = ID.M(ID);
recv.visit(ID);
```

```
Adapter recv = EXP;
recv.visit(ID);
```

`x0.f(ID)` → `x1 = ID.M(ID); x0.f(x1)`

`x0.f(ID)` → `x1 = EXP; x0.f(x1)`

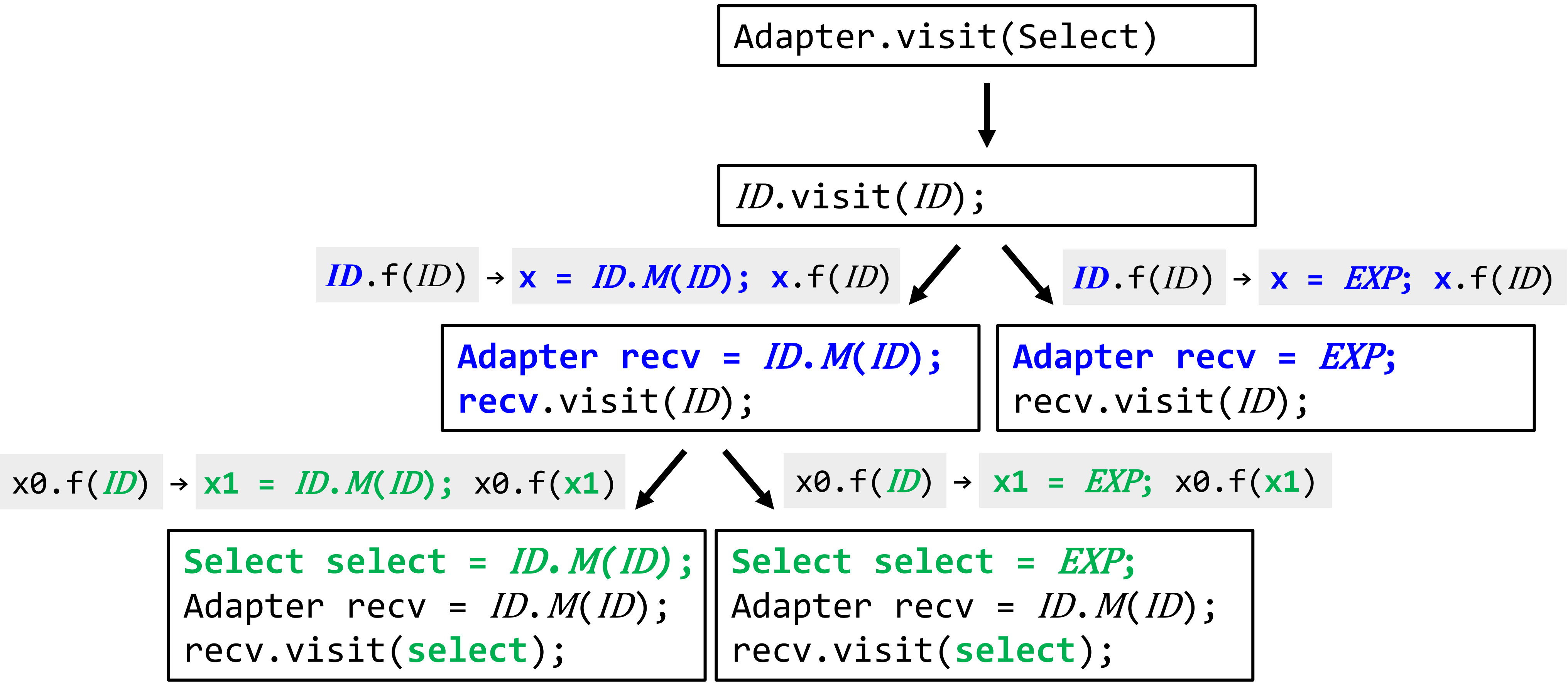
```
Select select = ID.M(ID);
Adapter recv = ID.M(ID);
recv.visit(select);
```

Top-down enumerative search

- Expand test cases top-down based on defined rules.

Rule form: `Before` → `After`

Error Method



```
Adapter.visit(Select)
```

```
ID.visit(ID);
```

`ID.f(ID)` → `x = ID.M(ID); x.f(ID)`

```
Adapter recv = ID.M(ID);
recv.visit(ID);
```

`ID.f(ID)` → `x = EXP; x.f(ID)`

```
Adapter recv = EXP;
recv.visit(ID);
```

`x0.f(ID)` → `x1 = ID.M(ID); x0.f(x1)`

```
Select select = ID.M(ID);
Adapter recv = ID.M(ID);
recv.visit(select);
```

`x0.f(ID)` → `x1 = EXP; x0.f(x1)`

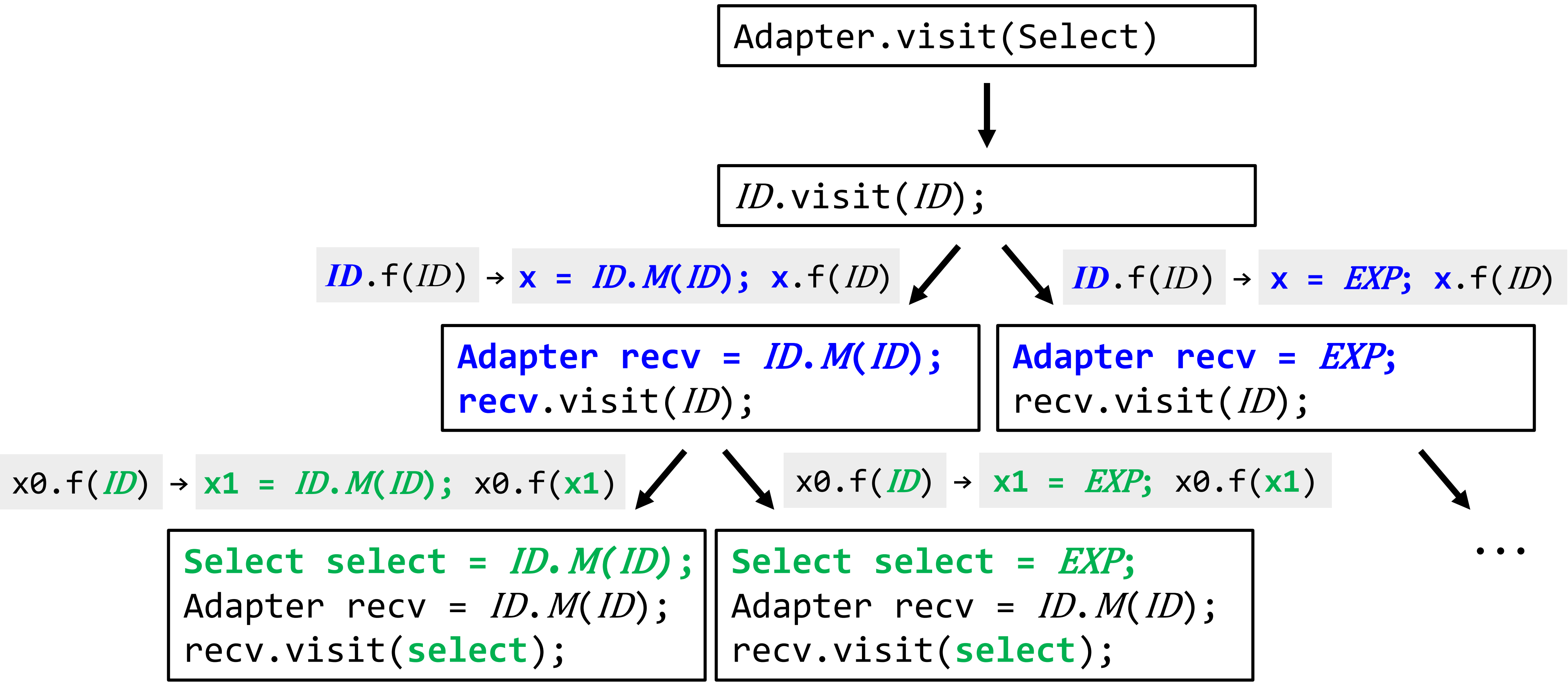
```
Select select = EXP;
Adapter recv = ID.M(ID);
recv.visit(select);
```

Top-down enumerative search

- Expand test cases top-down based on defined rules.

Rule form: `Before` → `After`

Error Method



`Adapter.visit(Select)`

ID.visit(*ID*);

ID.f(*ID*) → x = *ID*.*M*(*ID*); x.f(*ID*)

ID.f(*ID*) → x = *EXP*; x.f(*ID*)

Adapter recv = *ID*.*M*(*ID*);
recv.visit(*ID*);

Adapter recv = *EXP*;
recv.visit(*ID*);

x0.f(*ID*) → x1 = *ID*.*M*(*ID*); x0.f(x1)

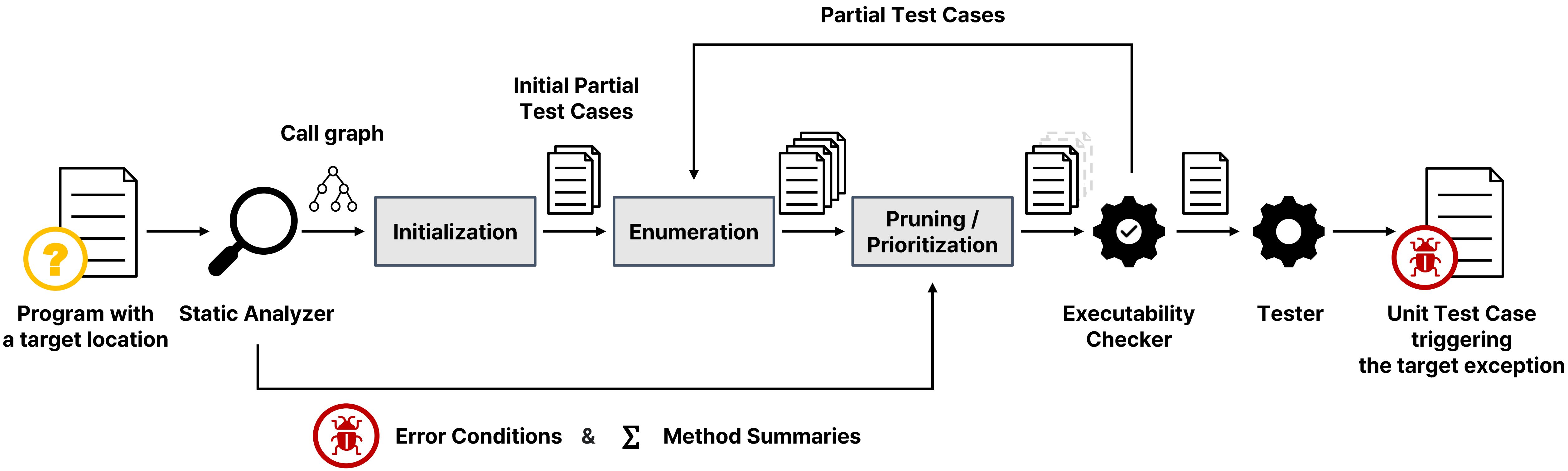
x0.f(*ID*) → x1 = *EXP*; x0.f(x1)

Select select = *ID*.*M*(*ID*);
Adapter recv = *ID*.*M*(*ID*);
recv.visit(select);

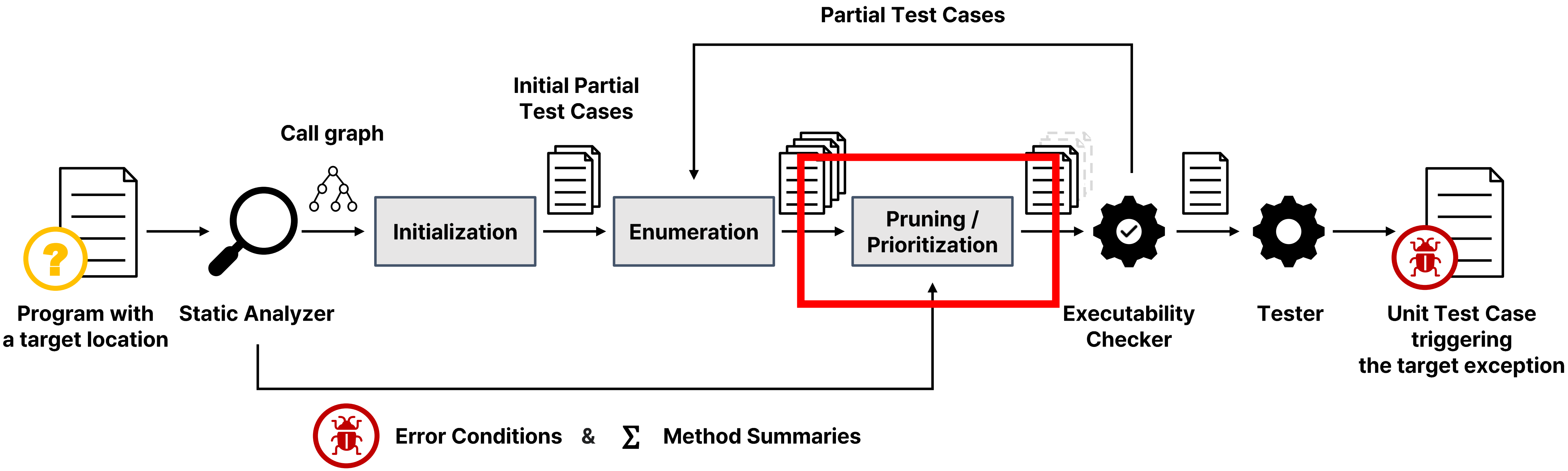
Select select = *EXP*;
Adapter recv = *ID*.*M*(*ID*);
recv.visit(select);

…

UnitCon System


Partial Test Cases
Initial Partial Test Cases
Call graph
Initialization
Enumeration
Pruning / Prioritization
Program with a target location
Static Analyzer
Executability Checker
Tester
Unit Test Case triggering the target exception
Error Conditions & Σ Method Summaries

UnitCon System


Partial Test Cases
Initial Partial Test Cases
Call graph
Initialization
Enumeration
Pruning / Prioritization
Program with a target location
Static Analyzer
Executability Checker
Tester
Unit Test Case triggering the target exception
Error Conditions & ∑ Method Summaries

Pruning

Pruning

- Discard partial test cases with identical semantics.

Pruning

- Discard partial test cases with identical semantics.

Current partial test case

```
Adapter recv = ID.M(ID);

Select select =
recv.visit(select);
```

Pruning

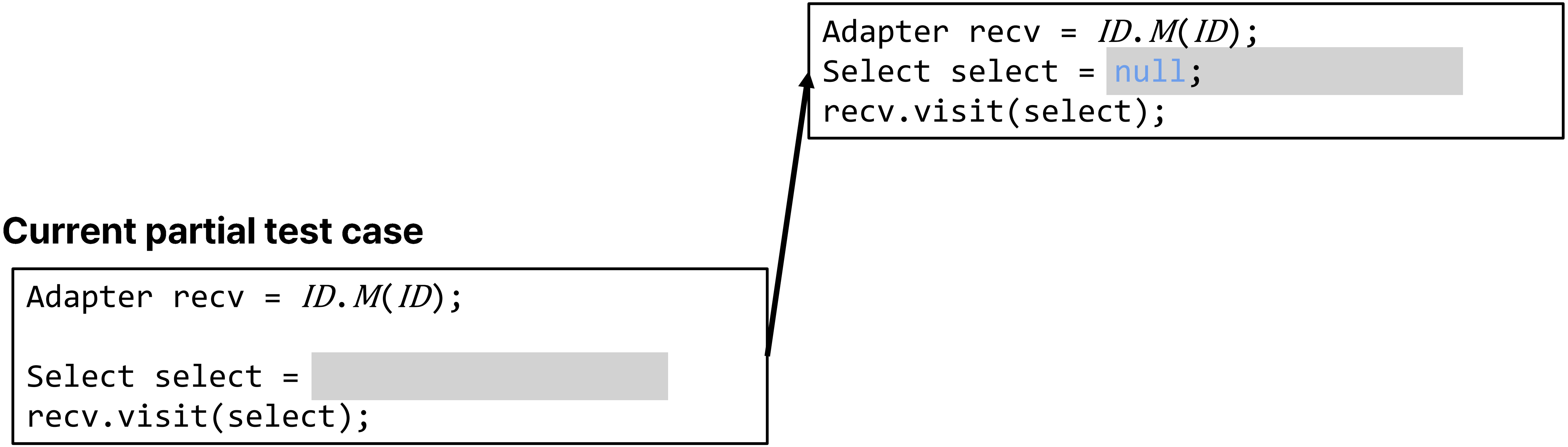
- Discard partial test cases with identical semantics.

Current partial test case

```
Adapter recv = ID.M(ID);

Select select = 
recv.visit(select);
```

Pruning

- Discard partial test cases with identical semantics.



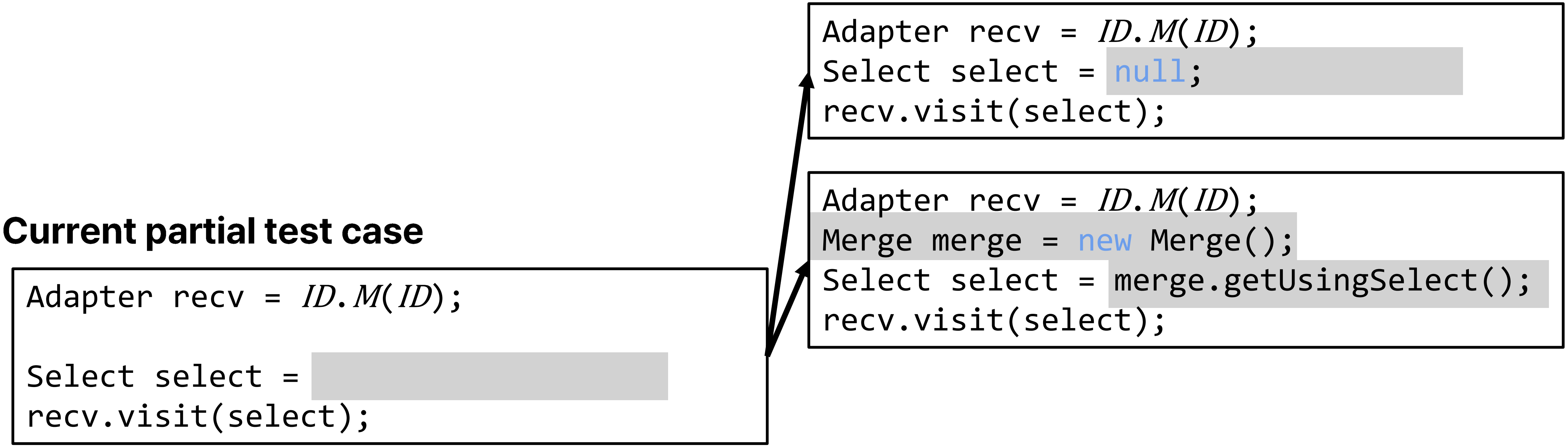
Current partial test case

```
Adapter recv = ID.M(ID);

Select select = 
recv.visit(select);
```

```
Adapter recv = ID.M(ID);
Select select = null;
recv.visit(select);
```

Pruning

- Discard partial test cases with identical semantics.



Current partial test case

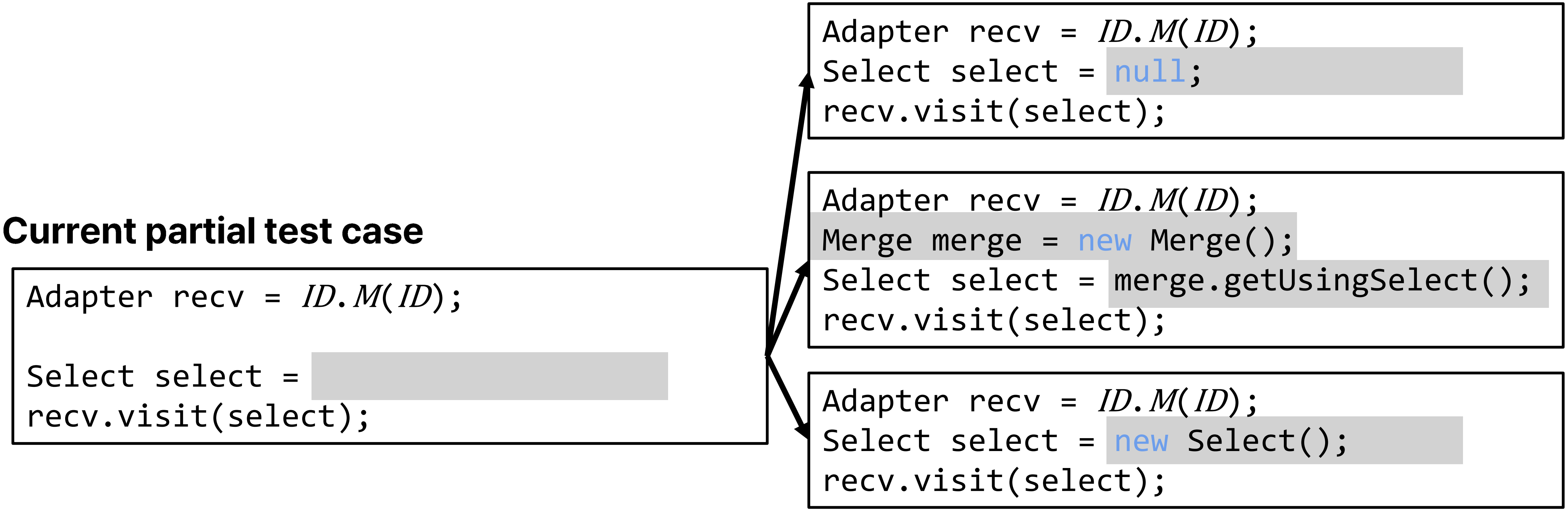
```
Adapter recv = ID.M(ID);

Select select = 
recv.visit(select);
```

```
Adapter recv = ID.M(ID);
Select select = null;
recv.visit(select);
```

```
Adapter recv = ID.M(ID);
Merge merge = new Merge();
Select select = merge.getUsingSelect();
recv.visit(select);
```

Pruning

- Discard partial test cases with identical semantics.



Current partial test case

```
Adapter recv = ID.M(ID);

Select select = 
recv.visit(select);
```

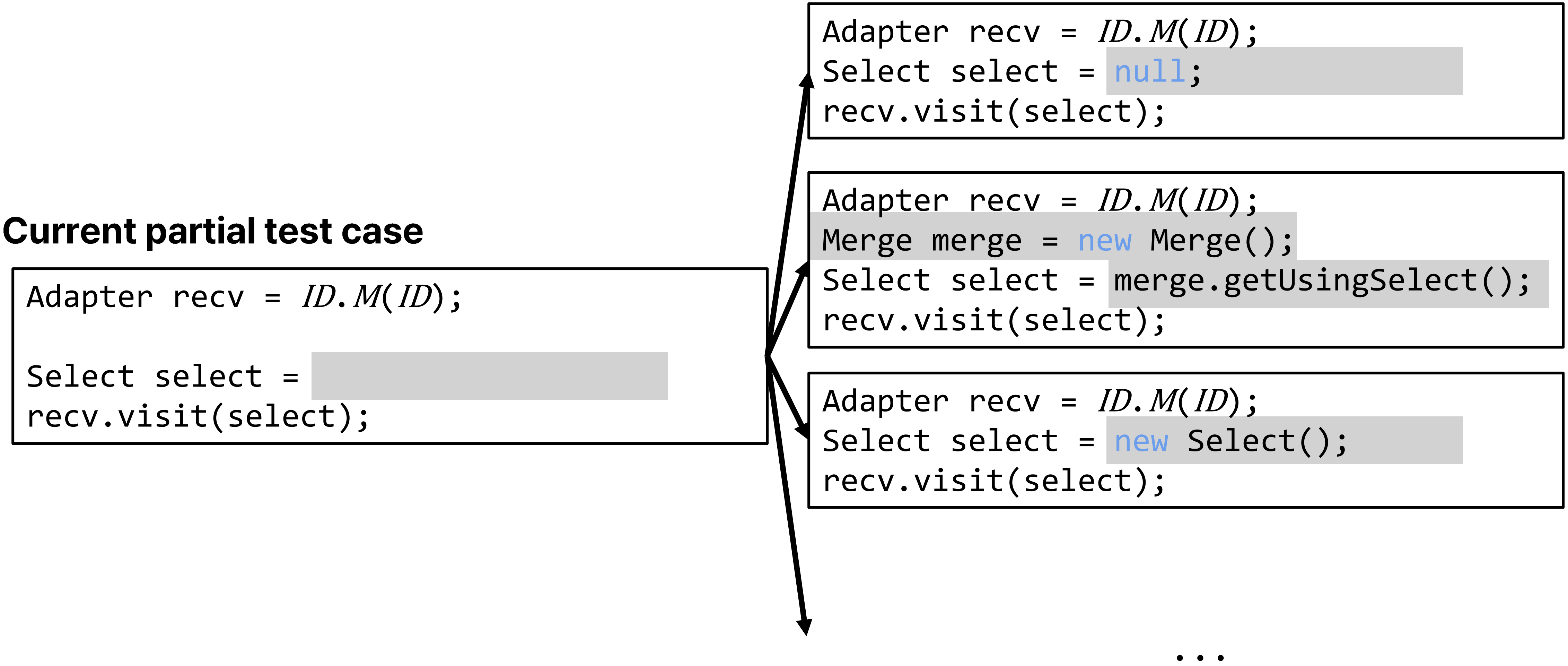
```
Adapter recv = ID.M(ID);
Select select = null;
recv.visit(select);
```

```
Adapter recv = ID.M(ID);
Merge merge = new Merge();
Select select = merge.getUsingSelect();
recv.visit(select);
```

```
Adapter recv = ID.M(ID);
Select select = new Select();
recv.visit(select);
```

Pruning

- Discard partial test cases with identical semantics.



Current partial test case

```
Adapter recv = ID.M(ID);

Select select = 
recv.visit(select);
```

```
Adapter recv = ID.M(ID);
Select select = null;
recv.visit(select);
```

```
Adapter recv = ID.M(ID);
Merge merge = new Merge();
Select select = merge.getUsingSelect();
recv.visit(select);
```

```
Adapter recv = ID.M(ID);
Select select = new Select();
recv.visit(select);
```

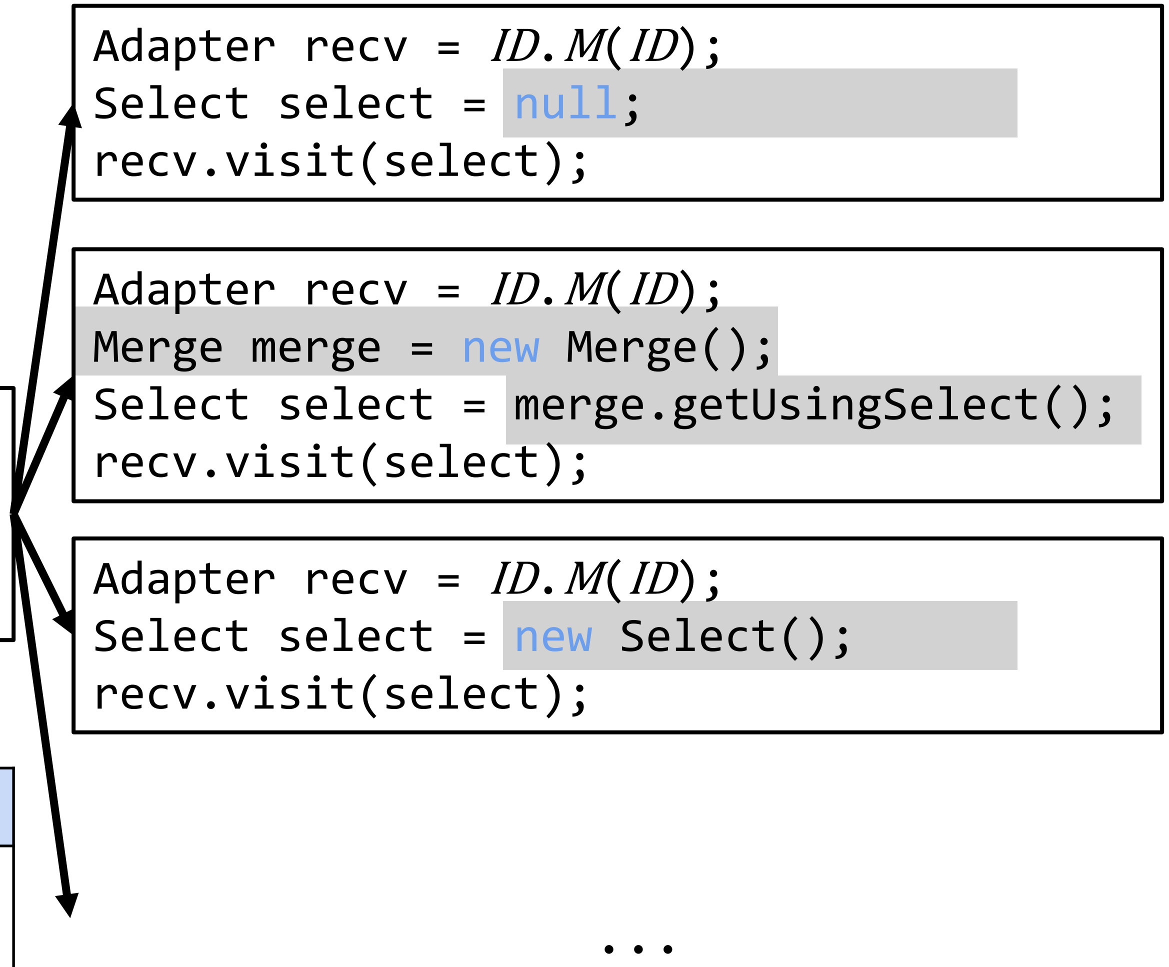
...

Pruning

- Discard partial test cases with identical semantics.

Current partial test case

```
Adapter recv = ID.M(ID);

Select select = 
recv.visit(select);
```



```
Adapter recv = ID.M(ID);
Select select = null;
recv.visit(select);
```

```
Adapter recv = ID.M(ID);
Merge merge = new Merge();
Select select = merge.getUsingSelect();
recv.visit(select);
```

```
Adapter recv = ID.M(ID);
Select select = new Select();
recv.visit(select);
```

...

Method Summaries

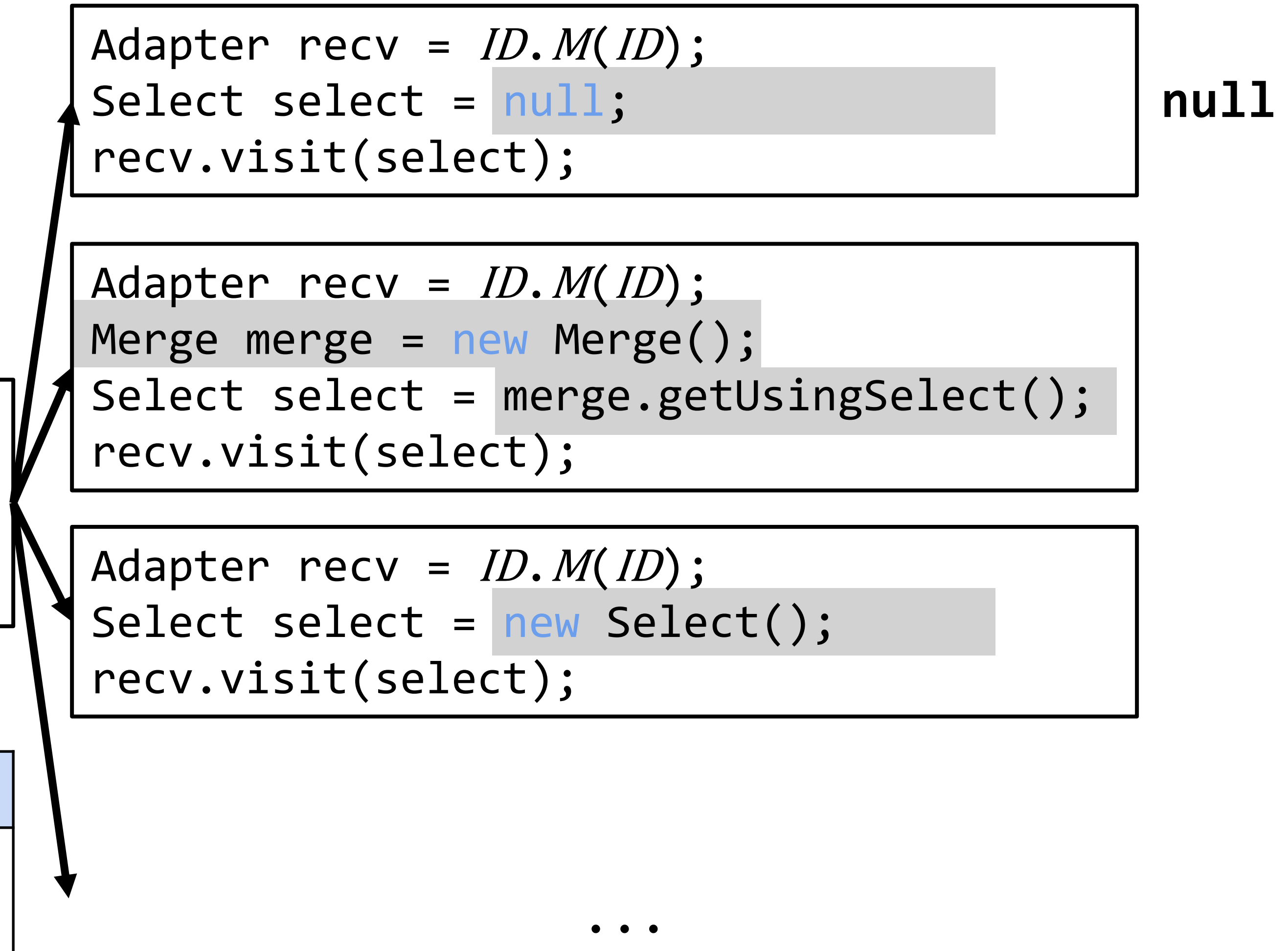
Method	Memory
`Merge`	`{ usingSelect ↦ null }`
`getUsingSelect`	`{ ret ↦ usingSelect }`
...	...

Pruning

- Discard partial test cases with identical semantics.

Current partial test case

```
Adapter recv = ID.M(ID);

Select select = 
recv.visit(select);
```



```
Adapter recv = ID.M(ID);
Select select = null;
recv.visit(select);
```

null

```
Adapter recv = ID.M(ID);
Merge merge = new Merge();
Select select = merge.getUsingSelect();
recv.visit(select);
```

```
Adapter recv = ID.M(ID);
Select select = new Select();
recv.visit(select);
```

...

Method Summaries

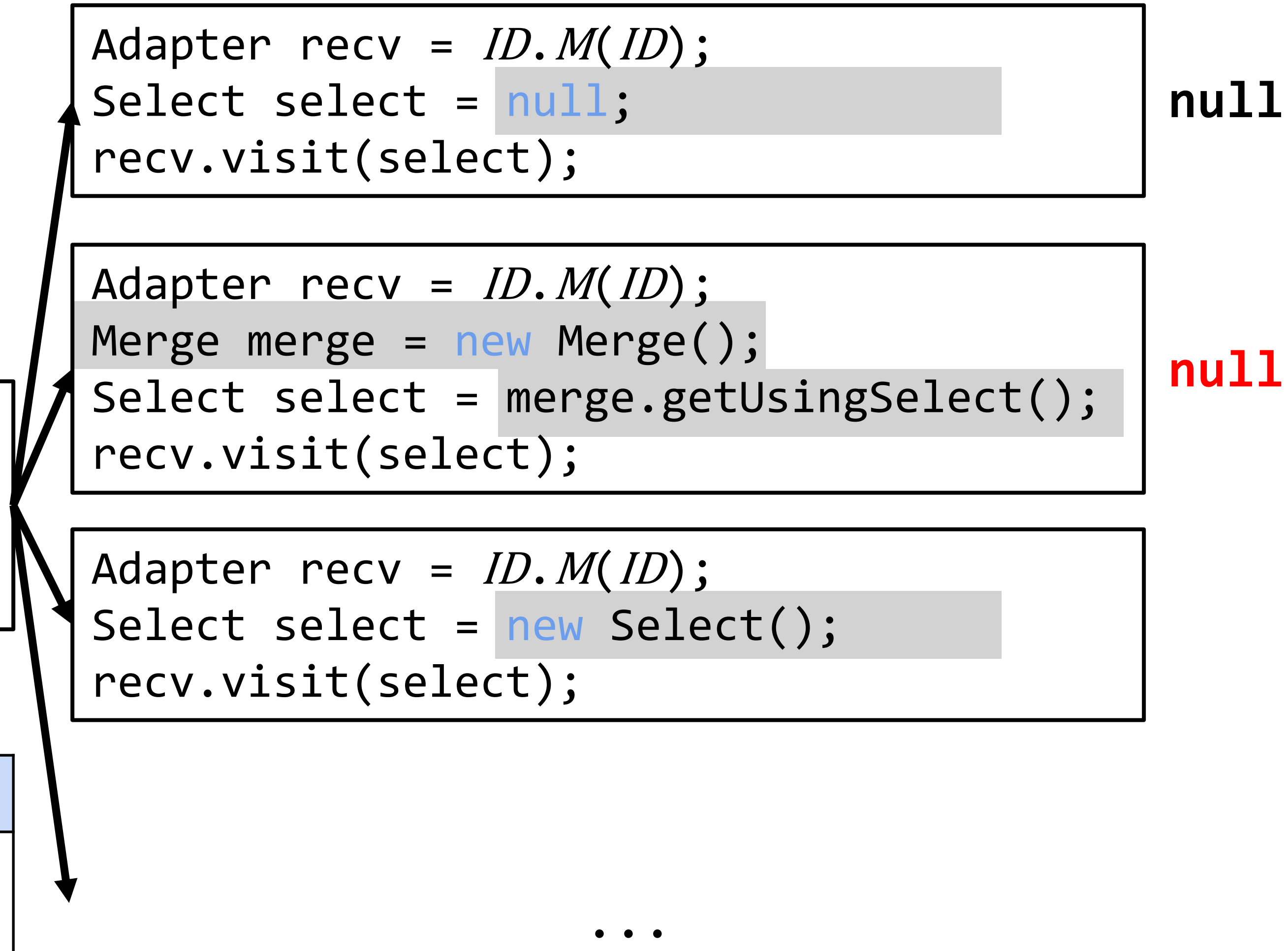
Method	Memory
Merge	{ usingSelect ↦ null }
getUsingSelect	{ ret ↦ usingSelect }
...	...

Pruning

- Discard partial test cases with identical semantics.

Current partial test case

```
Adapter recv = ID.M(ID);

Select select = 
recv.visit(select);
```



```
Adapter recv = ID.M(ID);
Select select = null;
recv.visit(select);
```

```
Adapter recv = ID.M(ID);
Merge merge = new Merge();
Select select = merge.getUsingSelect();
recv.visit(select);
```

```
Adapter recv = ID.M(ID);
Select select = new Select();
recv.visit(select);
```

Method Summaries

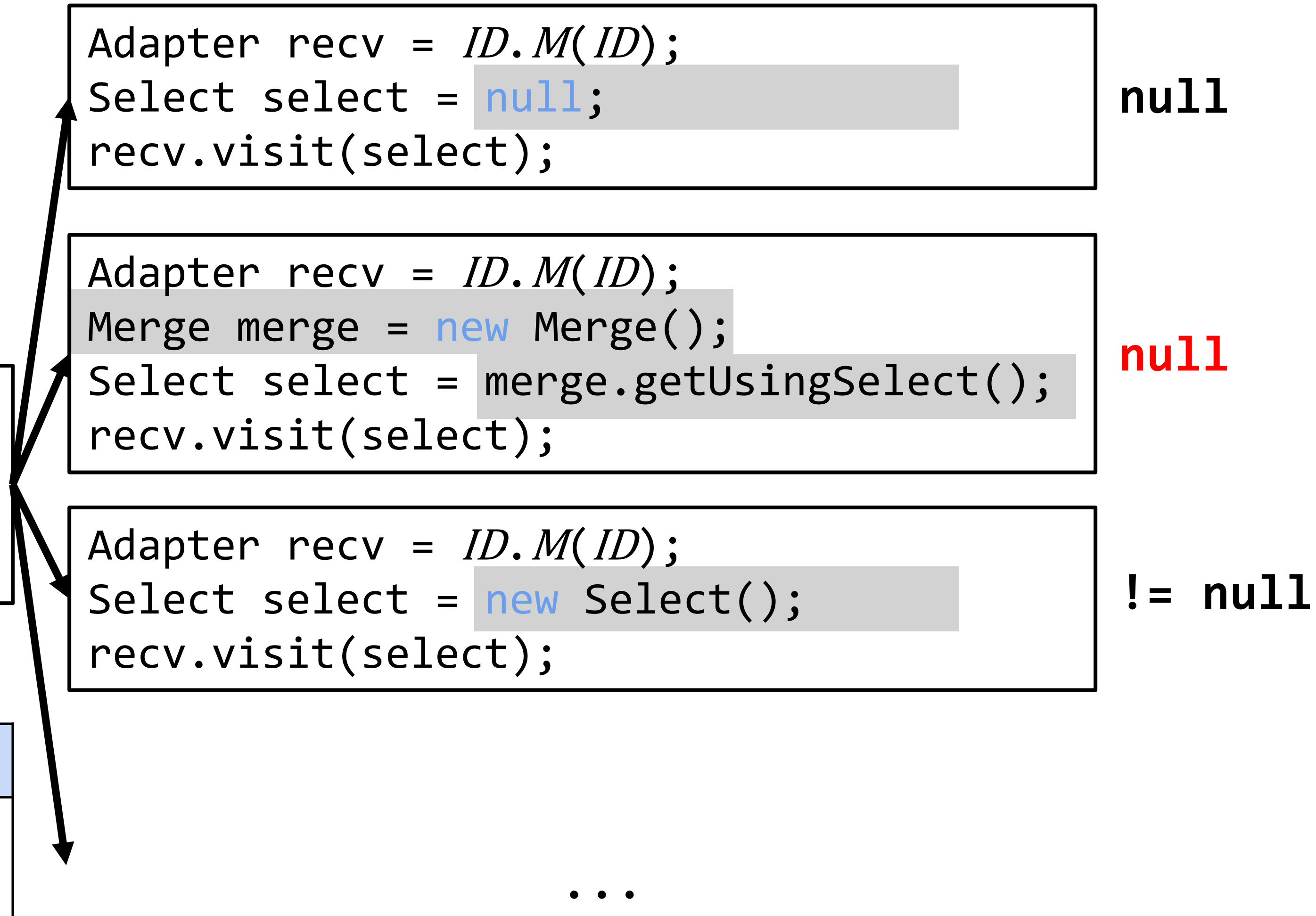
Method	Memory
Merge	{ usingSelect ↦ null }
getUsingSelect	{ ret ↦ usingSelect }
...	...

Pruning

- Discard partial test cases with identical semantics.

Current partial test case

```
Adapter recv = ID.M(ID);

Select select = 
recv.visit(select);
```



```
Adapter recv = ID.M(ID);
Select select = null;
recv.visit(select);
```

null

```
Adapter recv = ID.M(ID);
Merge merge = new Merge();
Select select = merge.getUsingSelect();
recv.visit(select);
```

null

```
Adapter recv = ID.M(ID);
Select select = new Select();
recv.visit(select);
```

!= null

...

Method Summaries

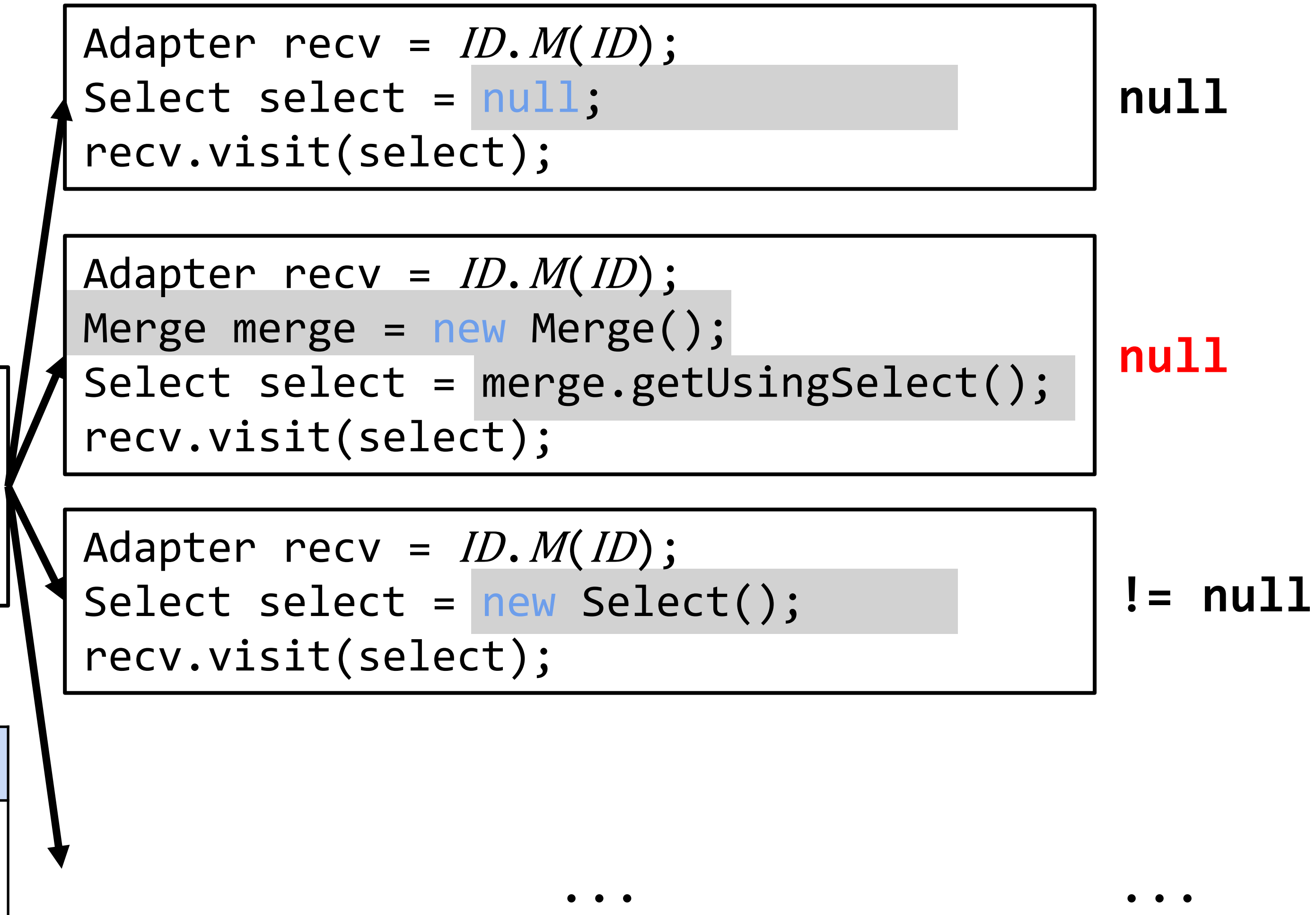
Method	Memory
Merge	{ usingSelect ↦ null }
getUsingSelect	{ ret ↦ usingSelect }
...	...

Pruning

- Discard partial test cases with identical semantics.

Current partial test case

```
Adapter recv = ID.M(ID);

Select select = 
recv.visit(select);
```



```
Adapter recv = ID.M(ID);
Select select = null;
recv.visit(select);
```

null

```
Adapter recv = ID.M(ID);
Merge merge = new Merge();
Select select = merge.getUsingSelect();
recv.visit(select);
```

null

```
Adapter recv = ID.M(ID);
Select select = new Select();
recv.visit(select);
```

!= null

...

...

Method Summaries

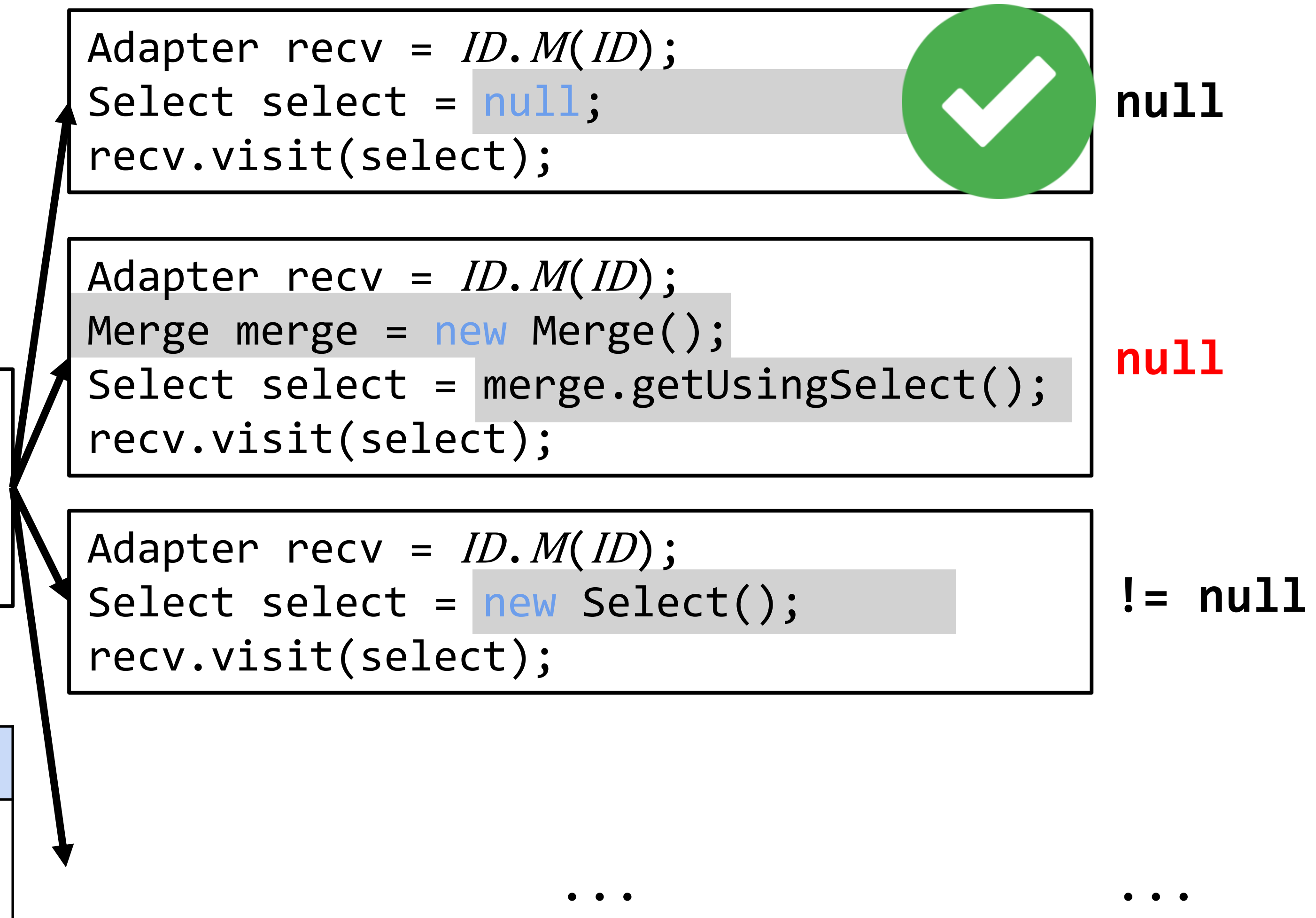
Method	Memory
Merge	{ usingSelect ↦ null }
getUsingSelect	{ ret ↦ usingSelect }
...	...

Pruning

- Discard partial test cases with identical semantics.

Current partial test case

```
Adapter recv = ID.M(ID);

Select select = 
recv.visit(select);
```



```
Adapter recv = ID.M(ID);
Select select = null;
recv.visit(select);
```

```
Adapter recv = ID.M(ID);
Merge merge = new Merge();
Select select = merge.getUsingSelect();
recv.visit(select);
```

```
Adapter recv = ID.M(ID);
Select select = new Select();
recv.visit(select);
```

Method Summaries

Method	Memory
Merge	{ usingSelect ↦ null }
getUsingSelect	{ ret ↦ usingSelect }
...	...

Pruning

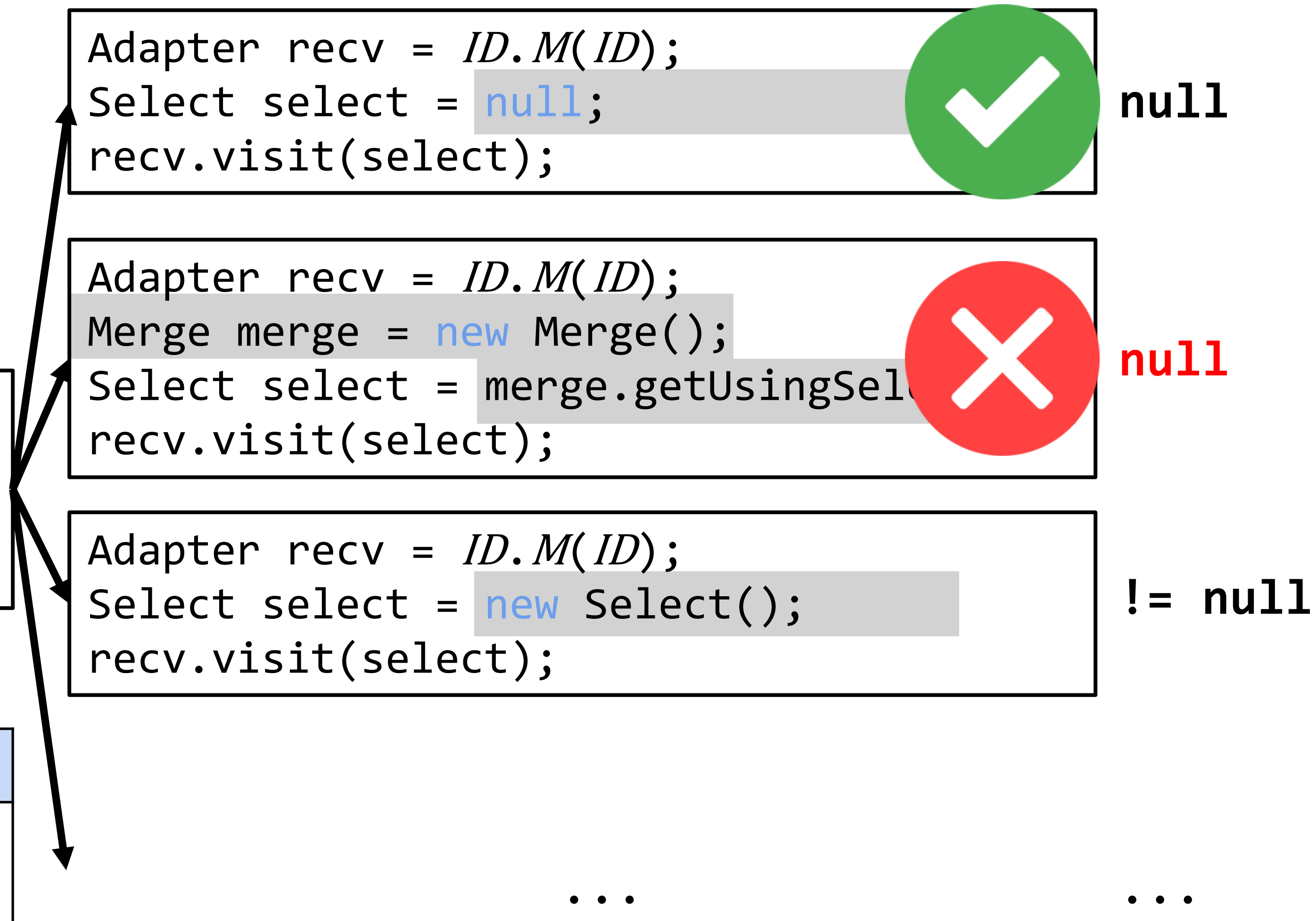
- Discard partial test cases with identical semantics.

Current partial test case

```
Adapter recv = ID.M(ID);

Select select = 
recv.visit(select);
```

Method Summaries

Method	Memory
Merge	{ usingSelect ↦ null }
getUsingSelect	{ ret ↦ usingSelect }
...	...



```
Adapter recv = ID.M(ID);
Select select = null;
recv.visit(select);
```

✓ null

```
Adapter recv = ID.M(ID);
Merge merge = new Merge();
Select select = merge.getUsingSel
recv.visit(select);
```

✗ null

```
Adapter recv = ID.M(ID);
Select select = new Select();
recv.visit(select);
```

!= null

...

...

Pruning

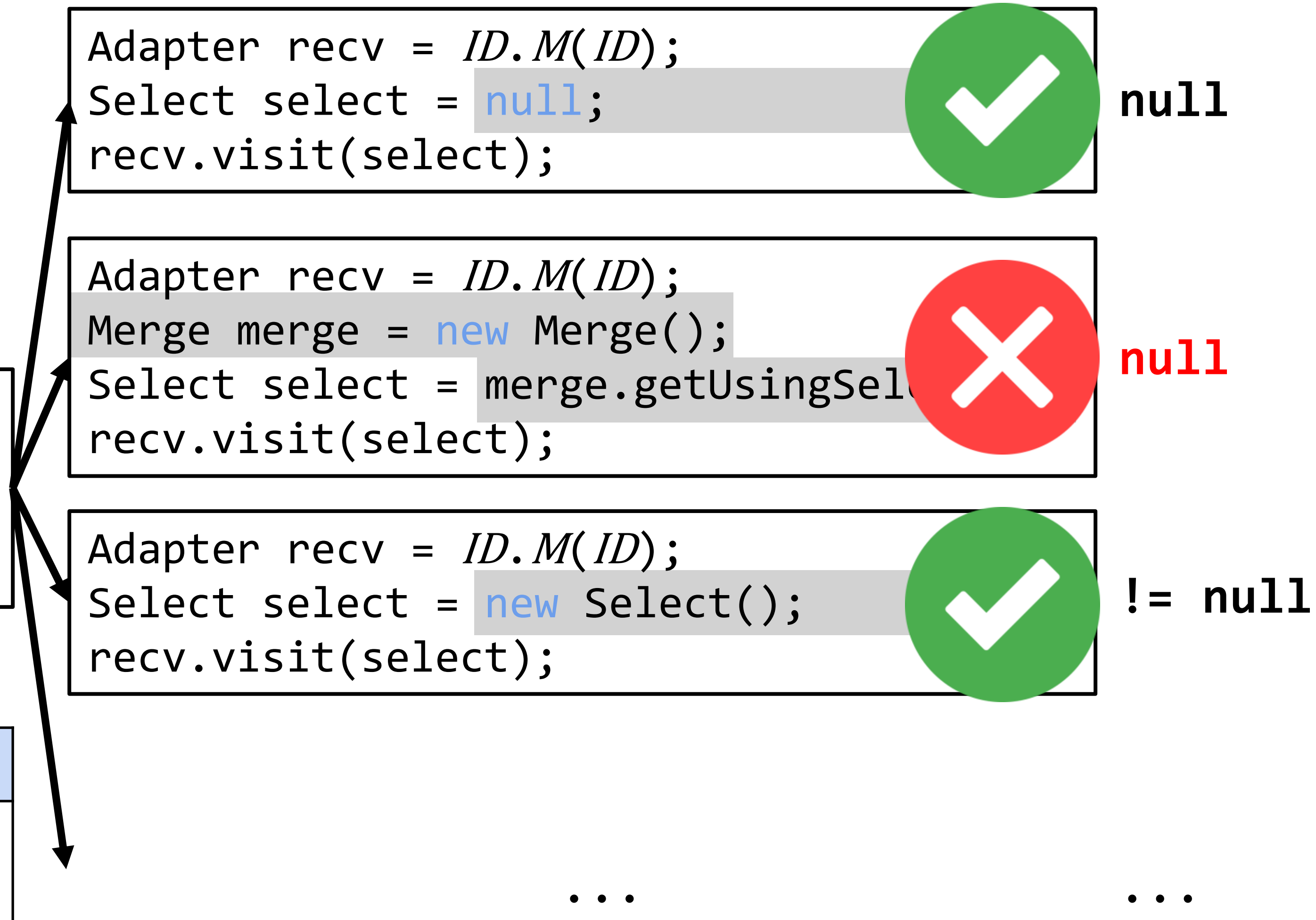
- Discard partial test cases with identical semantics.

Current partial test case

```
Adapter recv = ID.M(ID);

Select select = 
recv.visit(select);
```

Method Summaries

Method	Memory
Merge	{ usingSelect ↦ null }
getUsingSelect	{ ret ↦ usingSelect }
...	...



```
Adapter recv = ID.M(ID);
Select select = null;
recv.visit(select);
```

✓ null

```
Adapter recv = ID.M(ID);
Merge merge = new Merge();
Select select = merge.getUsingSel
recv.visit(select);
```

✗ null

```
Adapter recv = ID.M(ID);
Select select = new Select();
recv.visit(select);
```

✓ != null

... ...

Prioritization

Prioritization

```
Adapter recv = ID.M(ID);
Select select = null;
recv.visit(select);
```

Prioritization

```
Adapter recv = ID.M(ID);
Select select = null;
recv.visit(select);
```

```
Adapter recv = ID.M(ID);
Select select = new Select();
recv.visit(select);
```

Prioritization

```
Adapter recv = ID.M(ID);
Select select = null;
recv.visit(select);
```

```
Adapter recv = ID.M(ID);
Select select = new Select();
recv.visit(select);
```

...

Prioritization

- Prioritize test cases that are more likely to satisfy the error conditions.

```
Adapter recv = ID.M(ID);
Select select = null;
recv.visit(select);
```

```
Adapter recv = ID.M(ID);
Select select = new Select();
recv.visit(select);
```

...

Prioritization

- Prioritize test cases that are more likely to satisfy the error conditions.

```
Adapter recv = ID.M(ID);
Select select = null;
recv.visit(select);
```

```
Adapter recv = ID.M(ID);
Select select = new Select();
recv.visit(select);
```

...

Error Conditions

Object	Condition
`select.itemsList`	`== null`
`this.visitor`	`!= null`
`select`	`!= null`

Prioritization

- Prioritize test cases that are more likely to satisfy the error conditions.

```
Adapter recv = ID.M(ID);
Select select = null;
recv.visit(select);
```

```
Adapter recv = ID.M(ID);
Select select = new Select();
recv.visit(select);
```

...

Error Conditions

Object	Condition
`select.itemsList`	`== null`
`this.visitor`	`!= null`
`select`	`!= null`

Prioritization

- Prioritize test cases that are more likely to satisfy the error conditions.

```
Adapter recv = ID.M(ID);
Select select = null;
recv.visit(select);
```

```
Adapter recv = ID.M(ID);
Select select = new Select();
recv.visit(select);
```

...

Error Conditions

Object	Condition
select.itemsList	== null
this.visitor	!= null
select	!= null

Prioritization

- Prioritize test cases that are more likely to satisfy the error conditions.

```
Adapter recv = ID.M(ID);
Select select = null;
recv.visit(select);
```

```
Adapter recv = ID.M(ID);
Select select = new Select();
recv.visit(select);
```

...

Error Conditions

Object	Condition
`select.itemsList`	`== null`
`this.visitor`	`!= null`
`select`	`!= null`

Prioritization

- Prioritize test cases that are more likely to satisfy the error conditions.

```
Adapter recv = ID.M(ID);
Select select = null;
recv.visit(select);
```

```
Adapter recv = ID.M(ID);
Select select = new Select();
recv.visit(select);
```

. . .

Error Conditions

Object	Condition
`select.itemsList`	`== null`
`this.visitor`	`!= null`
`select`	`!= null`

Prioritization

- Prioritize test cases that are more likely to satisfy the error conditions.

👎

```
Adapter recv = ID.M(ID);
Select select = null;
recv.visit(select);
```

👍

```
Adapter recv = ID.M(ID);
Select select = new Select();
recv.visit(select);
```

...

Error Conditions

Object	Condition
`select.itemsList`	`== null`
`this.visitor`	`!= null`
`select`	`!= null`

Example

```java
public class Adapter {
  private Visitor visitor;

  public void setVisitor(Visitor visitor) {
    this.visitor = visitor;
  }

  public void visit(Select select) {
    if (visitor != null) {
      ItemsList itemsList = select.getItemsList();
      for (Item item: itemsList) { // Target location
        ...
}

public class Select {
  private List <Item> itemsList;

  public List <Item> getItemsList() { return itemsList; }
}

public class Merge {
  private Select usingSelect;

  public Select getUsingSelect() { return usingSelect; }
}
```

```java
public void test() {
  Adapter recv = new Adapter();
  Visitor visitor = new Visitor();
  recv.setVisitor(visitor);
  Select select = new Select();
  recv.visit(select);
}
```

Exception reproduced in **39 seconds**.

Evaluation: Known Bug Reproduction

Evaluation: Known Bug Reproduction

Benchmarks

Evaluation: Known Bug Reproduction

Benchmarks

- 198 Java programs (1 target exception per program).

Evaluation: Known Bug Reproduction

Benchmarks

- 198 Java programs (1 target exception per program).
  - 40K – 100K LOC

Evaluation: Known Bug Reproduction

Benchmarks

- 198 Java programs (1 target exception per program).
  - 40K – 100K LOC
  - Defects4J (ISSTA'14), Bears (SANER'19), GENESIS (ESEC/FSE'17), NPEX (ICSE'22), VFIX (ICSE'19)

Evaluation: Known Bug Reproduction

Benchmarks

- 198 Java programs (1 target exception per program).
  - 40K – 100K LOC
  - Defects4J (ISSTA’14), Bears (SANER’19), GENESIS (ESEC/FSE’17), NPEX (ICSE’22), VFIX (ICSE’19)

Baselines

Evaluation: Known Bug Reproduction

Benchmarks

- 198 Java programs (1 target exception per program).
  - 40K – 100K LOC
  - Defects4J (ISSTA’14), Bears (SANER’19), GENESIS (ESEC/FSE’17), NPEX (ICSE’22), VFIX (ICSE’19)

Baselines

- EvoSuite (ESEC/FSE’11), EvoFuzz (SBFT’24), NPETest (ASE’24), UTBot (SBFT’23), Randoop (OOPSLA’07)

Evaluation: Known Bug Reproduction

Benchmarks

- 198 Java programs (1 target exception per program).
    - 40K – 100K LOC
    - Defects4J (ISSTA'14), Bears (SANER'19), GENESIS (ESEC/FSE'17), NPEX (ICSE'22), VFIX (ICSE'19)

Baselines

- EvoSuite (ESEC/FSE'11), EvoFuzz (SBFT'24), NPETest (ASE'24), UTBot (SBFT'23), Randoop (OOPSLA'07)
- Repeat the 10 runs for tools with randomness.

Evaluation: Known Bug Reproduction

Benchmarks

- 198 Java programs (1 target exception per program).
  - 40K – 100K LOC
  - Defects4J (ISSTA'14), Bears (SANER'19), GENESIS (ESEC/FSE'17), NPEX (ICSE'22), VFIX (ICSE'19)

Baselines

- EvoSuite (ESEC/FSE'11), EvoFuzz (SBFT'24), NPETest (ASE'24), UTBot (SBFT'23), Randoop (OOPSLA'07)
- Repeat the 10 runs for tools with randomness.

Task

Evaluation: Known Bug Reproduction

Benchmarks

- 198 Java programs (1 target exception per program).
  - 40K – 100K LOC
  - Defects4J (ISSTA’14), Bears (SANER’19), GENESIS (ESEC/FSE’17), NPEX (ICSE’22), VFIX (ICSE’19)

Baselines

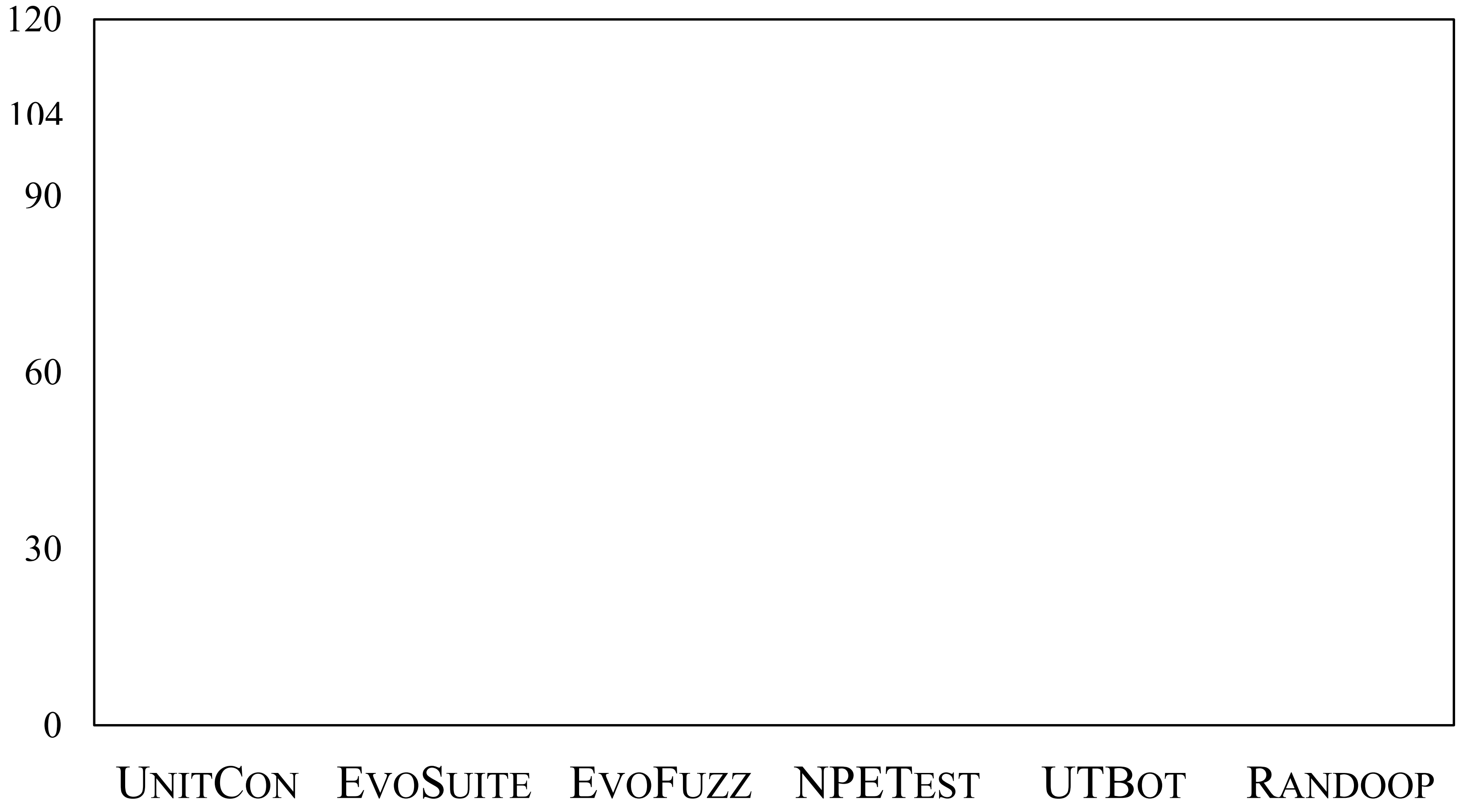
- EvoSuite (ESEC/FSE’11), EvoFuzz (SBFT’24), NPETest (ASE’24), UTBot (SBFT’23), Randoop (OOPSLA’07)
- Repeat the 10 runs for tools with randomness.

Task

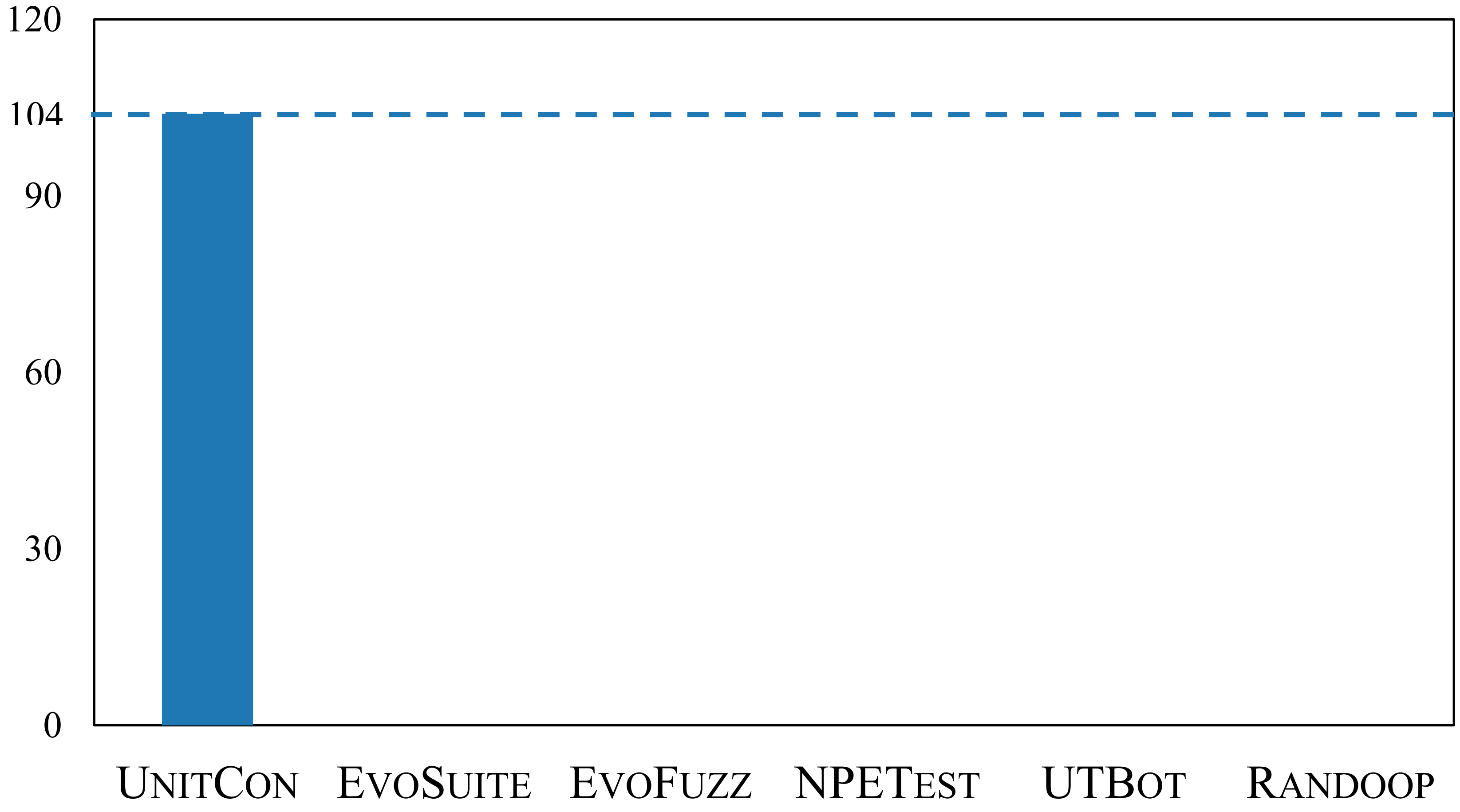
- 10 minutes time limit per 1 target exception.

Comparison to Baselines

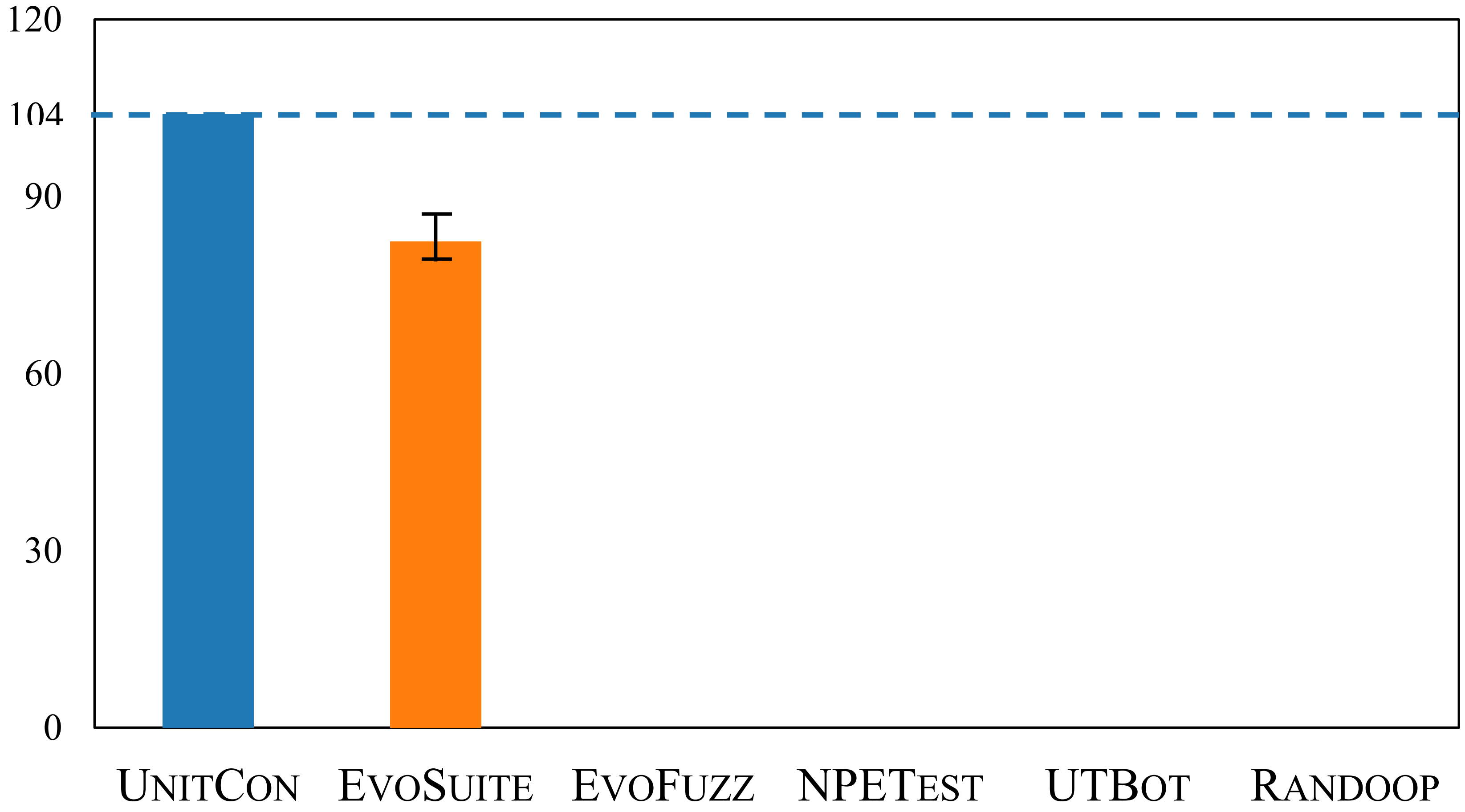
Comparison to Baselines



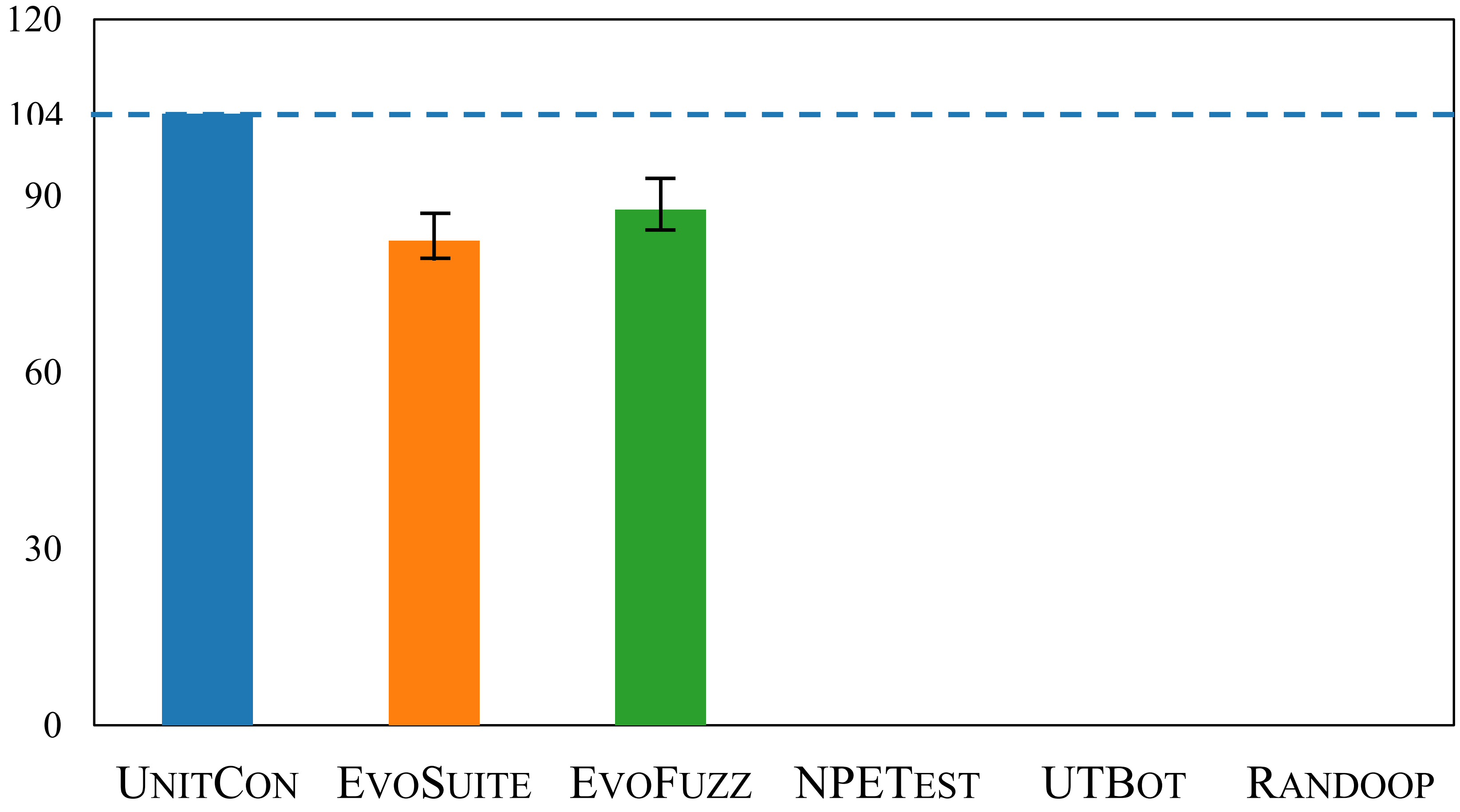
Comparison to Baselines



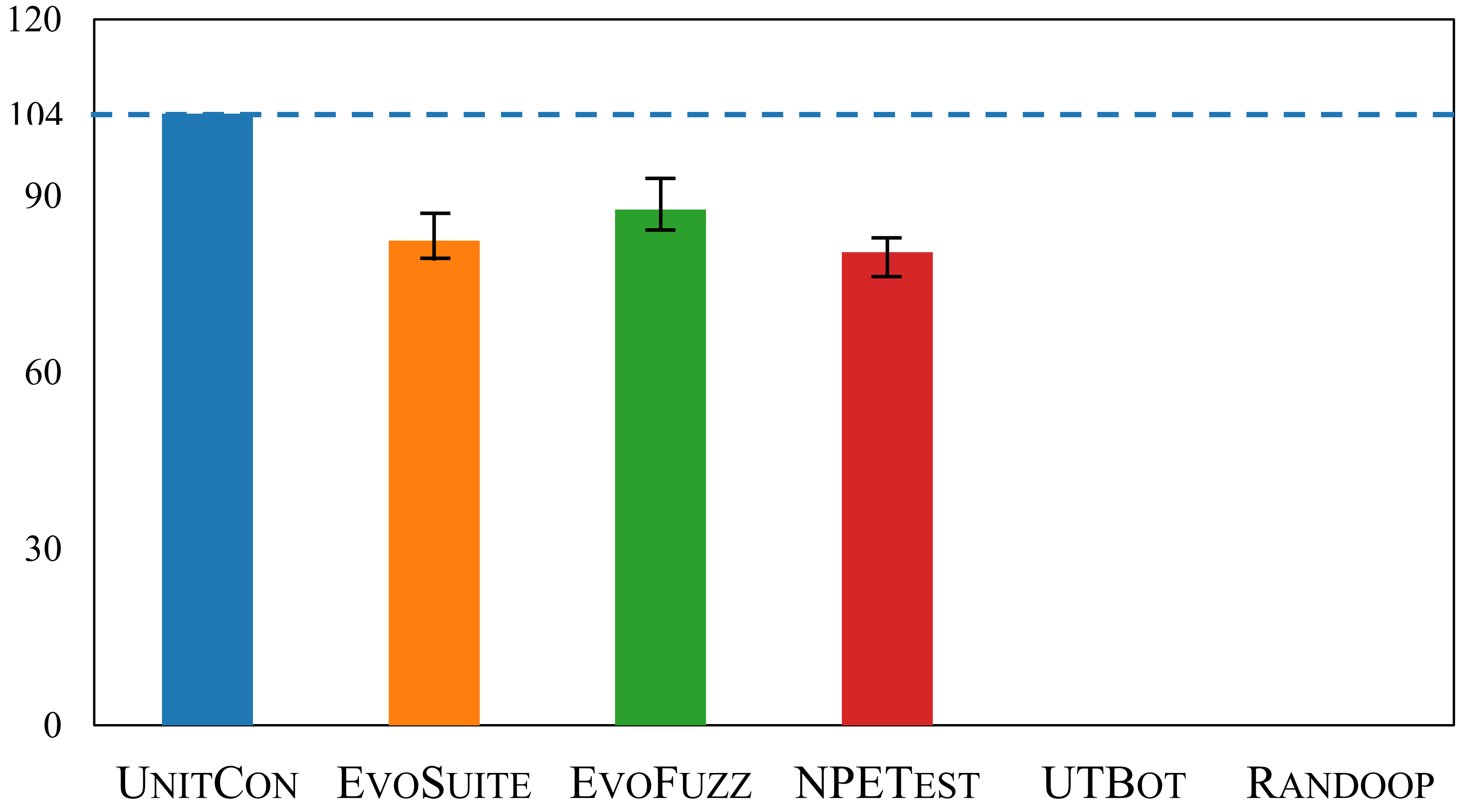
Comparison to Baselines



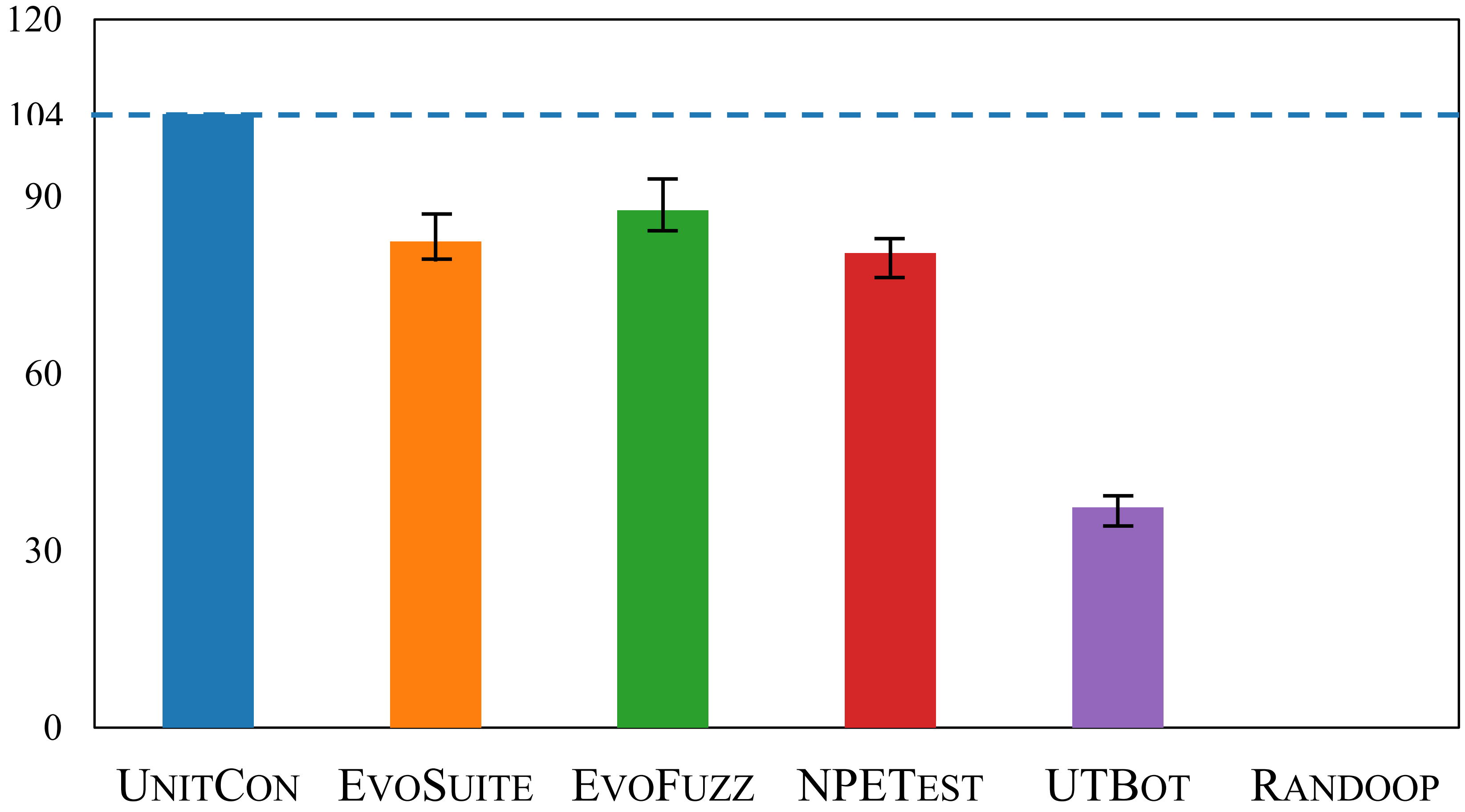
Comparison to Baselines



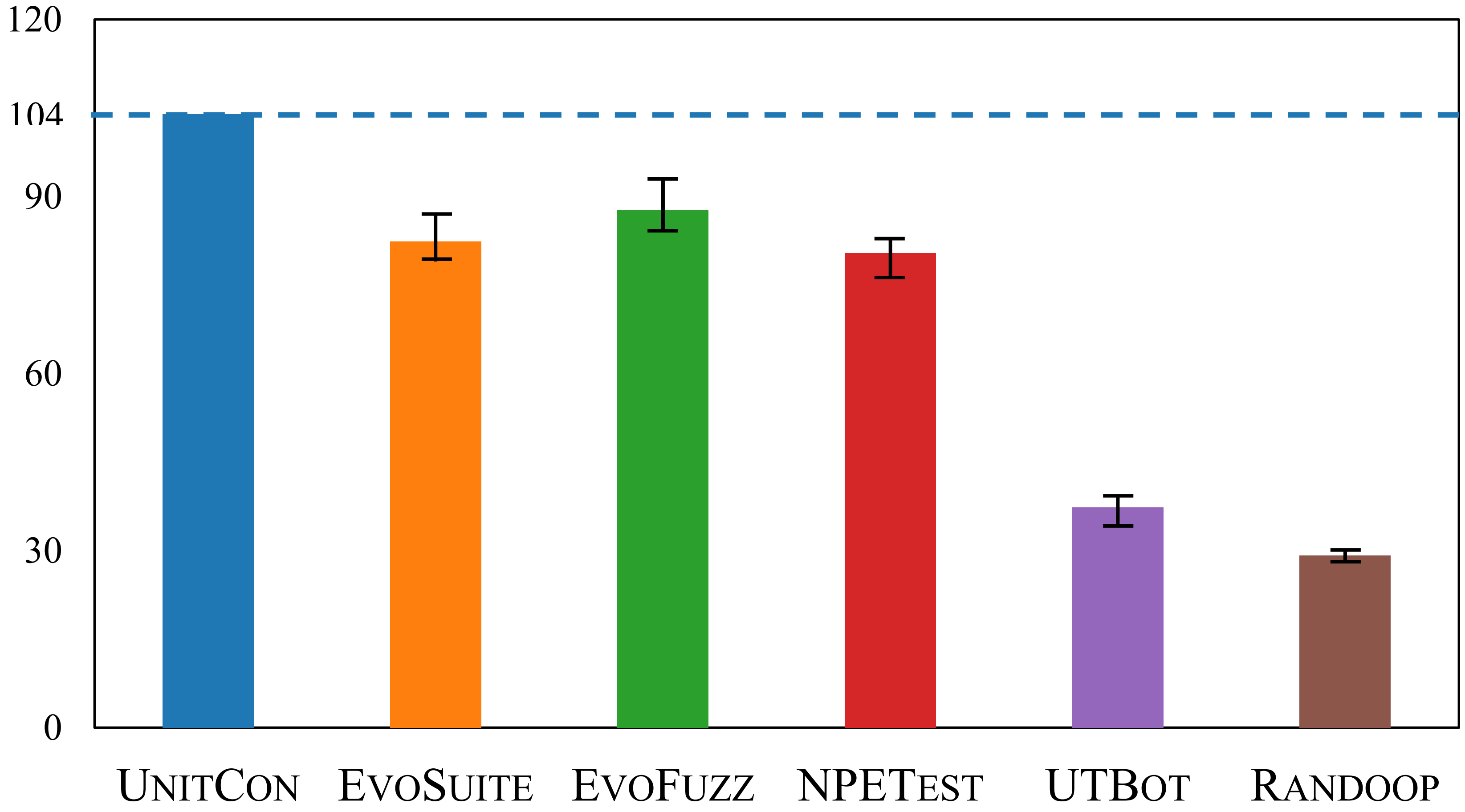
Comparison to Baselines



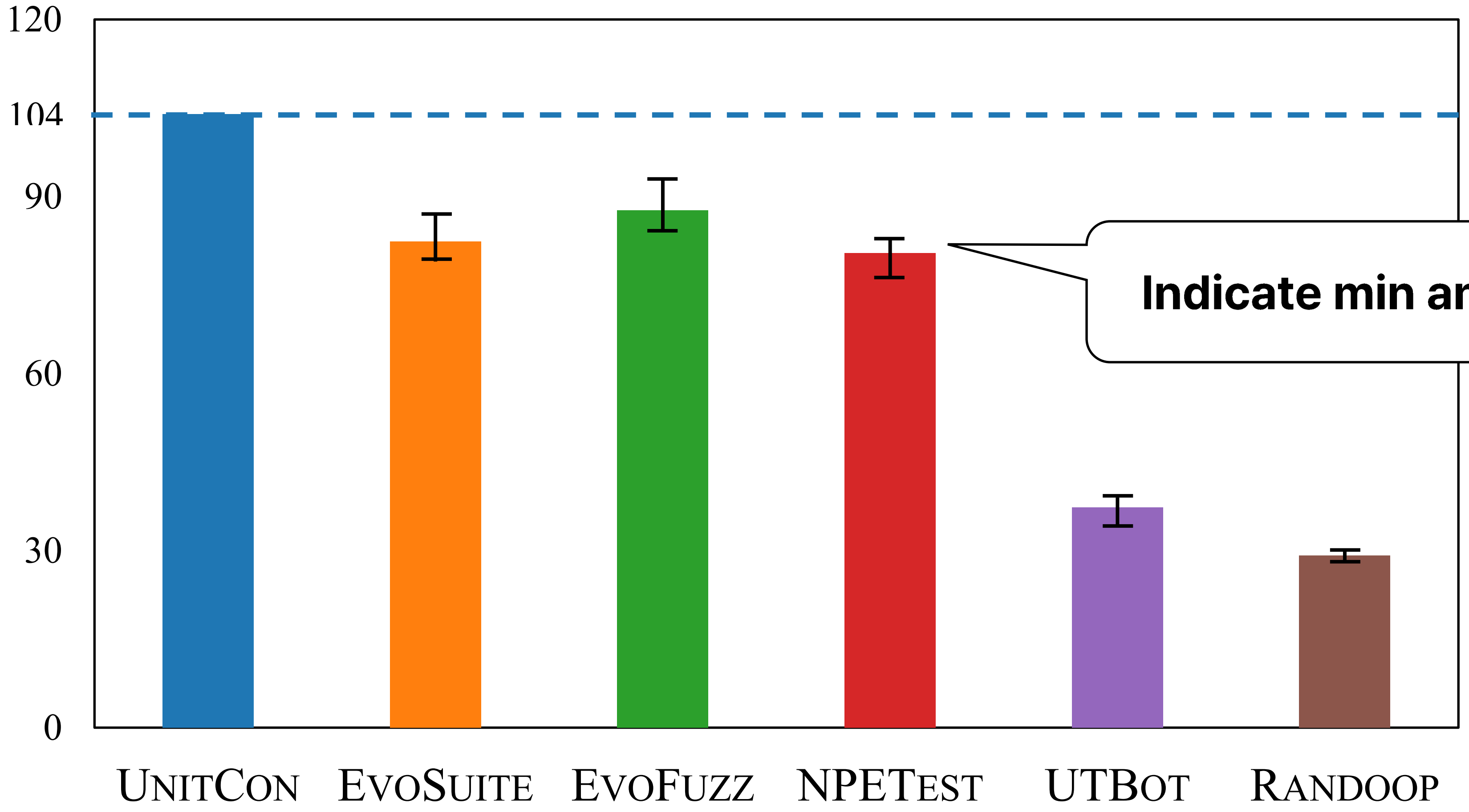
Comparison to Baselines



Comparison to Baselines



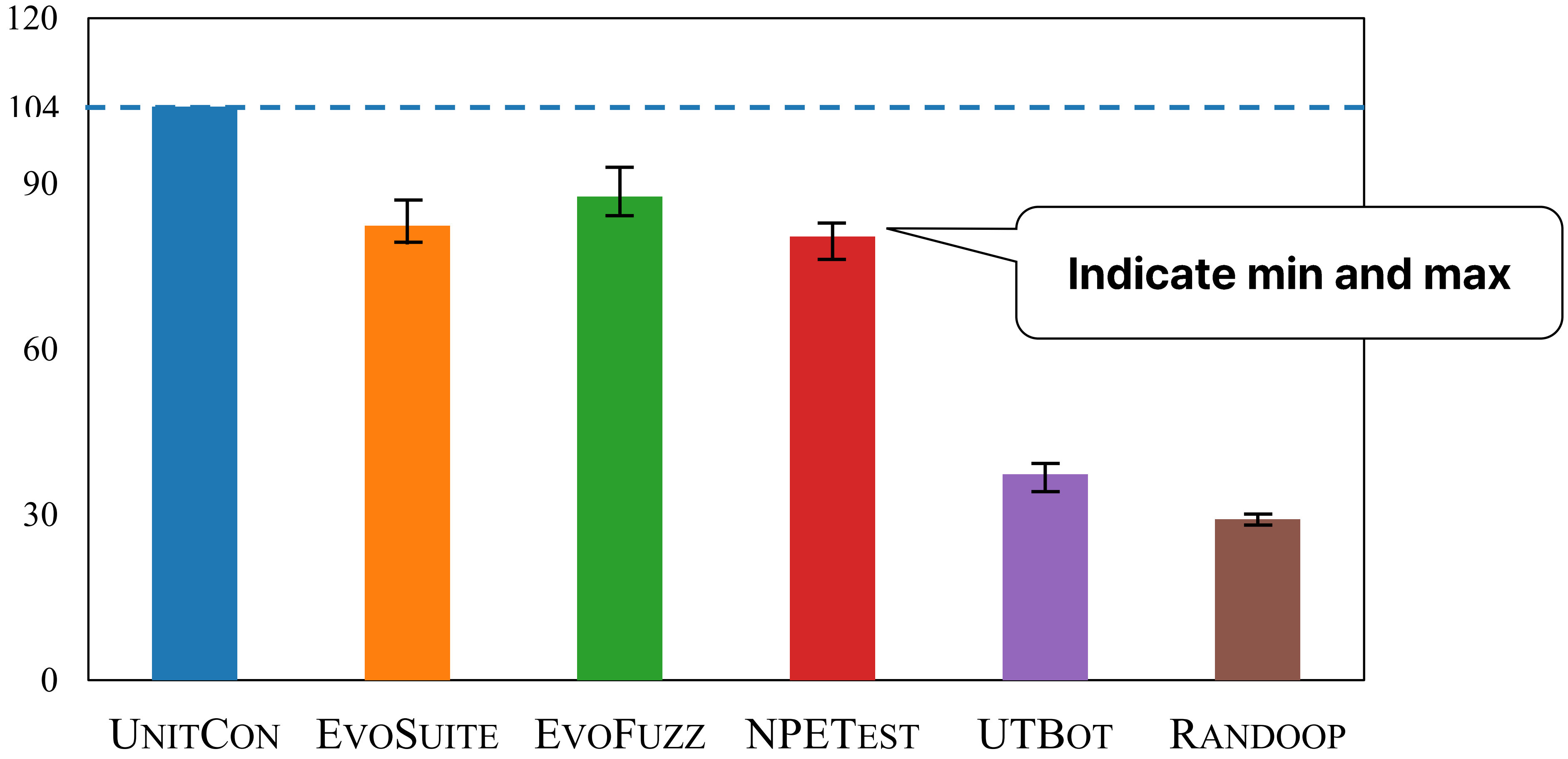
Comparison to Baselines



Indicate min and max

Comparison to Baselines

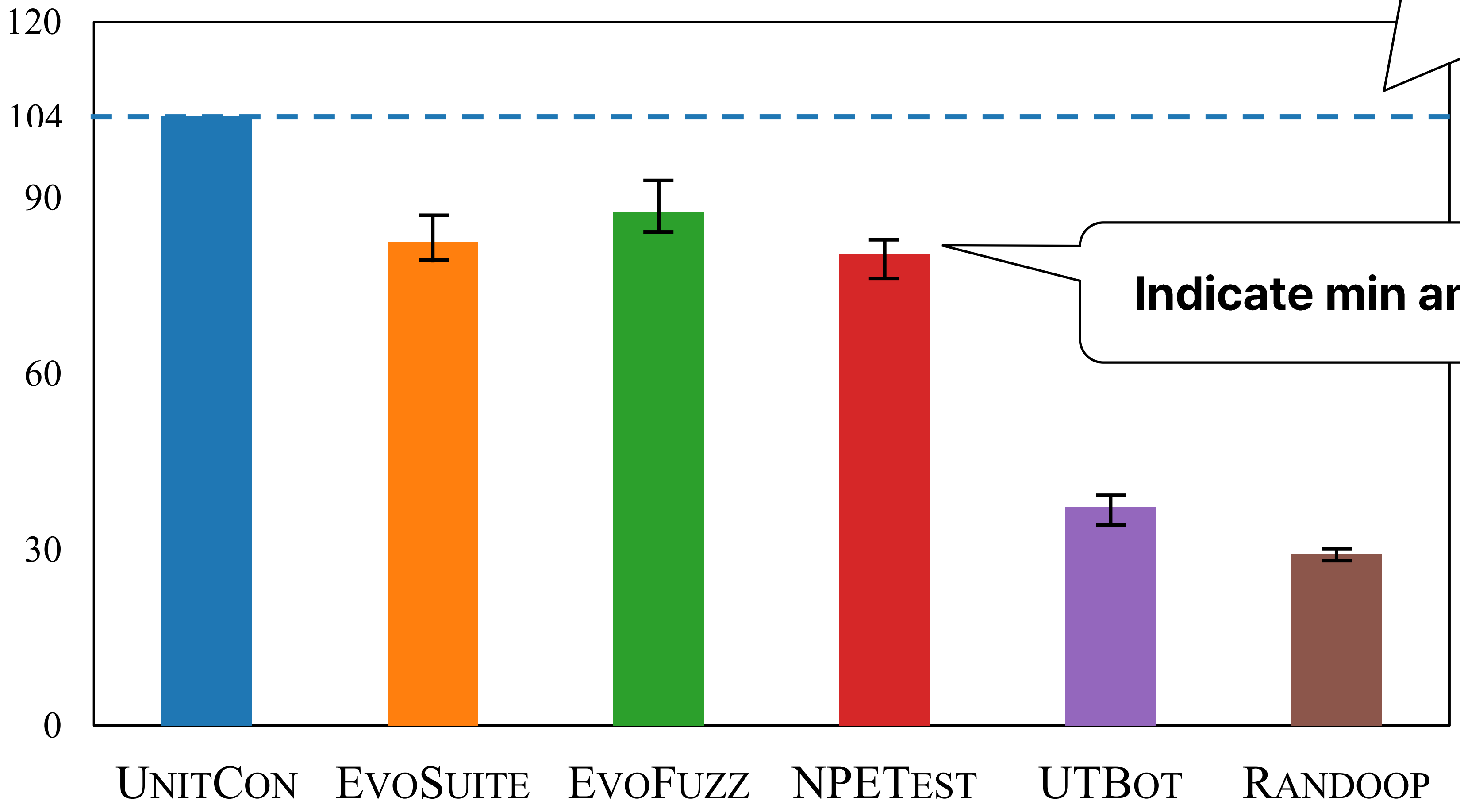
- **<span>1.2 – 3.6 x</span> more success than baselines.**



Indicate min and max

Comparison to Baselines

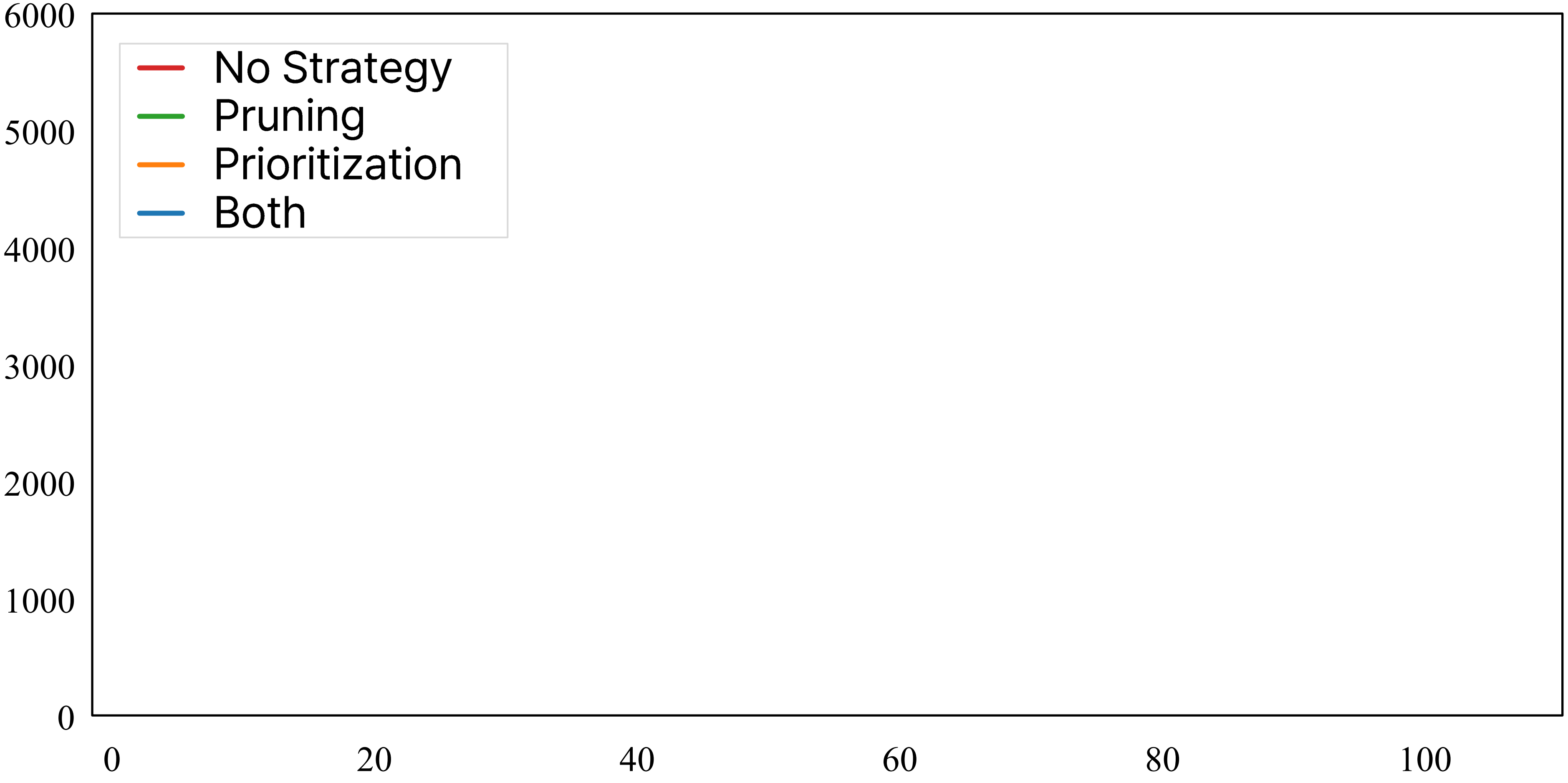
- 1.2 – 3.6 x more success than baselines.



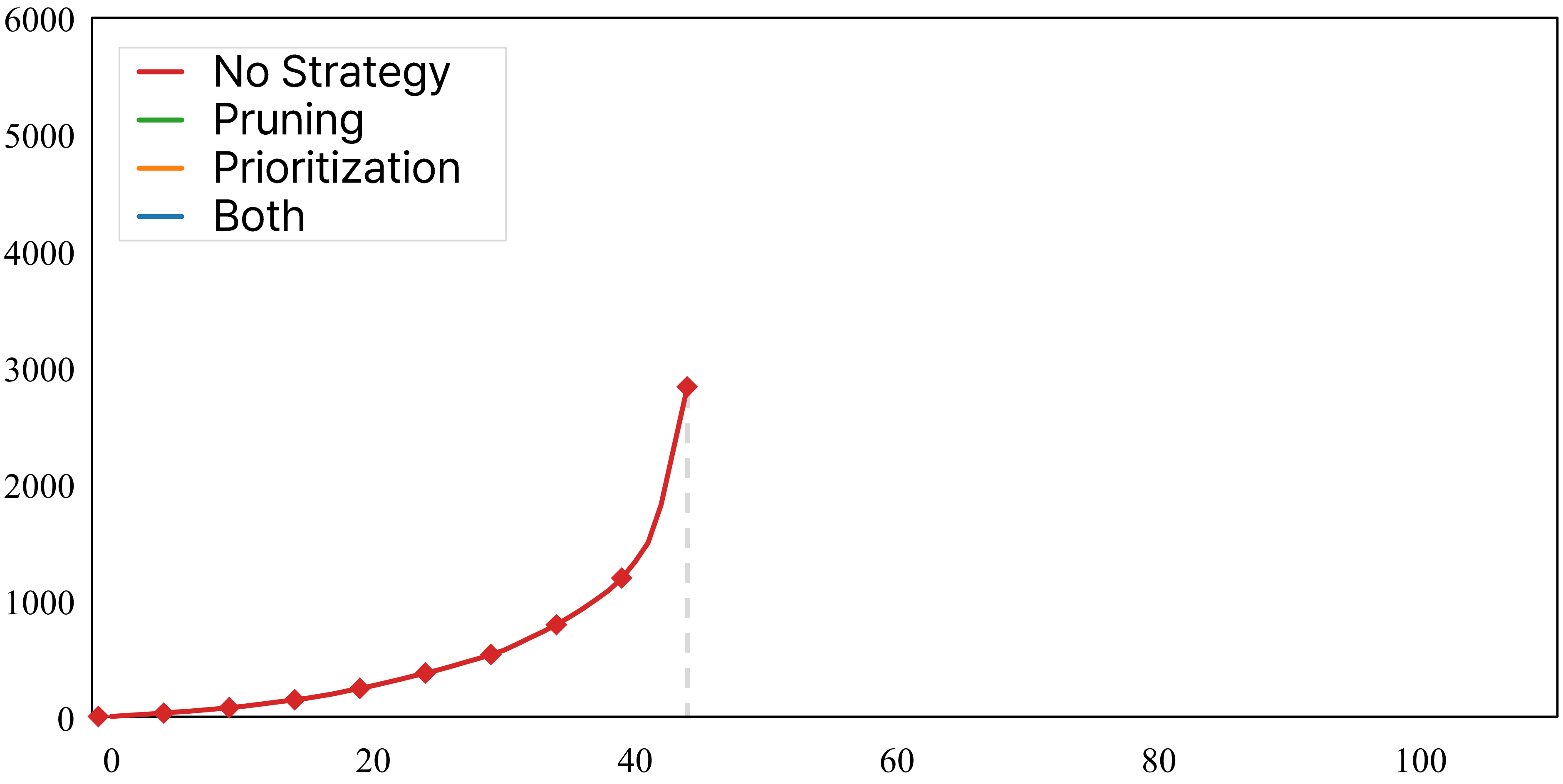
Deterministically outperform baselines.

Indicate min and max

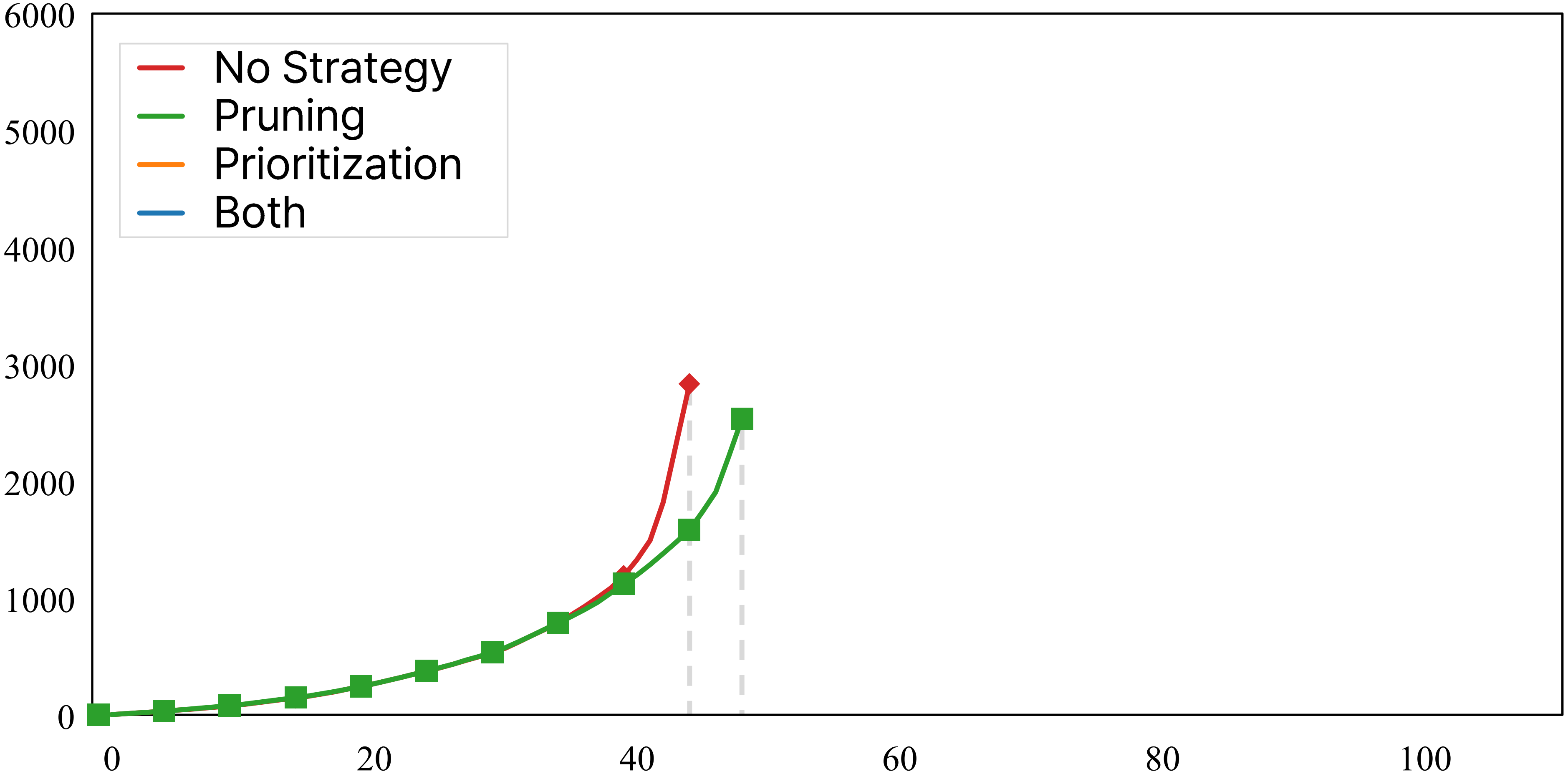
Impact of Each Strategy



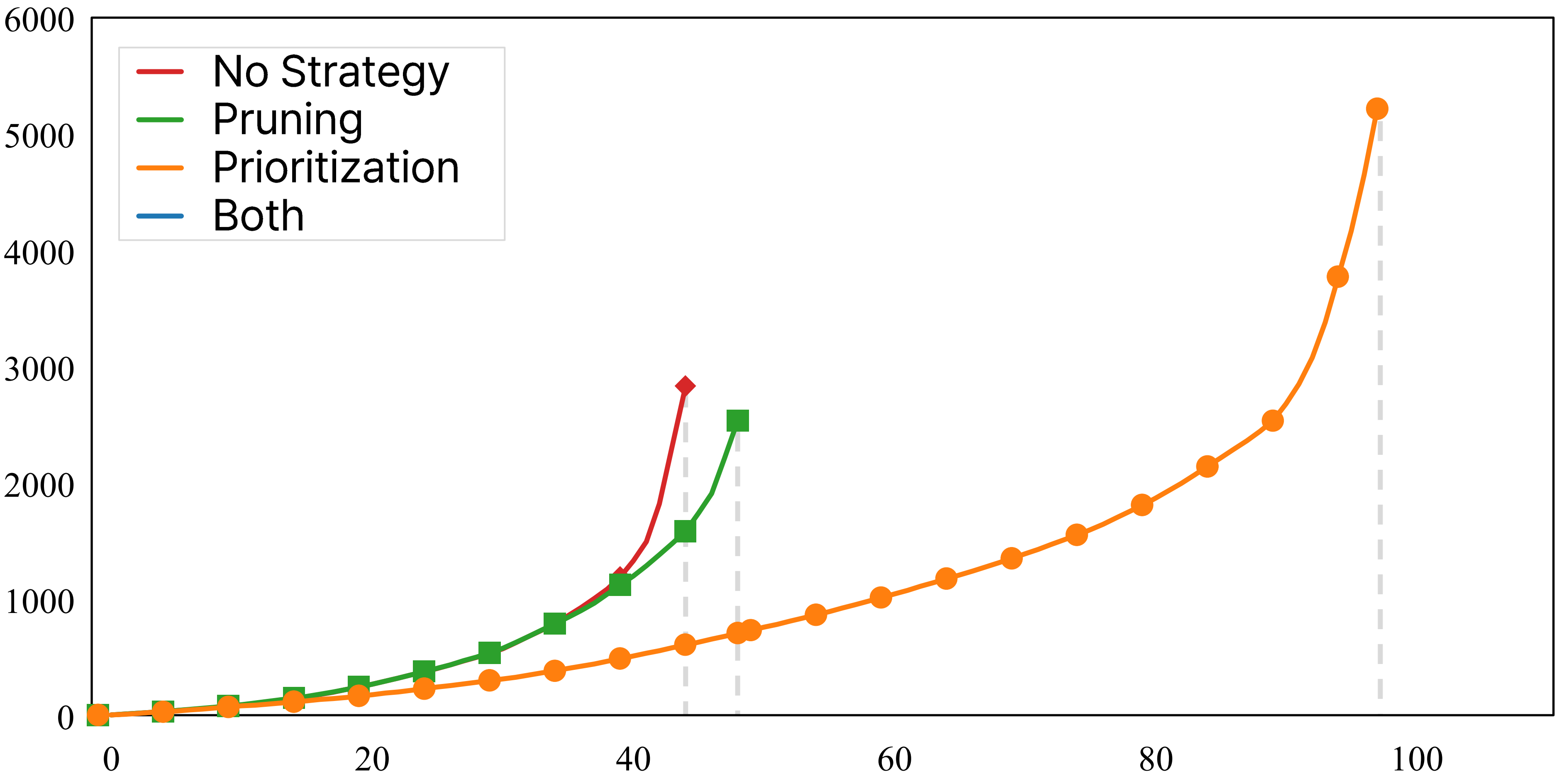
Impact of Each Strategy



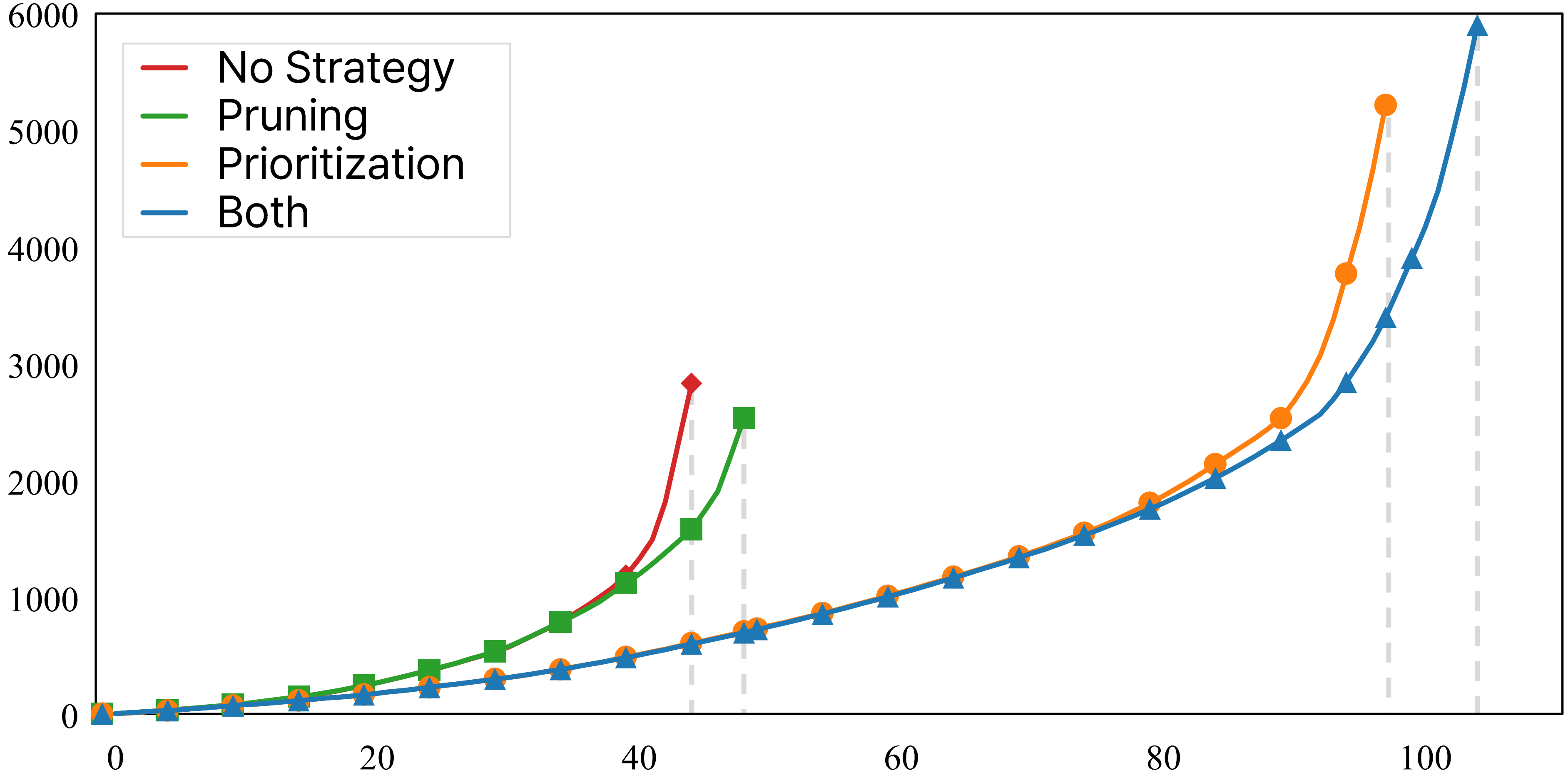
Impact of Each Strategy



Impact of Each Strategy

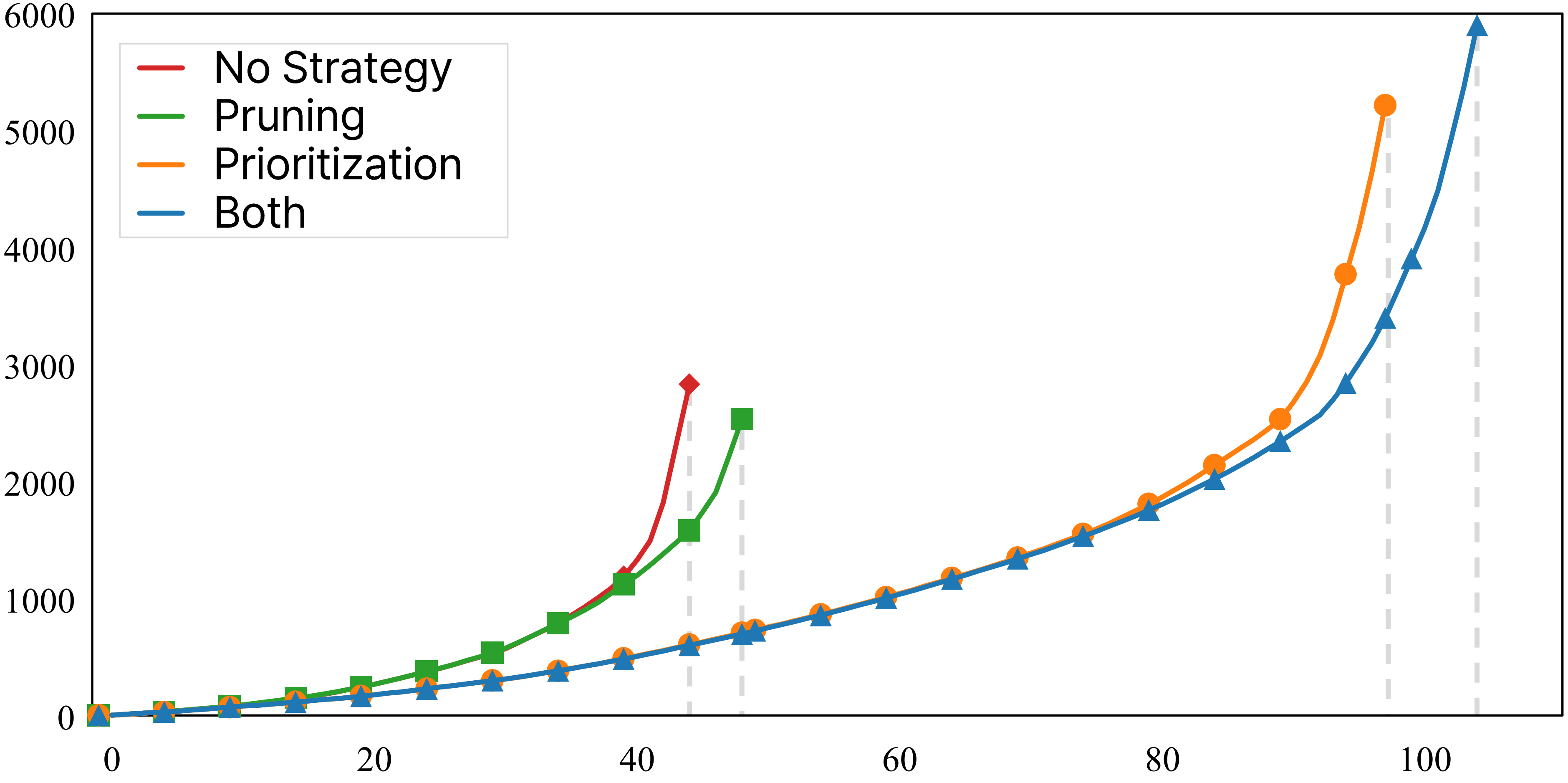


Impact of Each Strategy



Impact of Each Strategy

- **Using both strategies yields the best performance.**

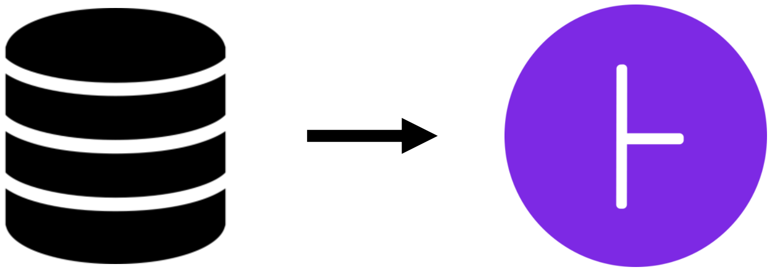


Find New Bugs

Find New Bugs

51 Java programs
(e.g., Apache, Kubernetes)

Find New Bugs



51 Java programs
(e.g., Apache, Kubernetes)

Static Analysis
(Facebook Infer)

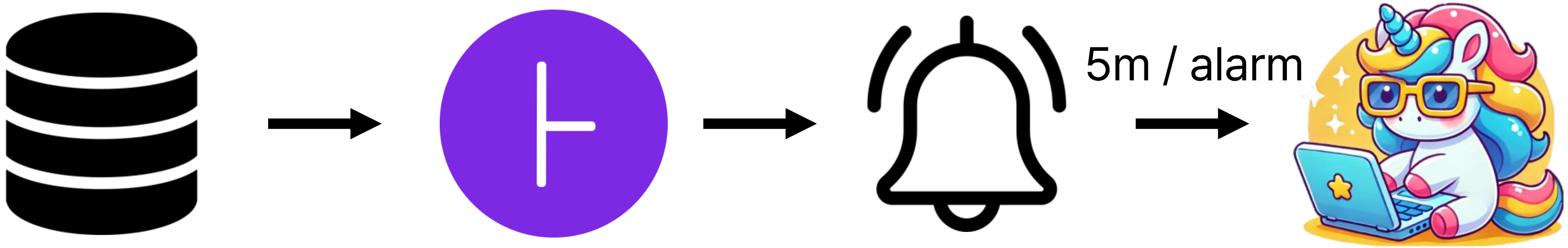
Find New Bugs

51 Java programs
(e.g., Apache, Kubernetes)

Static Analysis
(Facebook Infer)

4,290 NPE alarms

Find New Bugs



5m / alarm

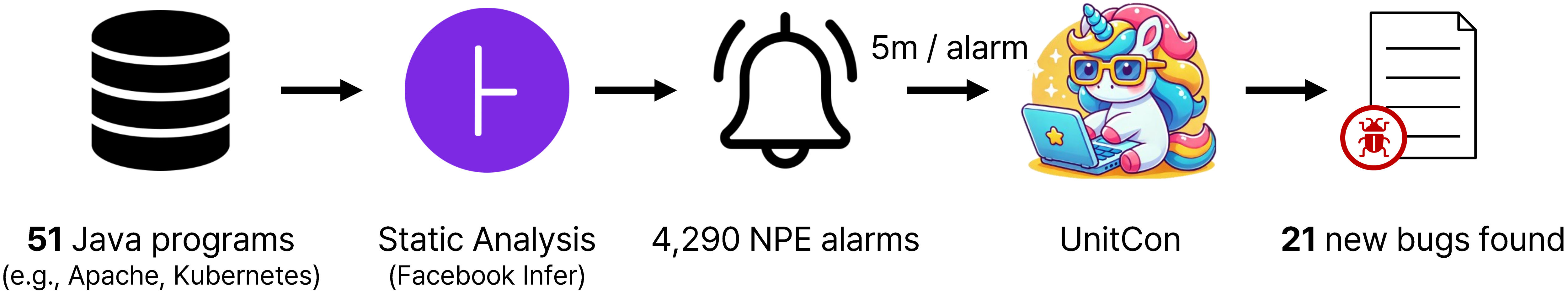
51 Java programs
(e.g., Apache, Kubernetes)

Static Analysis
(Facebook Infer)

4,290 NPE alarms

UnitCon

Find New Bugs



51 Java programs
(e.g., Apache, Kubernetes)

Static Analysis
(Facebook Infer)

4,290 NPE alarms

5m / alarm

UnitCon

21 new bugs found

Summary

Webpage & Artifact

Summary

Webpage & Artifact

- **UnitCon: Synthesizing targeted unit tests for Java runtime exceptions.**

Summary

Webpage & Artifact

- UnitCon: Synthesizing targeted unit tests for Java runtime exceptions.
- Key Idea: Guided Search via Abstract Semantics

Summary

Webpage & Artifact

- **UnitCon: Synthesizing targeted unit tests for Java runtime exceptions.**
- **Key Idea: Guided Search via Abstract Semantics**
    - Discard partial test cases with identical semantics.

Summary

Webpage & Artifact

- **UnitCon: Synthesizing targeted unit tests for Java runtime exceptions.**
- **Key Idea: Guided Search via Abstract Semantics**
  - Discard partial test cases with identical semantics.
  - Prioritize test cases that are more likely to satisfy the error conditions.

Summary

Webpage & Artifact

- **UnitCon: Synthesizing targeted unit tests for Java runtime exceptions.**
- **Key Idea: Guided Search via Abstract Semantics**
  - Discard partial test cases with identical semantics.
  - Prioritize test cases that are more likely to satisfy the error conditions.
- **Performance**

Summary

Webpage & Artifact

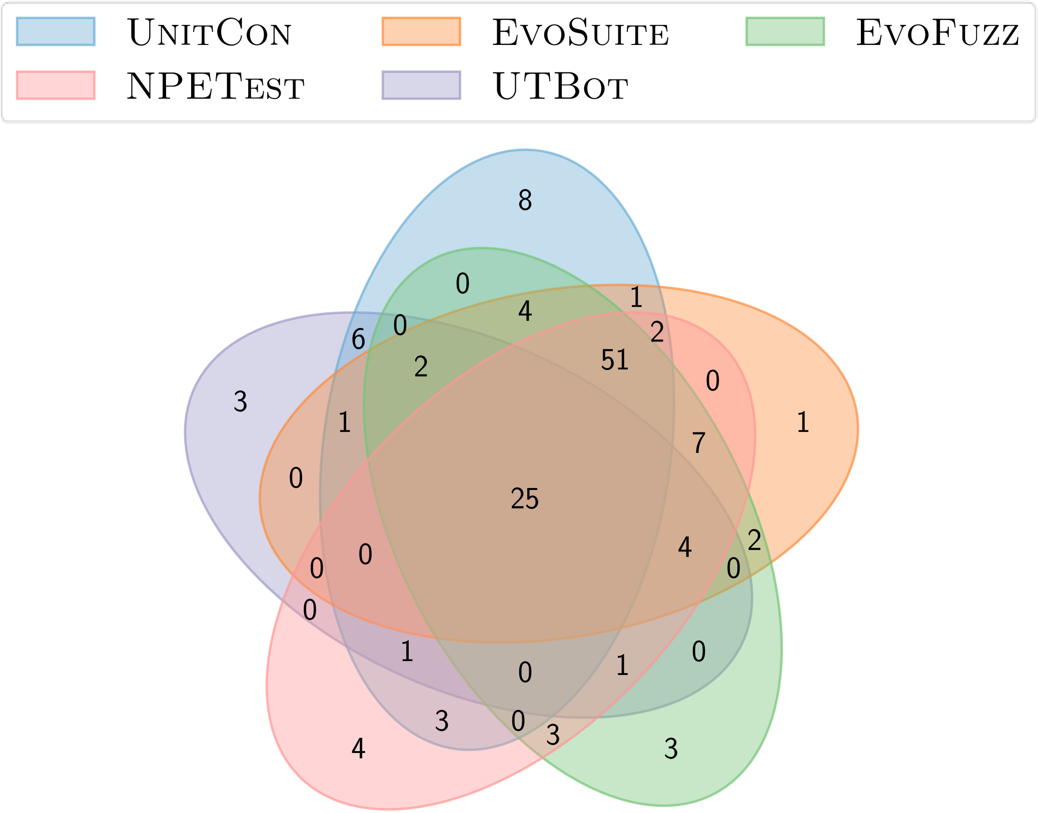
- **UnitCon: Synthesizing targeted unit tests for Java runtime exceptions.**
- **Key Idea: Guided Search via Abstract Semantics**
  - Discard partial test cases with identical semantics.
  - Prioritize test cases that are more likely to satisfy the error conditions.
- **Performance**
  - **Deterministically** reproduces up to **3.6 X** more target errors than baselines.

Summary

Webpage & Artifact

- **UnitCon: Synthesizing targeted unit tests for Java runtime exceptions.**
- **Key Idea: Guided Search via Abstract Semantics**
  - Discard partial test cases with identical semantics.
  - Prioritize test cases that are more likely to satisfy the error conditions.
- **Performance**
  - **Deterministically** reproduces up to **3.6 X** more target errors than baselines.
  - Found **21** new bugs.

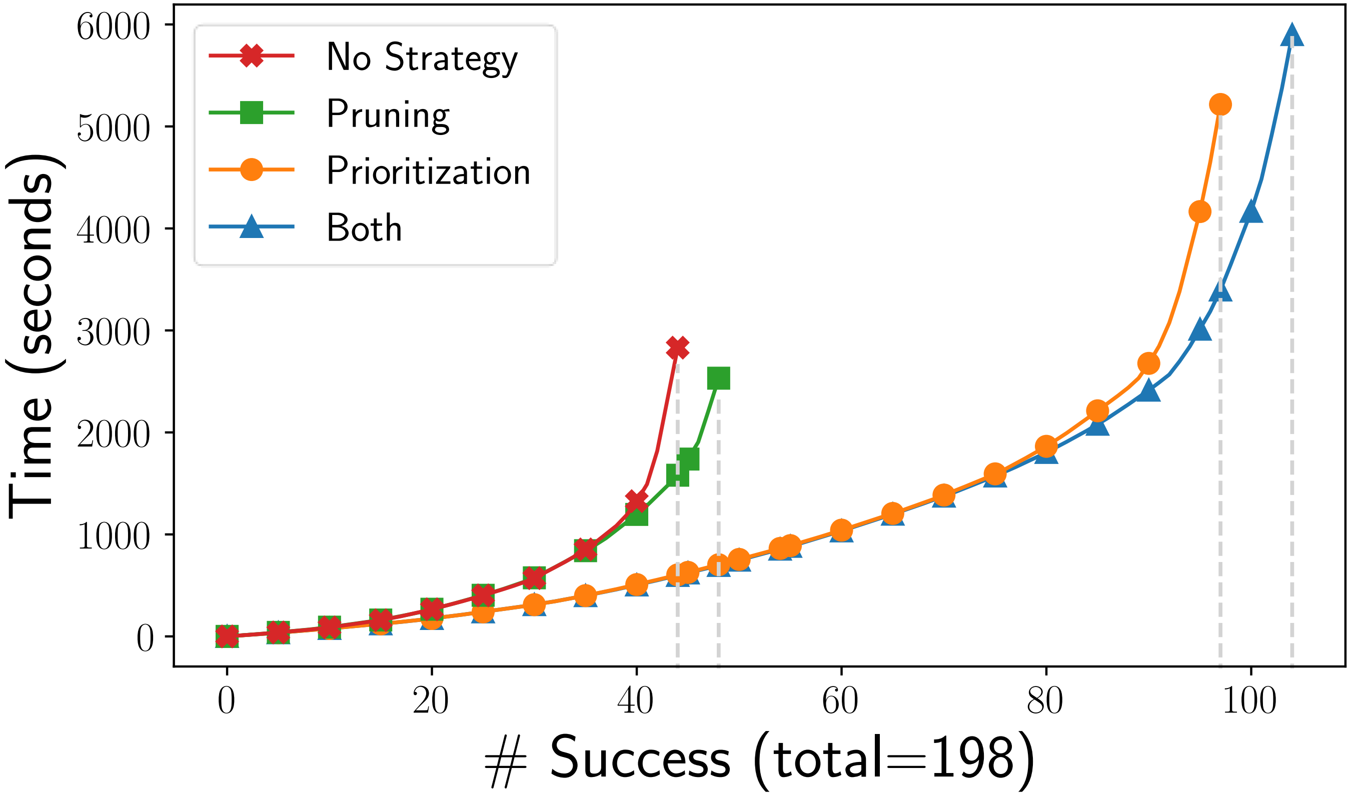
Relationship of Successful Cases Between Tools



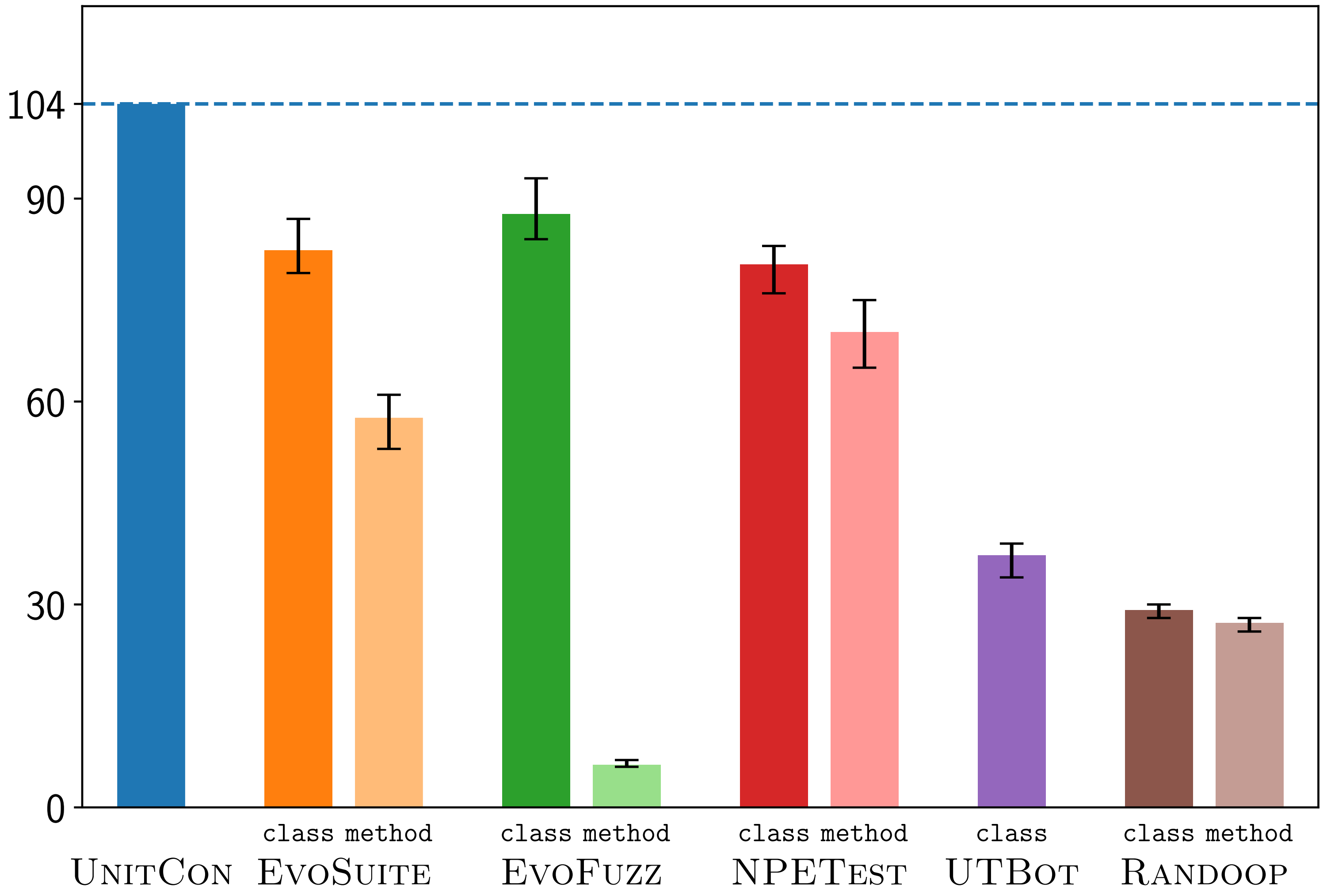
- Each tool has unique strengths.
- UnitCon uniquely reproduced the most errors.

Impact of UnitCon's Strategies

- Pruning conservatively reduces the search space.
- Make time-consuming explorations feasible.



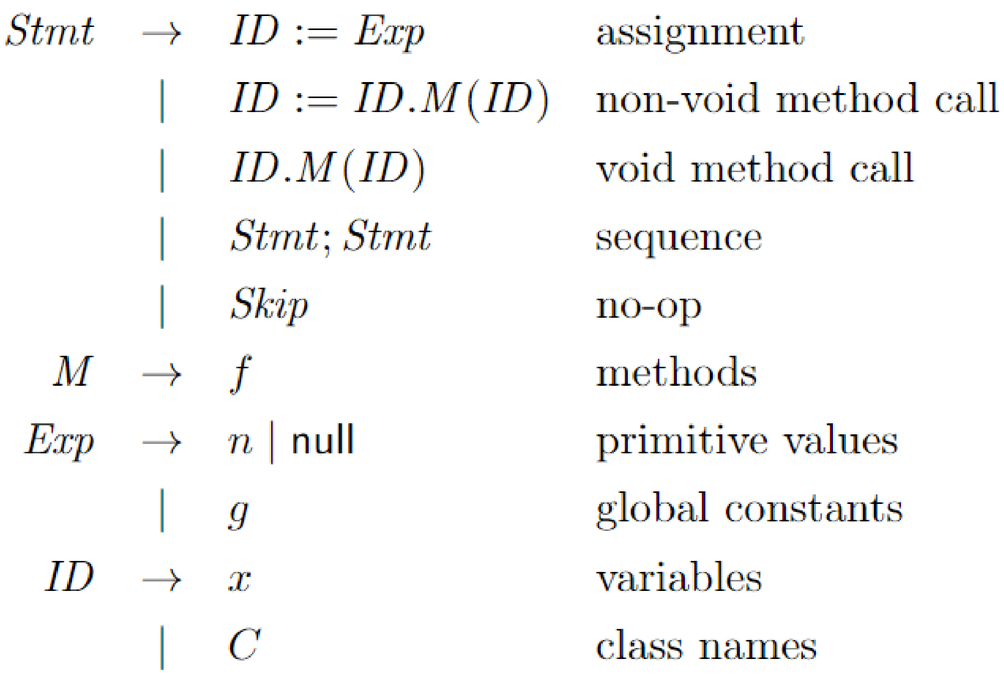
Options of baselines



Characteristics of Infer

- Under-approximate.
- Path-sensitive analyzer.
  - handle up to K (e.g., 5) distinct paths per method.
- Indirect calls are handled imprecisely.
  - e.g., overriding, abstract class

Domain-Specific Language



$$\begin{array}{rcll}
\mathit{Stmt} & \rightarrow & \mathit{ID} := \mathit{Exp} & \text{assignment} \\
 & \mid & \mathit{ID} := \mathit{ID}.M(\mathit{ID}) & \text{non-void method call} \\
 & \mid & \mathit{ID}.M(\mathit{ID}) & \text{void method call} \\
 & \mid & \mathit{Stmt}; \mathit{Stmt} & \text{sequence} \\
 & \mid & \mathit{Skip} & \text{no-op} \\
M & \rightarrow & f & \text{methods} \\
\mathit{Exp} & \rightarrow & n \mid \textsf{null} & \text{primitive values} \\
 & \mid & g & \text{global constants} \\
\mathit{ID} & \rightarrow & x & \text{variables} \\
 & \mid & C & \text{class names}
\end{array}$$

- Define a DSL to support Java at the source-code level in a simple yet powerful way.

Limitation of UnitCon

- No special limitations tied to specific types of bugs.
- However, it struggles with cases that require **strings** or **numbers** not already present in the program.
    - UnitCon’s goal: **deterministically** synthesizing the test cases.

Reliance on Static Analysis

9 NPE alarms

21 new bugs found

- Found the new bugs that the static analyzer was unable to find.
- Program analysis vs. Program synthesis