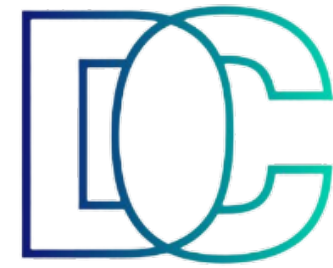


# Applying Learning Techniques to Oracle Synthesis

Facundo Molina

Advisor: Dr. Nazareno Aguirre

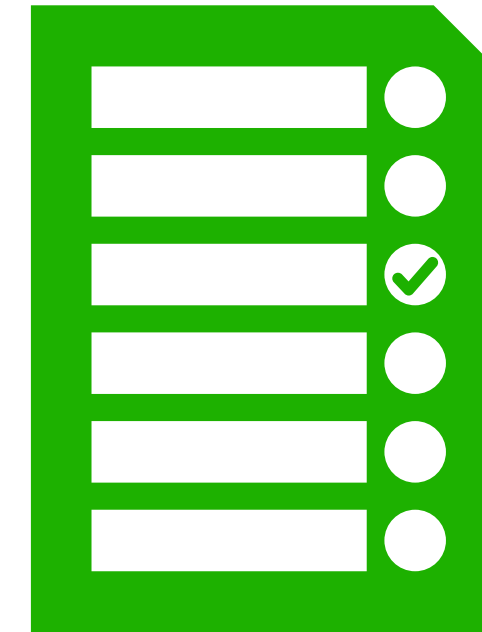
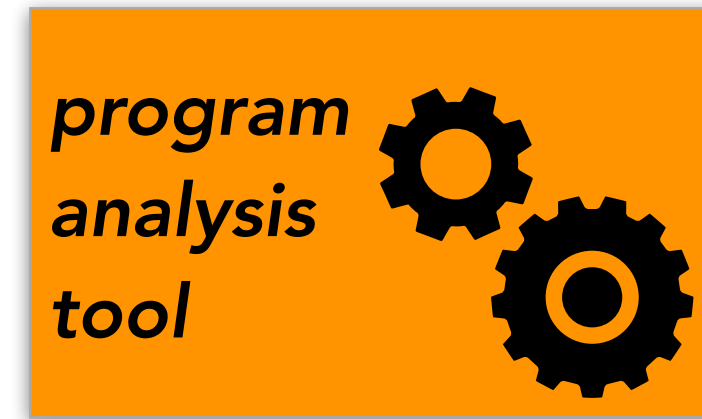
Department of Computer Science, University of Rio Cuarto, Argentina  
CONICET, Argentina



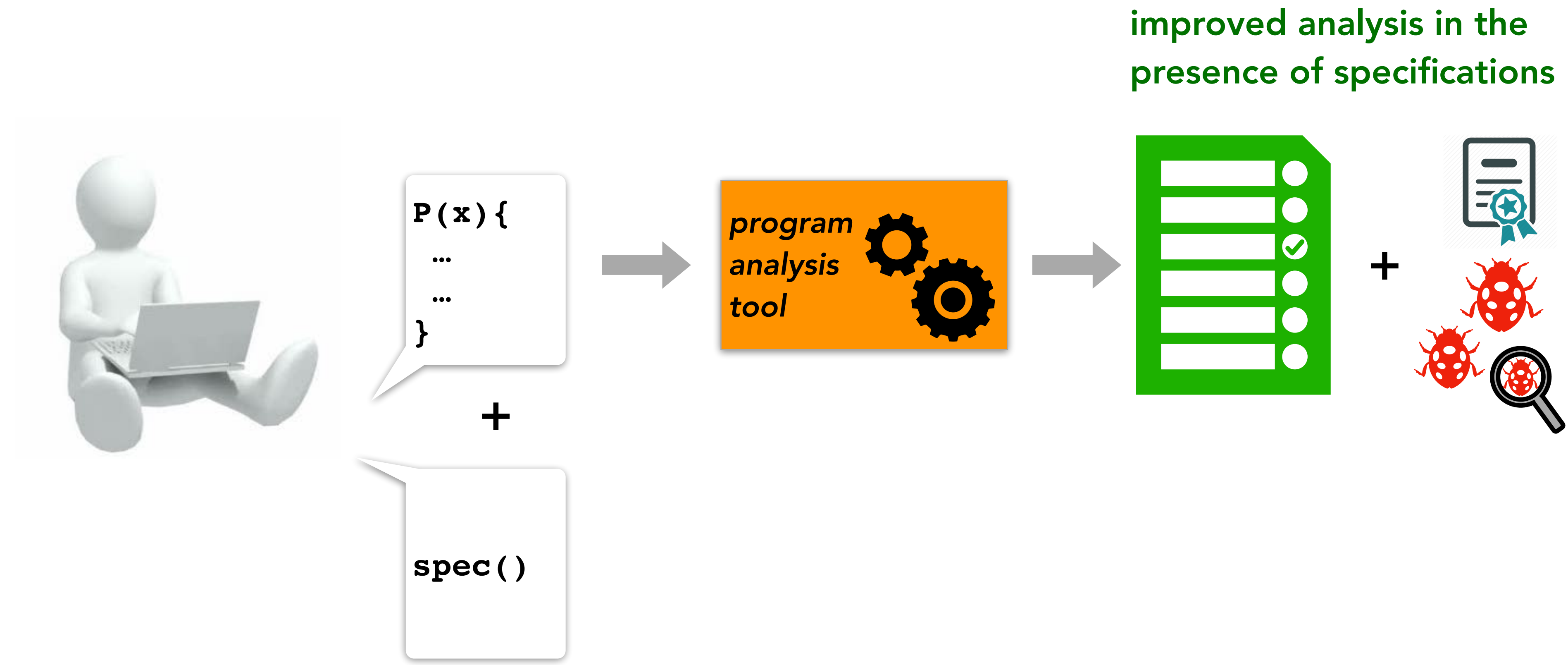
# An Automated Analysis Scenario



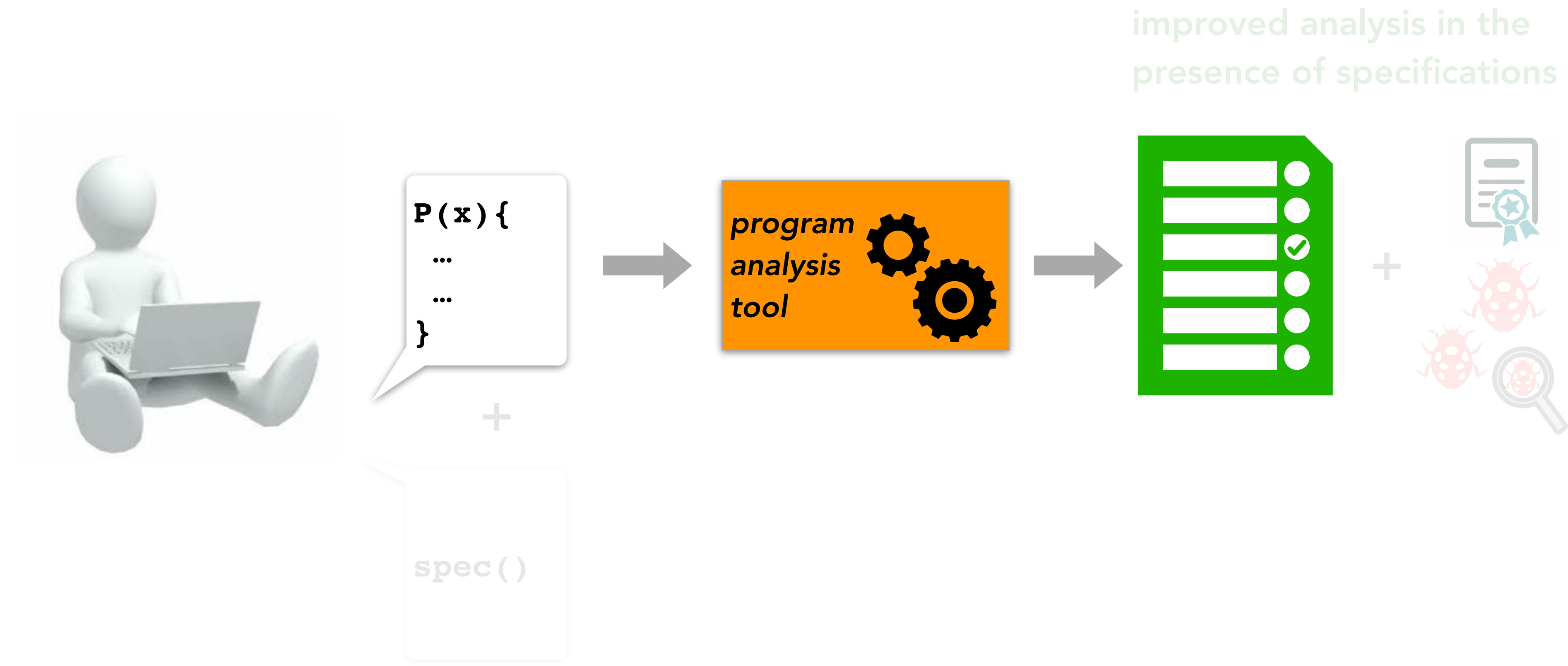
```
P(x) {  
  ...  
  ...  
}
```



# An Automated Analysis Scenario

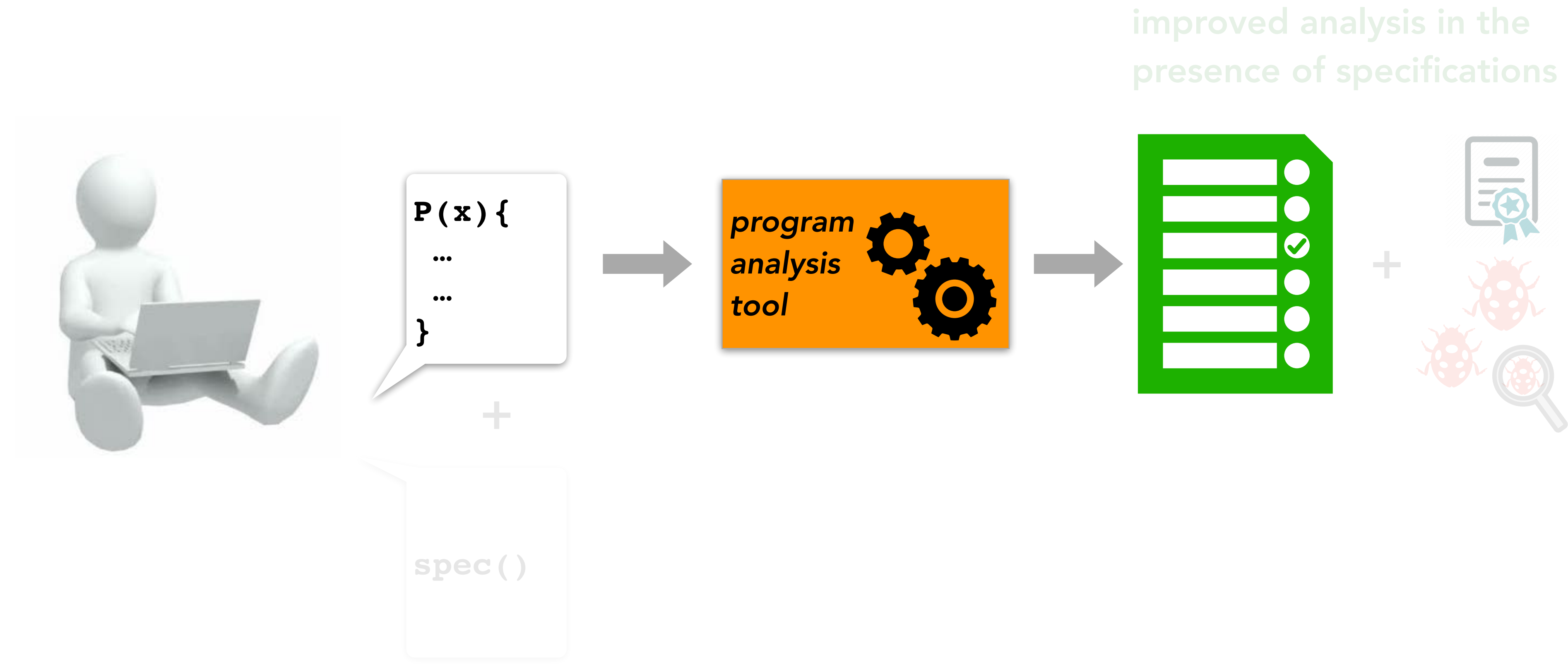


# An Automated Analysis Scenario



**unfortunately, specifications  
are often unavailable**

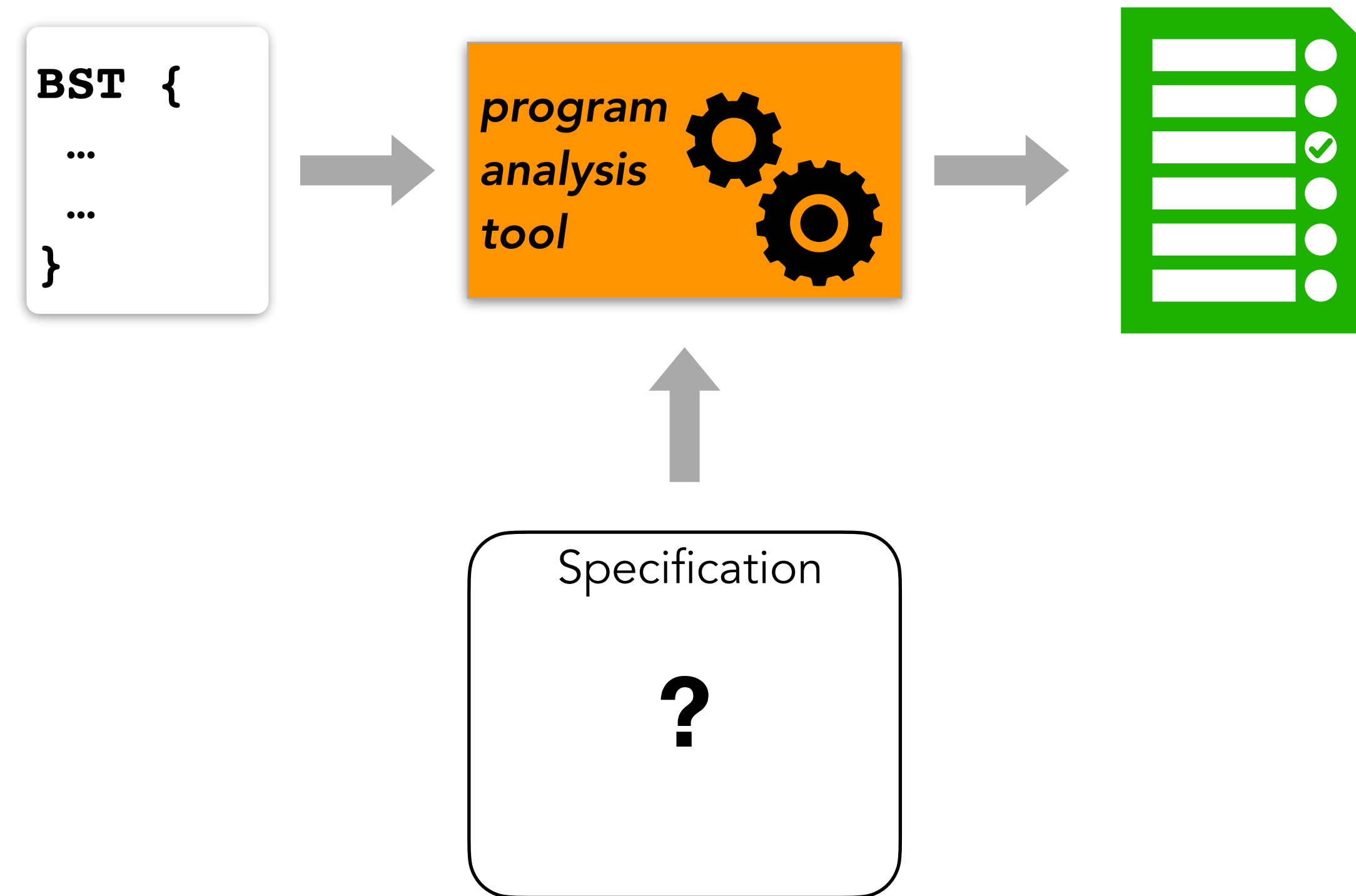
# An Automated Analysis Scenario



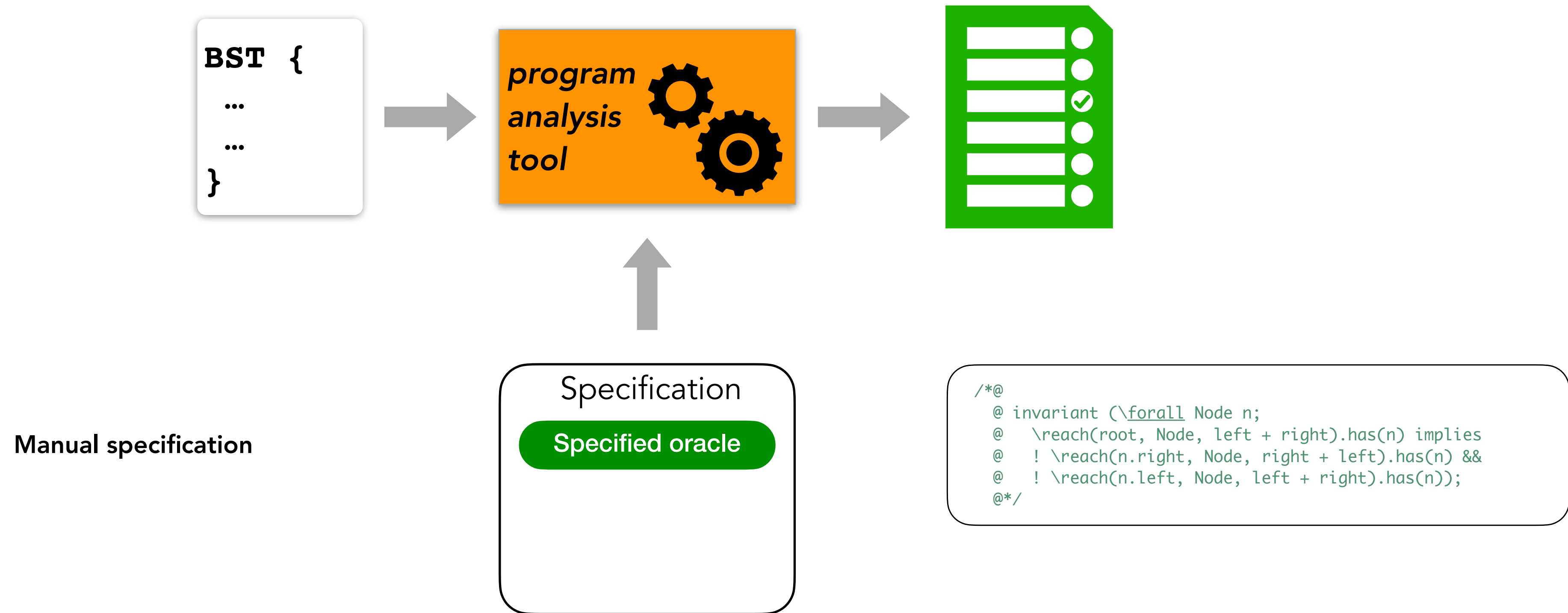
unfortunately, specifications  
are often unavailable

This emphasizes the relevance of the *oracle problem*

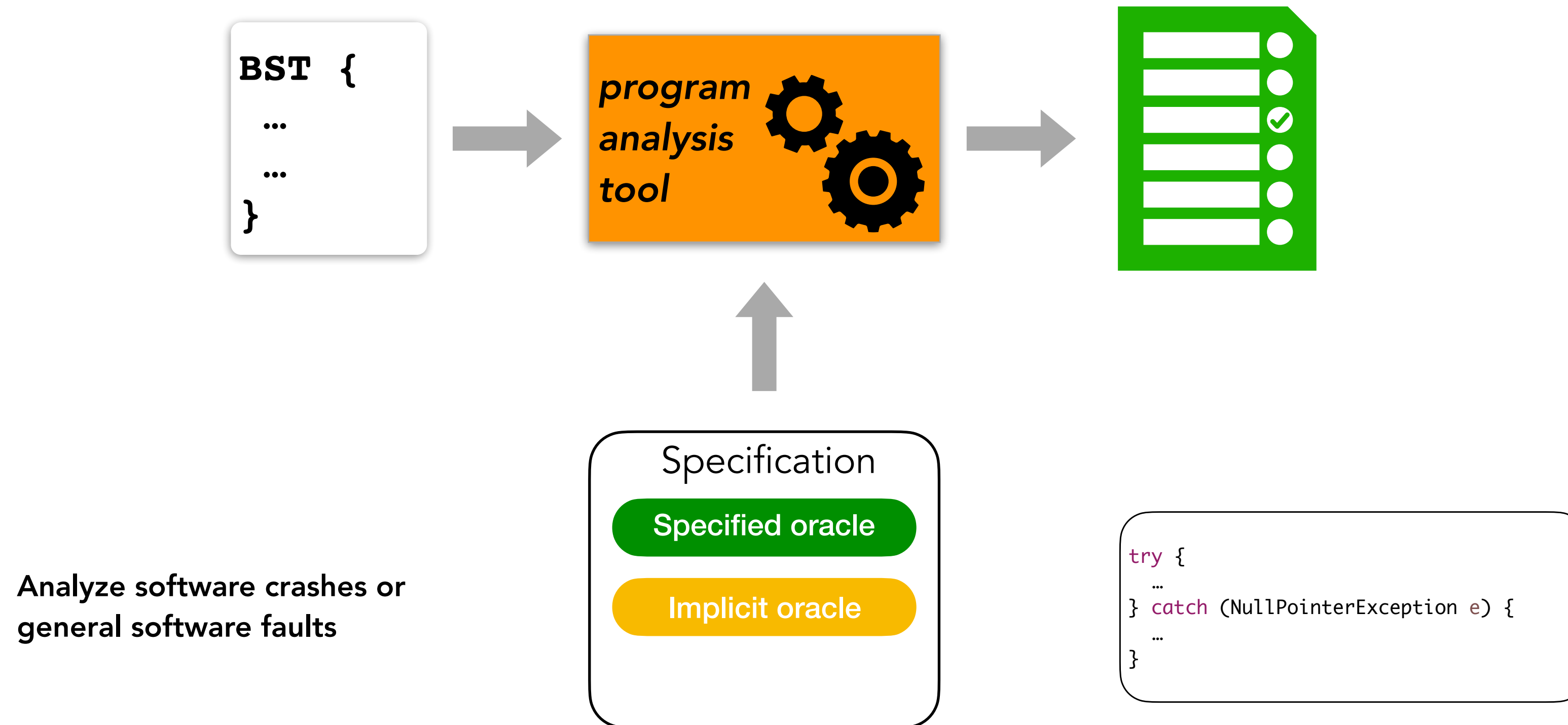
# Approaches to the Oracle Problem



# Approaches to the Oracle Problem

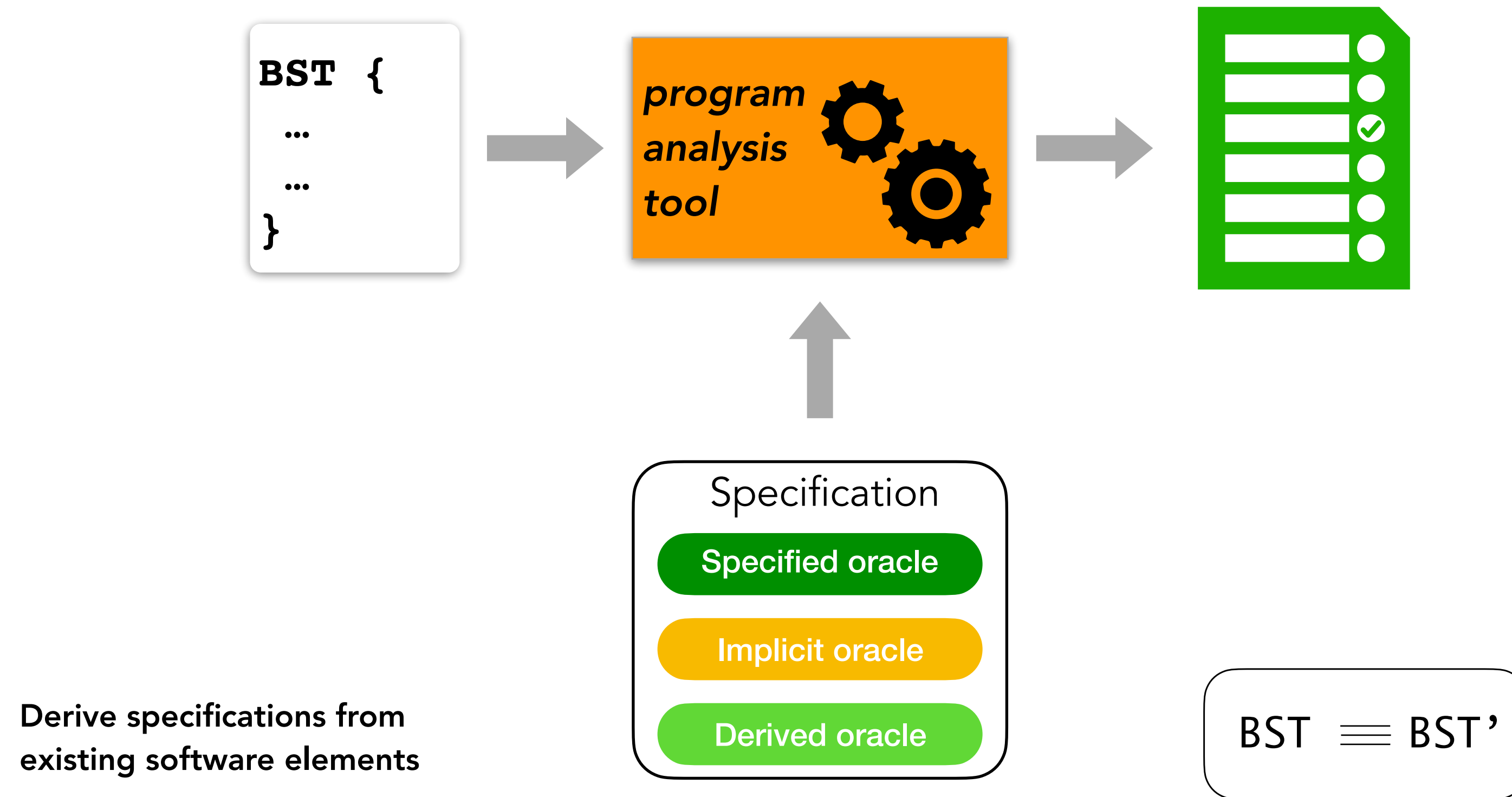


# Approaches to the Oracle Problem

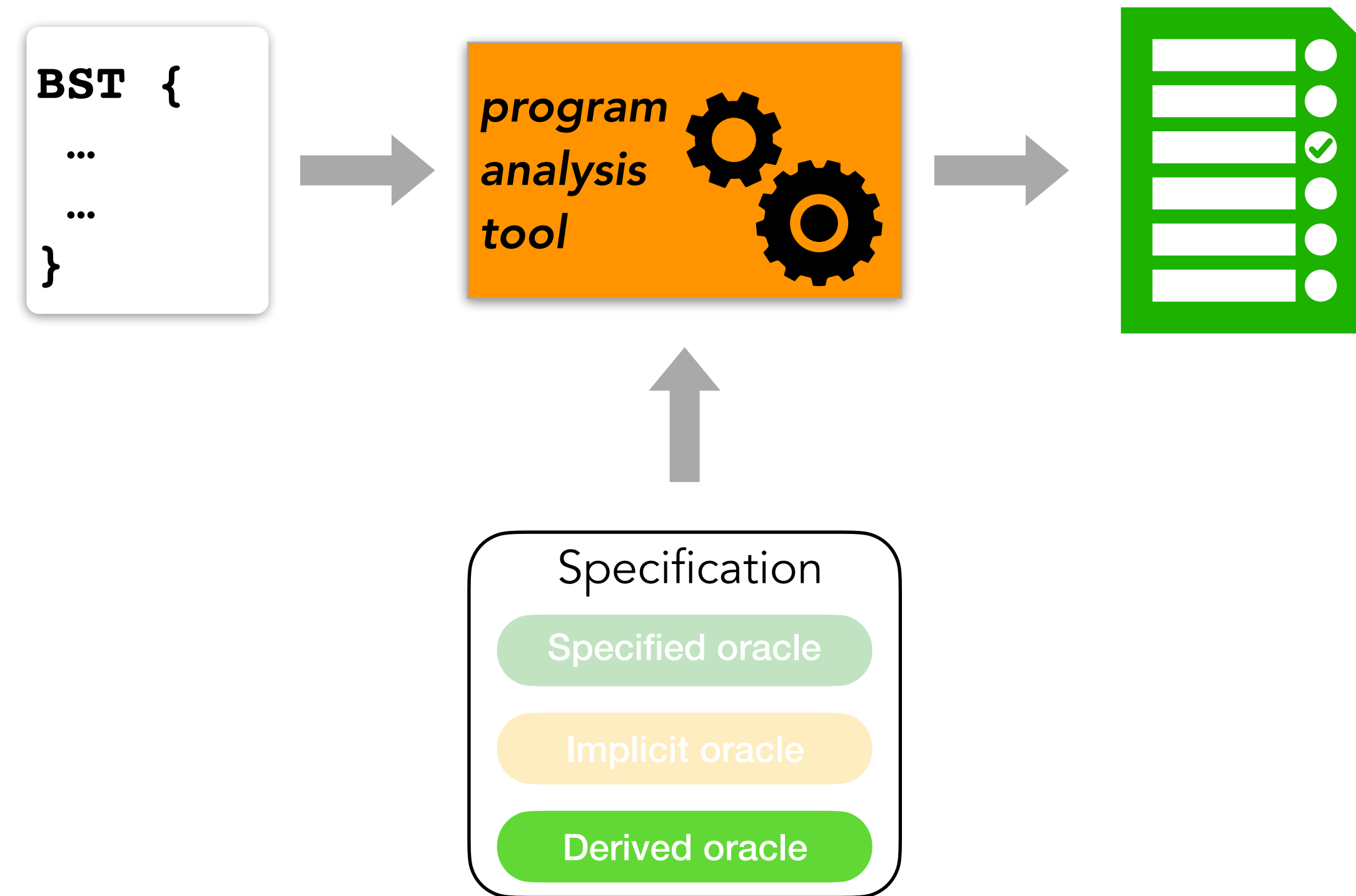




# Approaches to the Oracle Problem



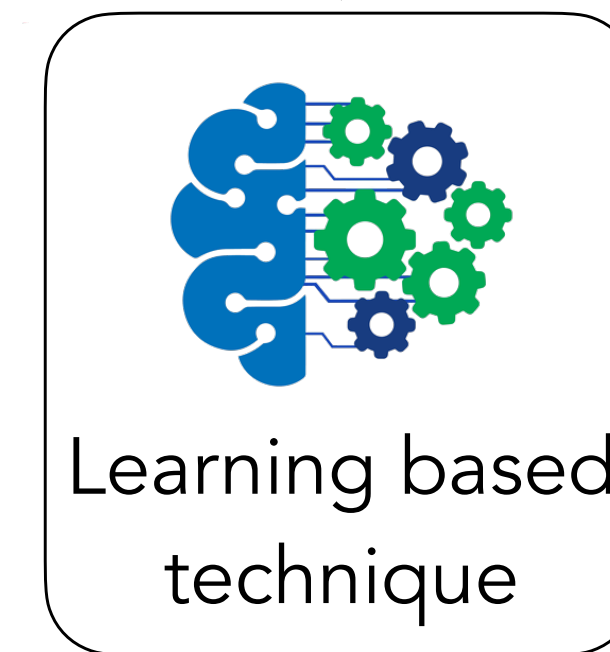
# Approaches to the Oracle Problem



# Our Approach to the Oracle Problem

The application of learning based techniques:

```
P(x) {  
  ...  
  ...  
}
```



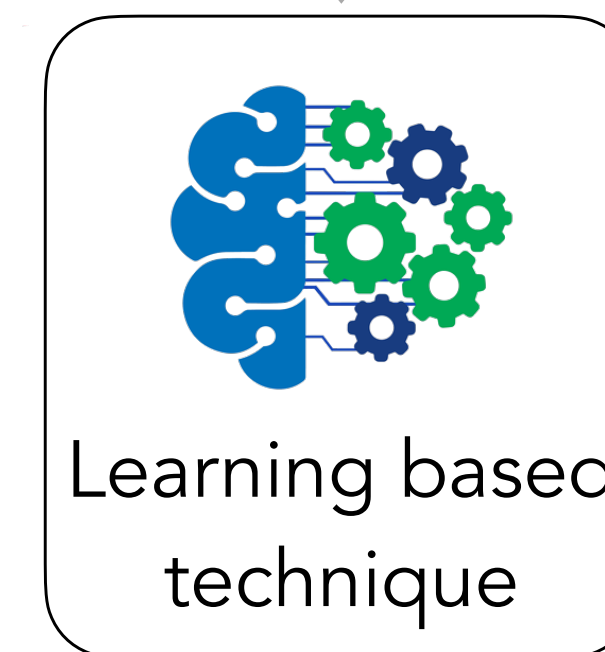
```
spec()
```

# Our Approach to the Oracle Problem

The application of learning based techniques:

- Neural Networks to capture Data Structure Invariants

```
P(x) {  
  ...  
  ...  
}
```



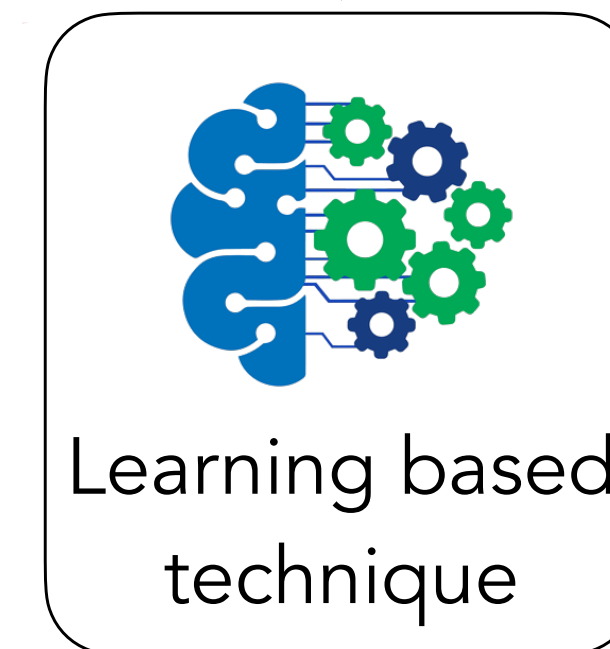
```
spec()
```

# Our Approach to the Oracle Problem

The application of learning based techniques:

- Neural Networks to capture Data Structure Invariants
- Genetic Algorithms for Learning Postconditions

```
P(x) {  
  ...  
  ...  
}
```



```
spec()
```

# Using Neural Networks to capture Data Structure Invariants

What is a data structure invariant (or representation invariant) for class C?

```
public class C {  
    public void C() {  
        ...  
    }  
  
    public void m1(...) {  
        ...  
    }  
  
    public int m2() {  
        ...  
    }  
}
```

# Using Neural Networks to capture Data Structure Invariants

What is a data structure invariant (or representation invariant) for class C?

```
public class C {  
    public void C() {  
        ...  
    }  
  
    public void m1(...) {  
        ...  
    }  
  
    public int m2() {  
        ...  
    }  
}
```

[Liskov 2000]: a statement of a property that all legitimate objects satisfy is called a **representation invariant**, or rep invariant.

A rep invariant **I** is a predicate

**I: C -> boolean**

that is true of legitimate objects.

# Using Neural Networks to capture Data Structure Invariants

```
public class BST {  
  
    public void BST() {  
  
    }  
  
    public void insert(...)  
    {  
        ...  
    }  
  
    public void remove()  
    {  
        ...  
    }  
    ...  
}
```

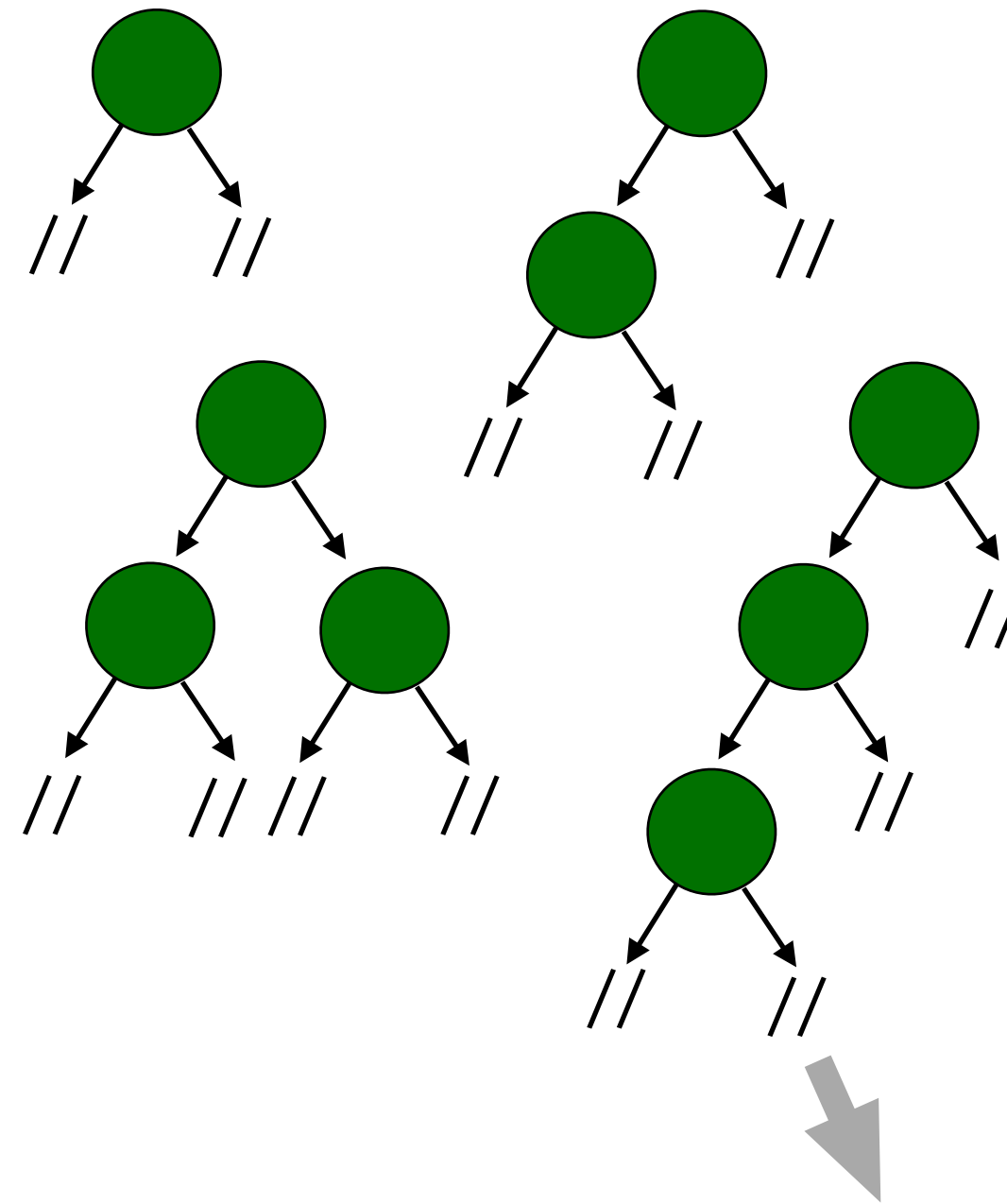
```
/*@  
  @ invariant (\forall Node n;  
  @   \reach(root, Node, left + right).has(n) implies  
  @   ! \reach(n.right, Node, right + left).has(n) &&  
  @   ! \reach(n.left, Node, left + right).has(n));  
  @*/
```



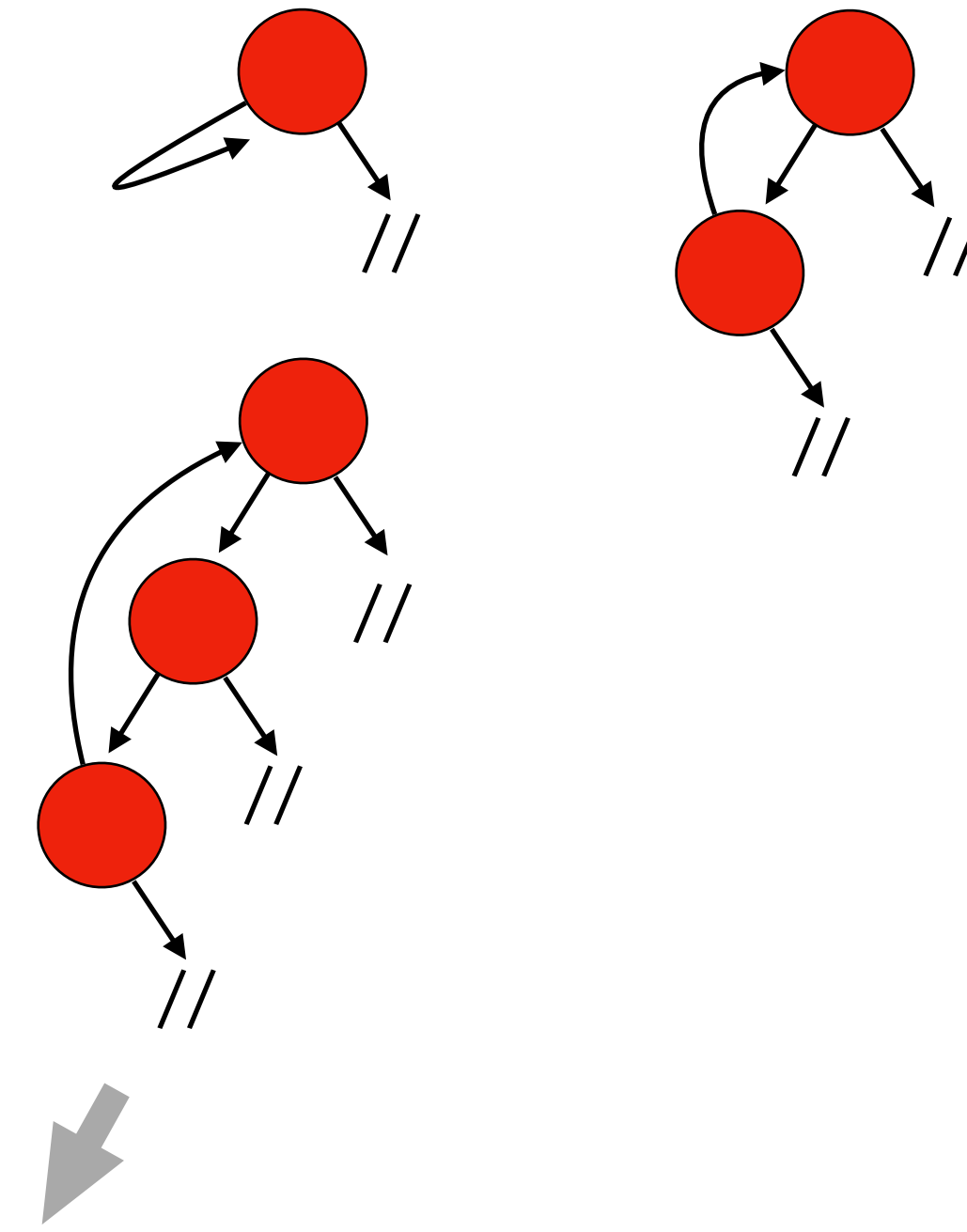
# Using Neural Networks to capture Data Structure Invariants

```
public class BST {  
    public void BST() {  
    }  
  
    public void insert(...)  
    {  
    ...  
    }  
  
    public void remove()  
    {  
    ...  
    }  
    ...  
}
```

legitimate objects



illegitimate objects



```
/*@  
 @ invariant (\forall n; Node n;  
 @ \reach(root, Node, left + right).has(n) implies  
 @ ! \reach(n.right, Node, right + left).has(n) &&  
 @ ! \reach(n.left, Node, left + right).has(n));  
 @*/
```

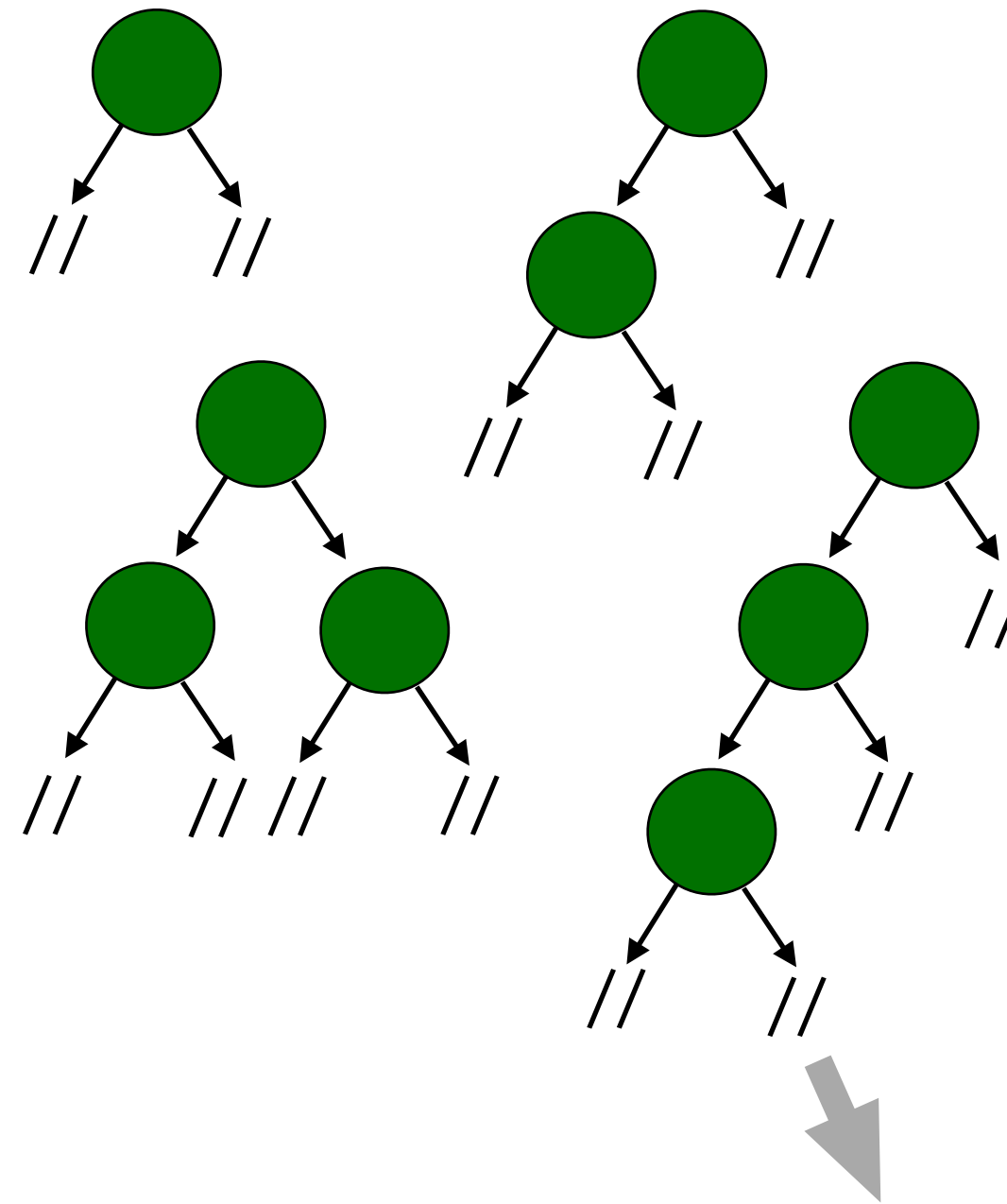
✓ return true

✗ return false

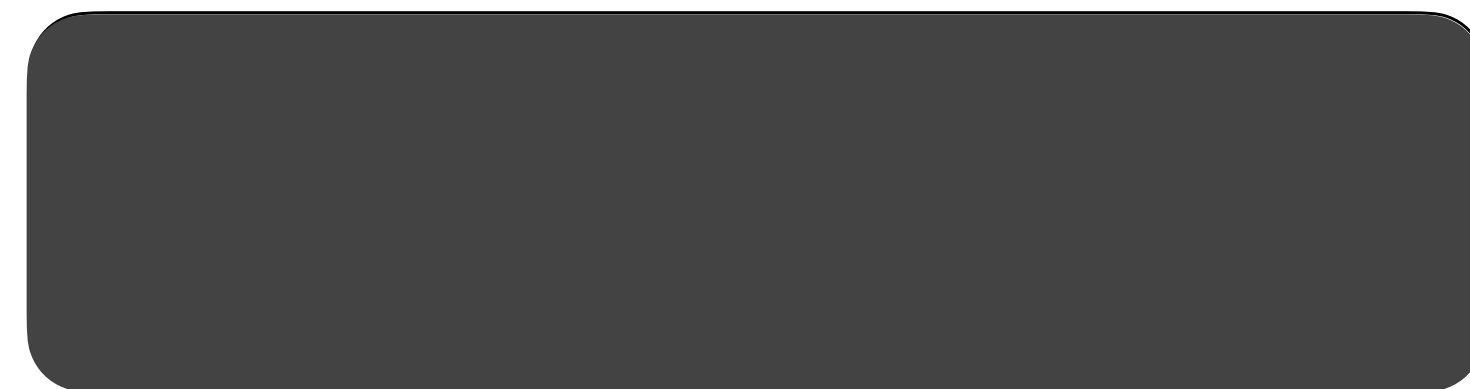
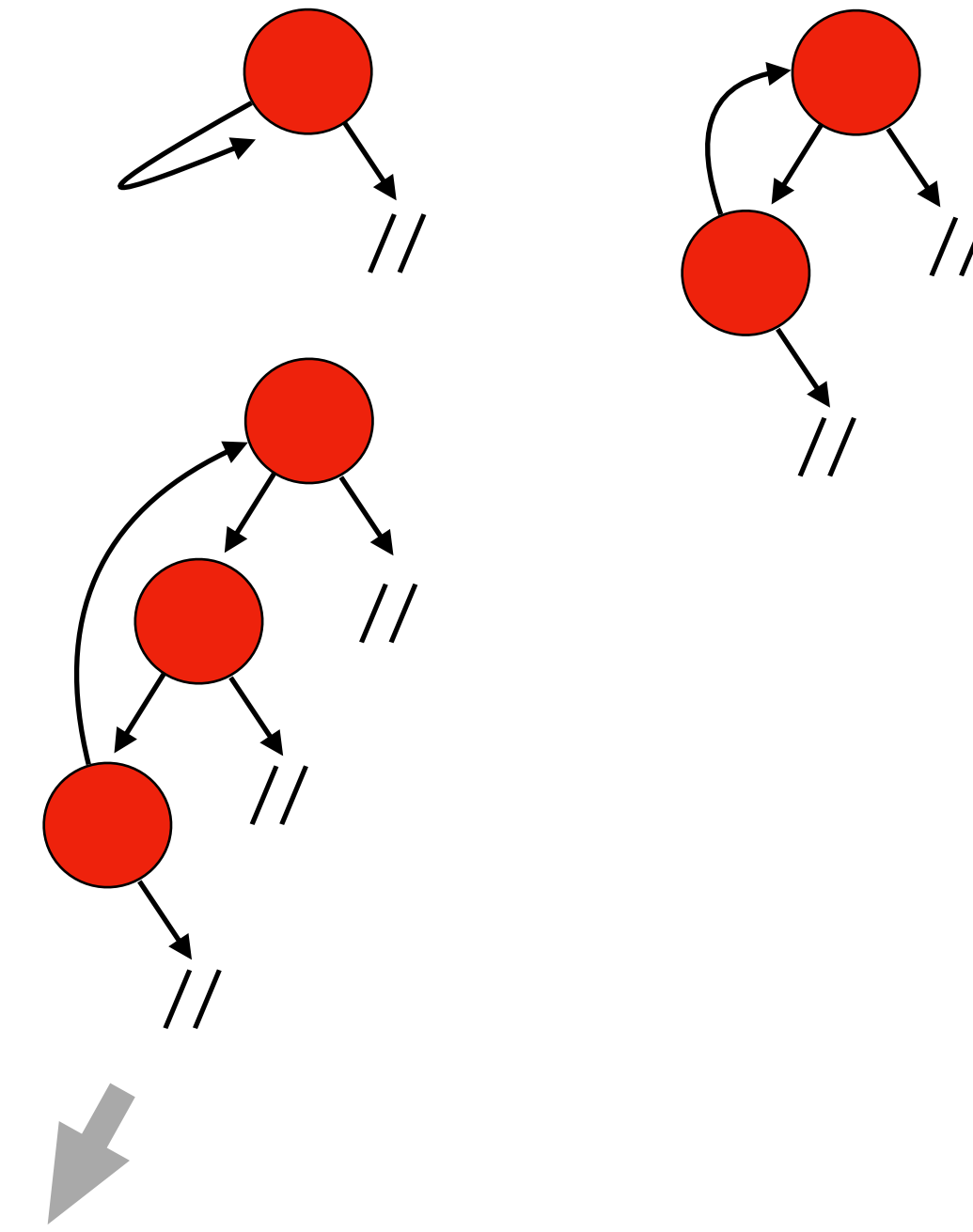
# Using Neural Networks to capture Data Structure Invariants

```
public class BST {  
  public void BST() {  
  }  
  public void insert(...)  
  {  
  ...  
  }  
  public void remove()  
  {  
  ...  
  }  
  ...  
}
```

legitimate objects



illegitimate objects



✓ return true

✗ return false

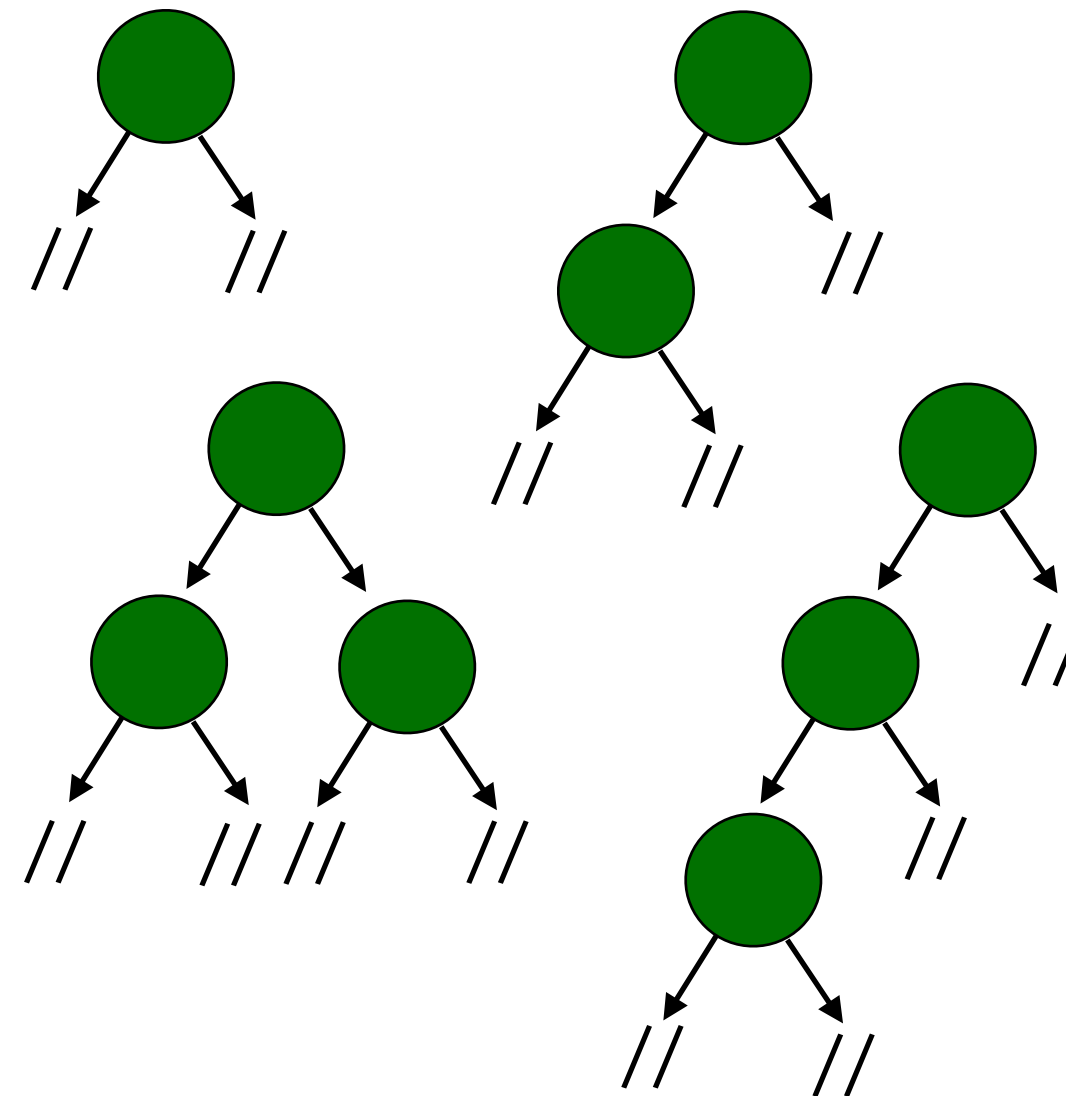
# Using Neural Networks to capture Data Structure Invariants

```
public class BST {  
    public void BST() {  
    }  
  
    public void insert(...)  
    {  
    ...  
    }  
  
    public void remove()  
    {  
    ...  
    }  
    ...  
}
```

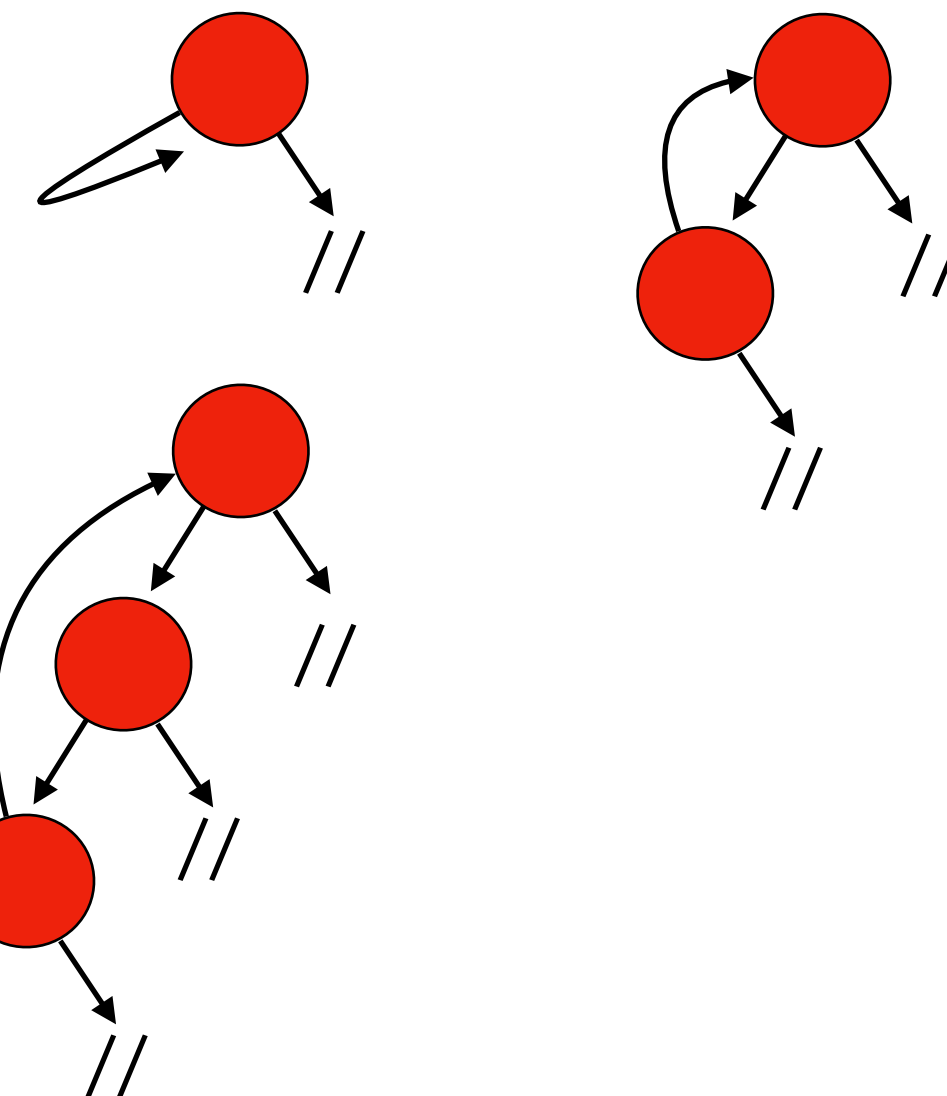
Train a neural network to capture data structure invariants



legitimate objects

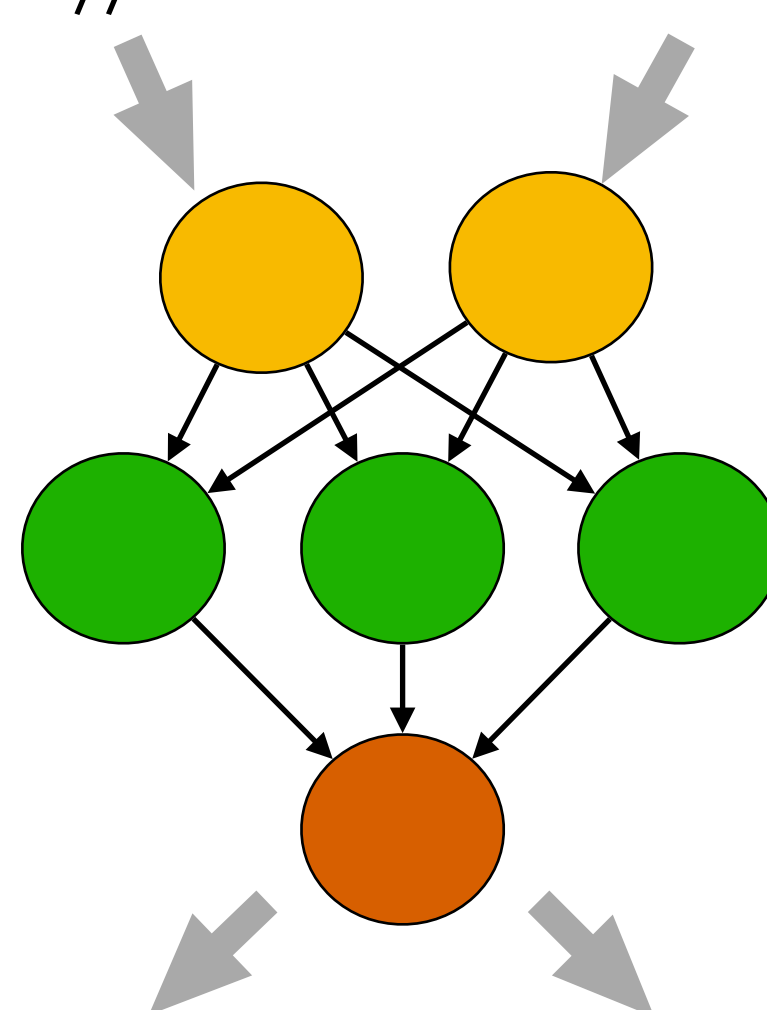


illegitimate objects



✓ return true

✗ return false



# Using Neural Networks to capture Data Structure Invariants

## Data Structure Implementation

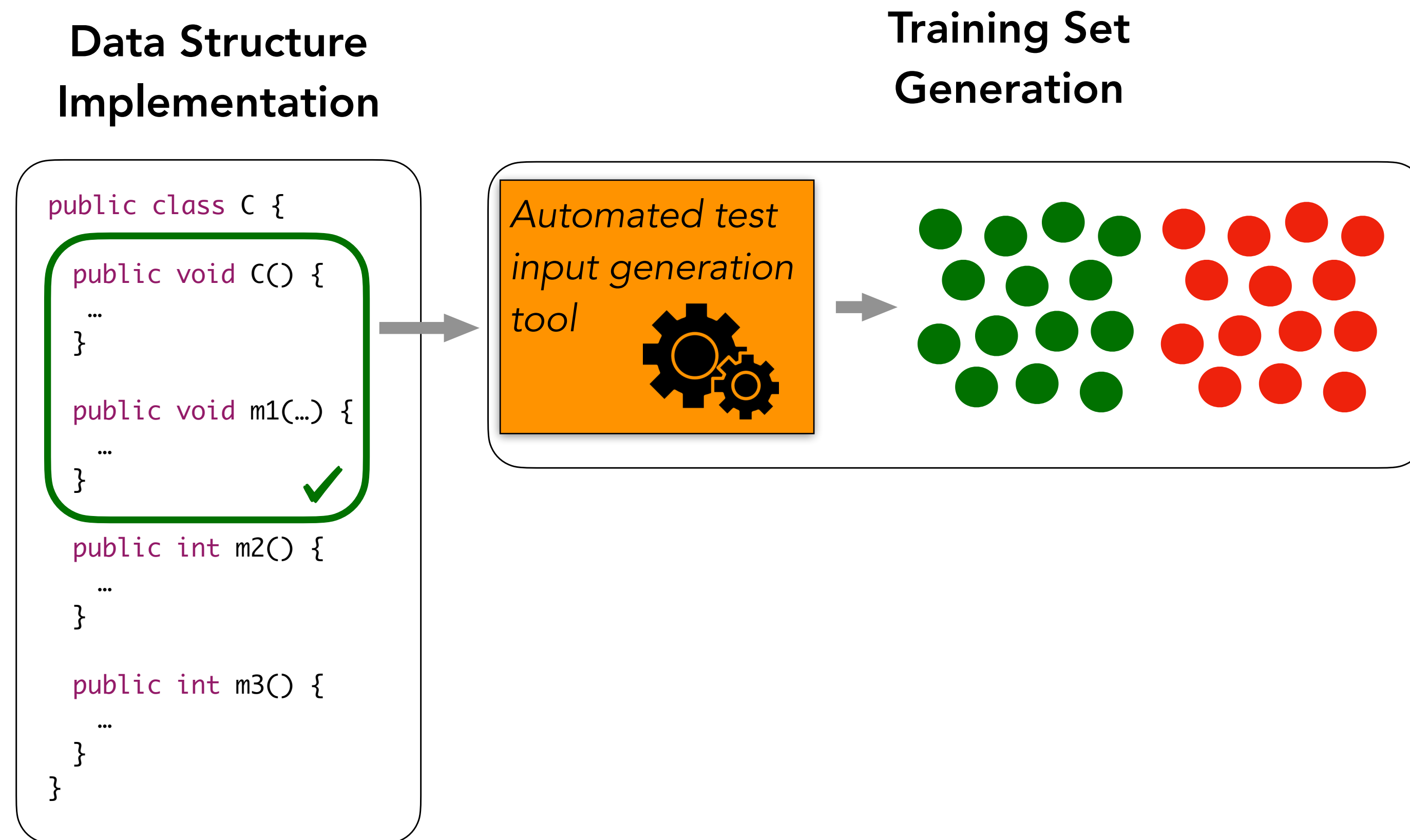
```
public class C {  
    public void C() {  
        ...  
    }  
  
    public void m1(...) {  
        ...  
    }  
  
    public int m2() {  
        ...  
    }  
  
    public int m3() {  
        ...  
    }  
}
```

# Using Neural Networks to capture Data Structure Invariants

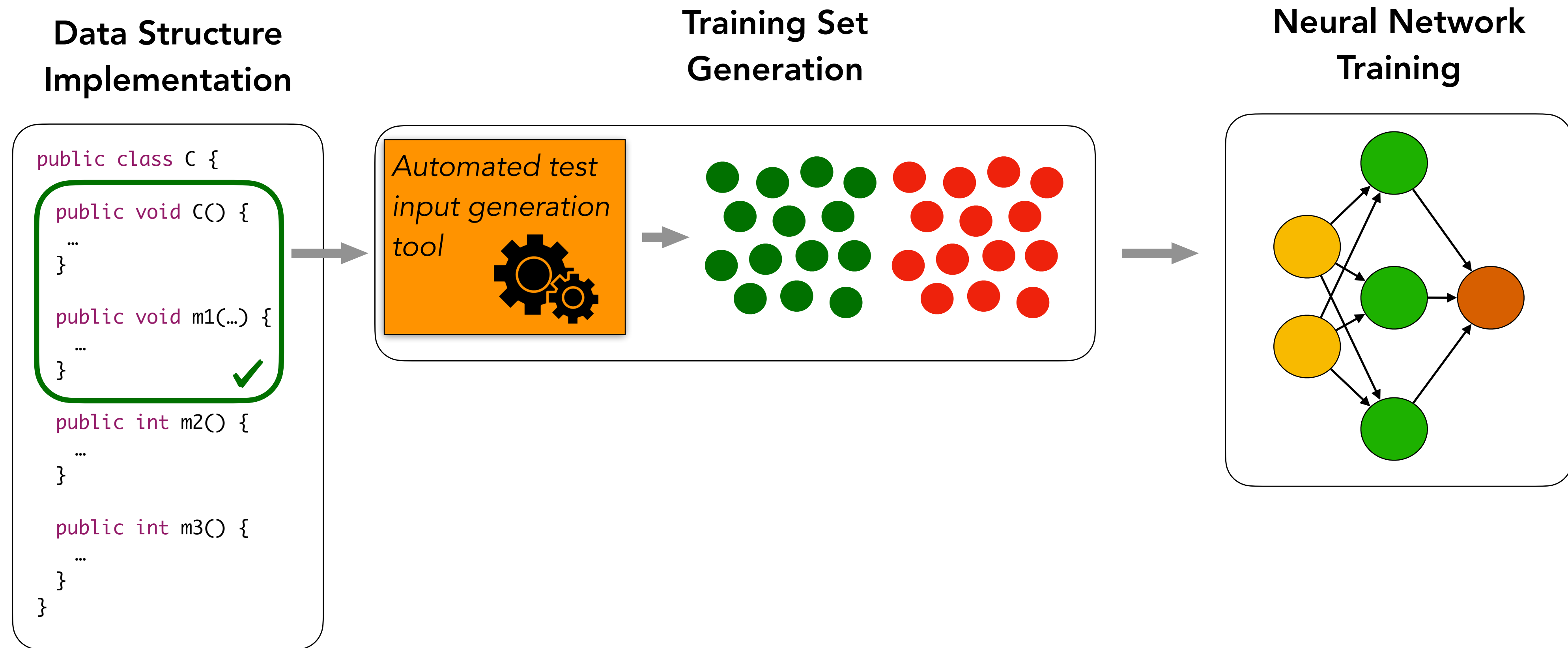
## Data Structure Implementation

```
public class C {  
    public void C() {  
        ...  
    }  
    public void m1(...) {  
        ...  
    }  
    public int m2() {  
        ...  
    }  
    public int m3() {  
        ...  
    }  
}
```

# Using Neural Networks to capture Data Structure Invariants



# Using Neural Networks to capture Data Structure Invariants



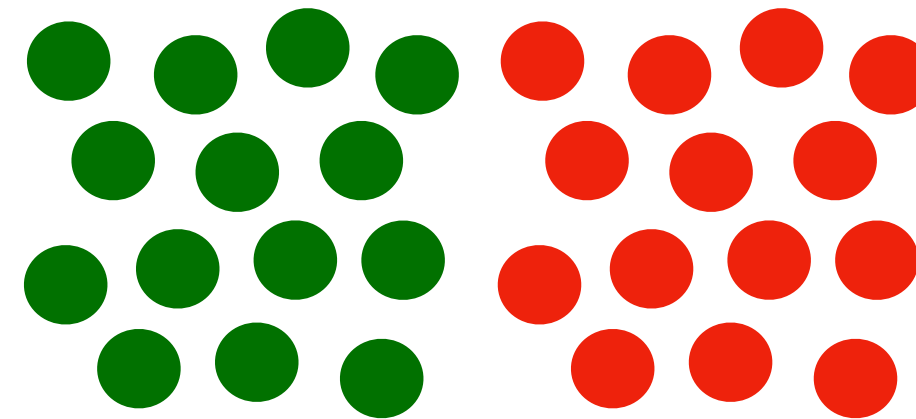
# Using Neural Networks to capture Data Structure Invariants

## Data Structure Implementation

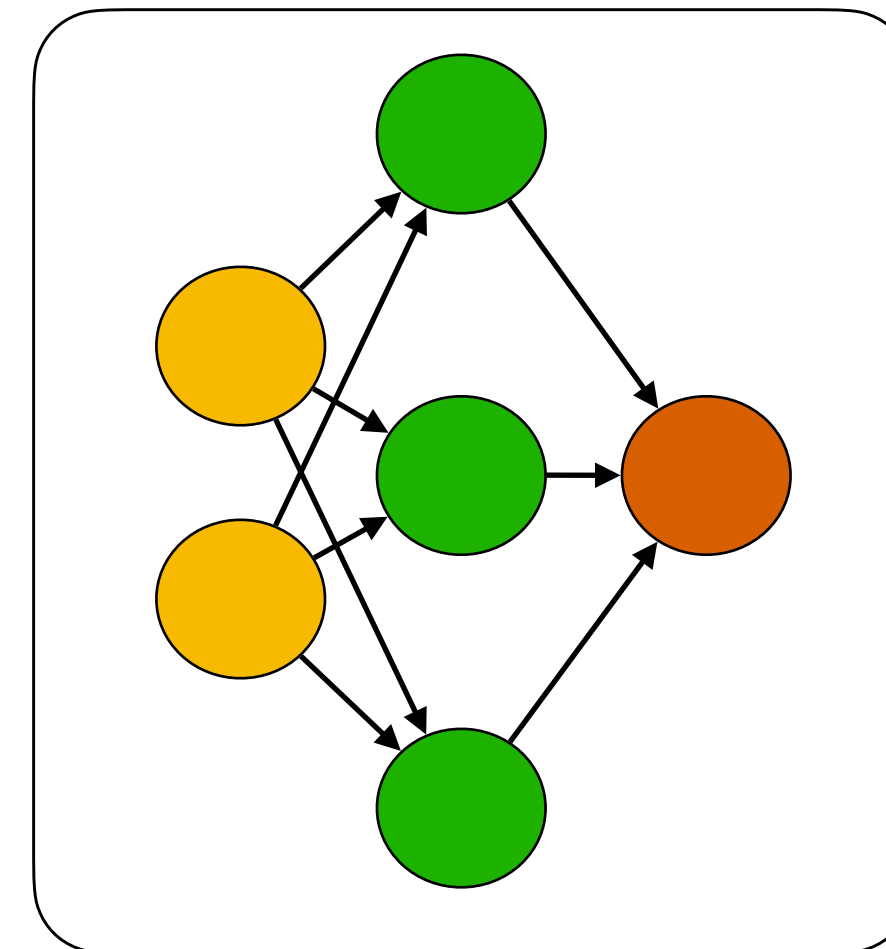
```
public class C {  
    public void C() {  
        ...  
    }  
    public void m1(...) {  
        ...  
    }  
    public int m2() {  
        ...  
    }  
    public int m3() {  
        ...  
    }  
}
```

Automated test input generation tool

## Training Set Generation



## Neural Network Training

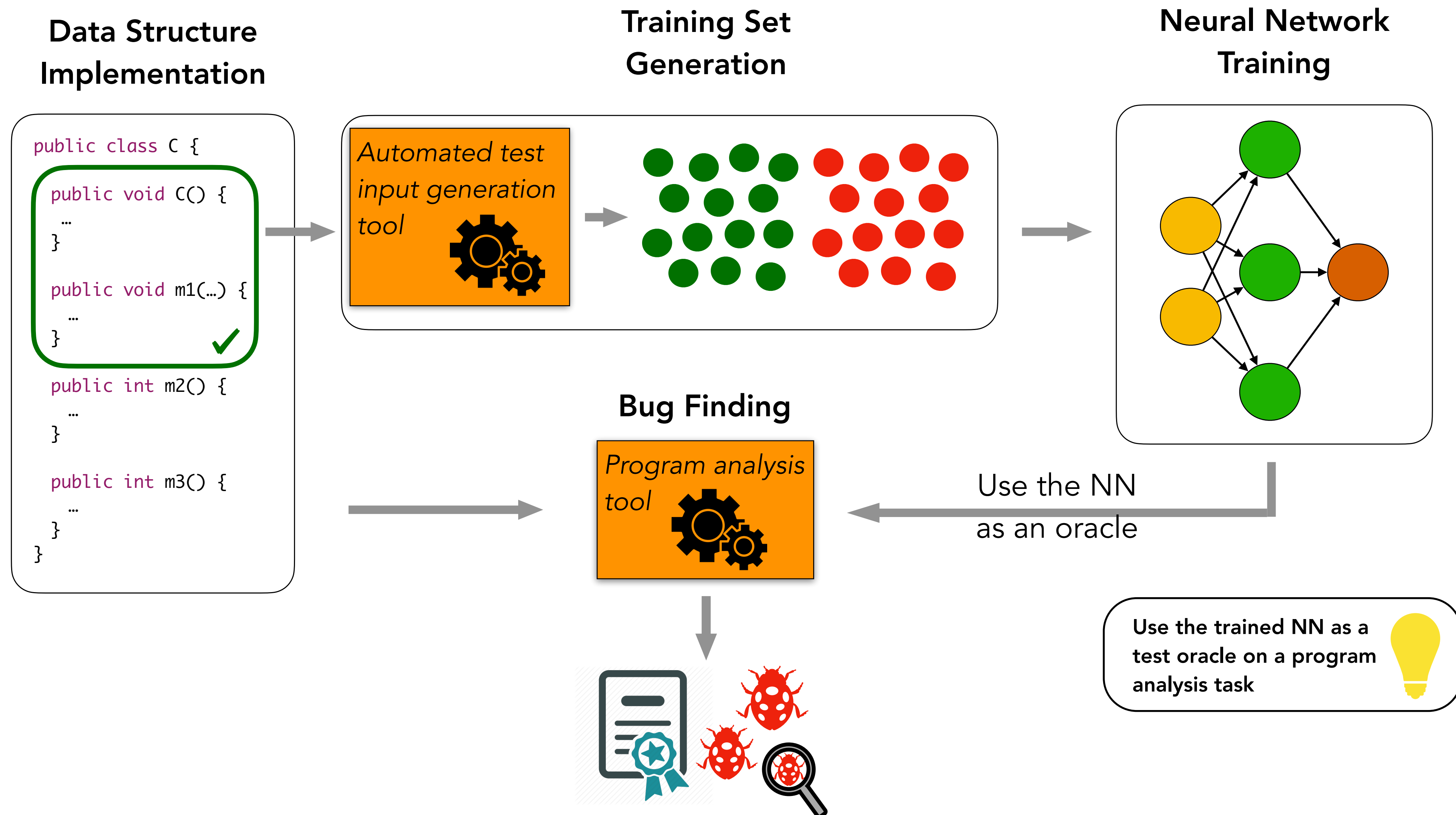


Use the trained NN as a test oracle on a program analysis task





# Using Neural Networks to capture Data Structure Invariants



# Using Neural Networks to capture Data Structure Invariants

## Remarks\*

+ Neural networks can closely approximate data structure invariants, with very high precision and recall (94.2%~99.9%). Also confirmed by \*\*

\* Facundo Molina, Renzo Degiovanni, Pablo Ponzio, Germán Regis, Nazareno Aguirre, and Marcelo Frias. **Training Binary Classifiers as Data Structure Invariants**. ICSE 2019.

\*\* Muhammad Usman, Wenxi Wang, Kaiyuan Wang, Cagdas Yelen, Nima Dini, and Sarfraz Khurshid. **A Study of Learning Data Structure Invariants Using Off-the-shelf Tools**. SPIN 2019.

# Using Neural Networks to capture Data Structure Invariants

## Remarks\*

- + Neural networks can closely approximate data structure invariants, with very high precision and recall (94.2%~99.9%). Also confirmed by \*\*
- + Bug detection can be improved when incorporating the trained classifiers as test oracles.

\* Facundo Molina, Renzo Degiovanni, Pablo Ponzio, Germán Regis, Nazareno Aguirre, and Marcelo Frias. **Training Binary Classifiers as Data Structure Invariants**. ICSE 2019.

\*\* Muhammad Usman, Wenxi Wang, Kaiyuan Wang, Cagdas Yelen, Nima Dini, and Sarfraz Khurshid. **A Study of Learning Data Structure Invariants Using Off-the-shelf Tools**. SPIN 2019.

# Using Neural Networks to capture Data Structure Invariants

## Remarks\*

- + Neural networks can closely approximate data structure invariants, with very high precision and recall (94.2%~99.9%). Also confirmed by \*\*
- + Bug detection can be improved when incorporating the trained classifiers as test oracles.
- Neural networks capturing data structure invariants are not directly interpretable.

\* Facundo Molina, Renzo Degiovanni, Pablo Ponzio, Germán Regis, Nazareno Aguirre, and Marcelo Frias. **Training Binary Classifiers as Data Structure Invariants**. ICSE 2019.

\*\* Muhammad Usman, Wenxi Wang, Kaiyuan Wang, Cagdas Yelen, Nima Dini, and Sarfraz Khurshid. **A Study of Learning Data Structure Invariants Using Off-the-shelf Tools**. SPIN 2019.

# Using Genetic Algorithms for Learning Postconditions

What is a postcondition?

```
public class C {  
    public void C() {  
        ...  
    }  
  
    public void m1(...) {  
        ...  
    }  
  
    public int m2() {  
        ...  
    }  
}
```

# Using Genetic Algorithms for Learning Postconditions

What is a postcondition?

```
public class C {  
    public void c() {  
        ...  
    }  
  
    public void m1(...) {  
        ...  
    }  
  
    public int m2() {  
        ...  
    }  
}
```

[Hoare 1969]: To state the required connection between a precondition ( $P$ ), a program ( $Q$ ) and a description of the result of its execution ( $R$ ), we introduce a new notation:

**$P\{Q\}R$**

This may be interpreted "If the assertion  $P$  is true before initiation of program  $Q$ , then **the assertion  $R$  will be true on its completion**"

# Using Genetic Algorithms for Learning Postconditions

```
public class BST {  
    public void BST() {  
    }  
    public void insert(int i)  
    {  
        ...  
    }  
    public void remove(int i) {  
        ...  
    }  
    ...  
}
```

# Using Genetic Algorithms for Learning Postconditions

```
public class BST {  
    public void BST() {  
    }  
    public void insert(int i)  
    {  
        ...  
    }  
    public void remove(int i) {  
        ...  
    }  
    ...  
}
```

A good **postcondition assertion** for method *insert* should check that:

- The resulting BST is a valid BST (i.e., the invariant is preserved)
- The element is effectively inserted into the BST
- All previous elements in the BST are still in the BST



# Using Genetic Algorithms for Learning Postconditions

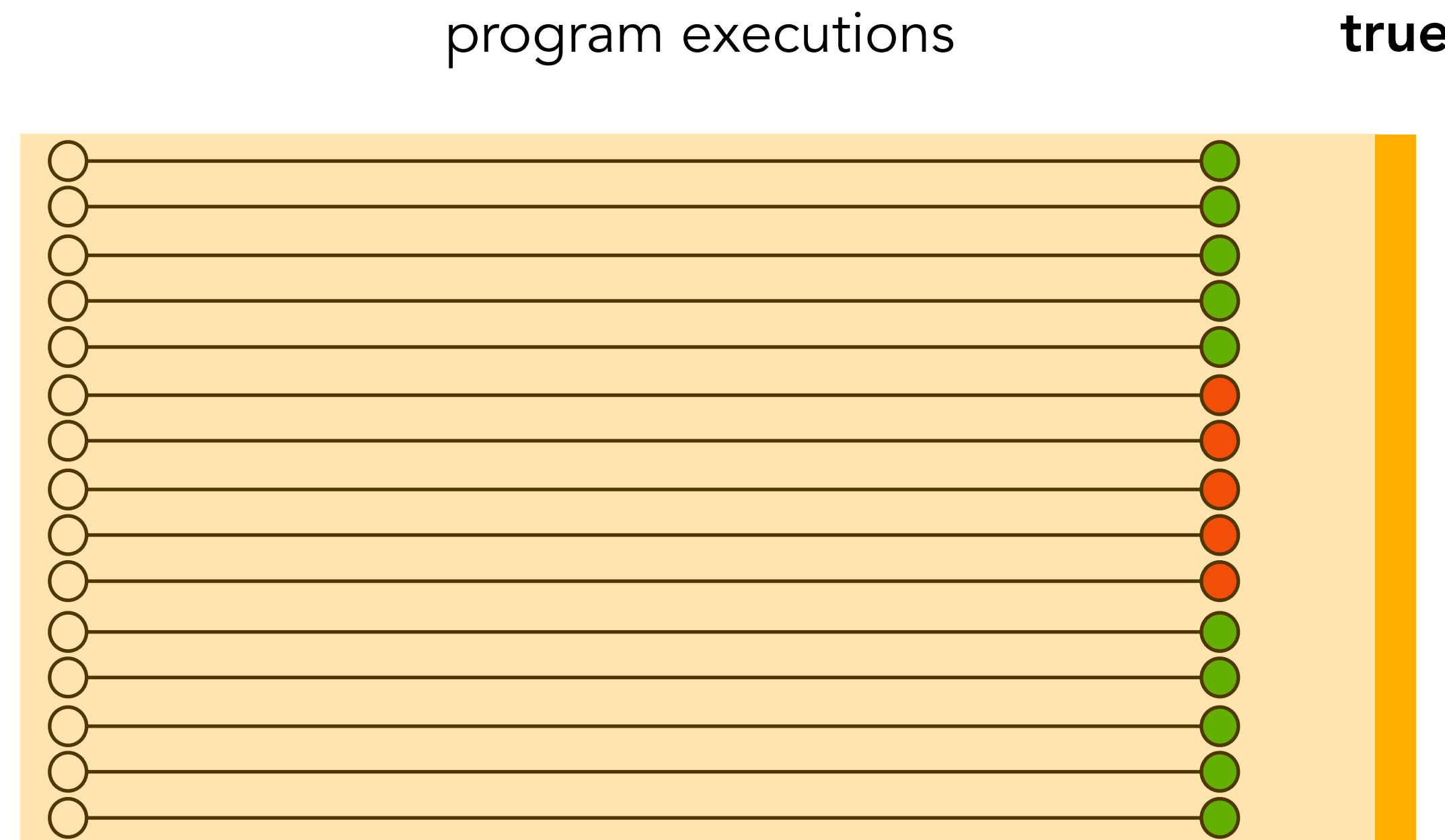
```
public class BST {  
    public void BST() {  
    }  
    public void insert(int i)  
    {  
        ...  
    }  
    public void remove(int i) {  
        ...  
    }  
    ...  
}
```

program executions



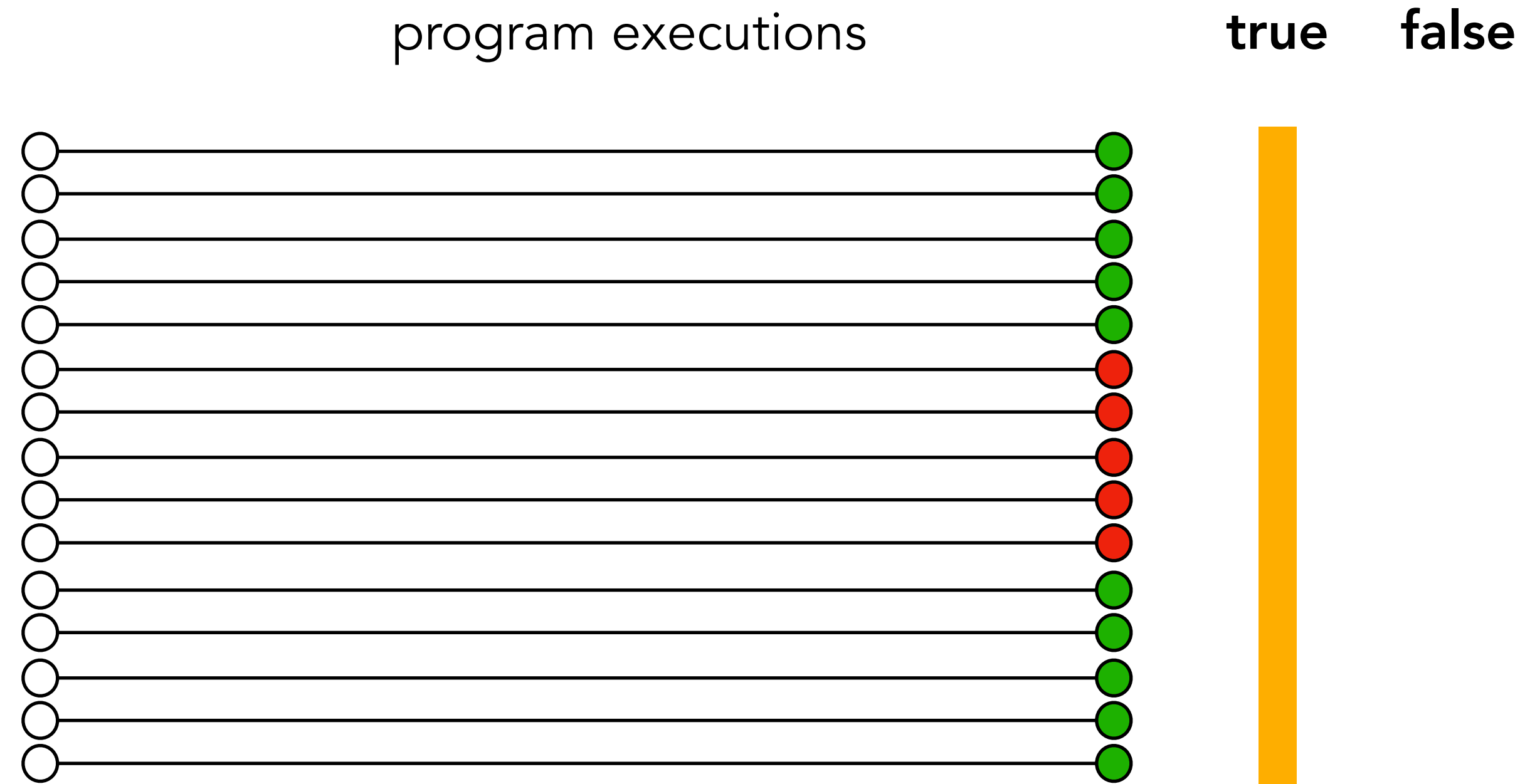
# Using Genetic Algorithms for Learning Postconditions

```
public class BST {  
    public void BST() {  
    }  
    public void insert(int i)  
    {  
        ...  
    }  
    public void remove(int i) {  
        ...  
    }  
    ...  
}
```



# Using Genetic Algorithms for Learning Postconditions

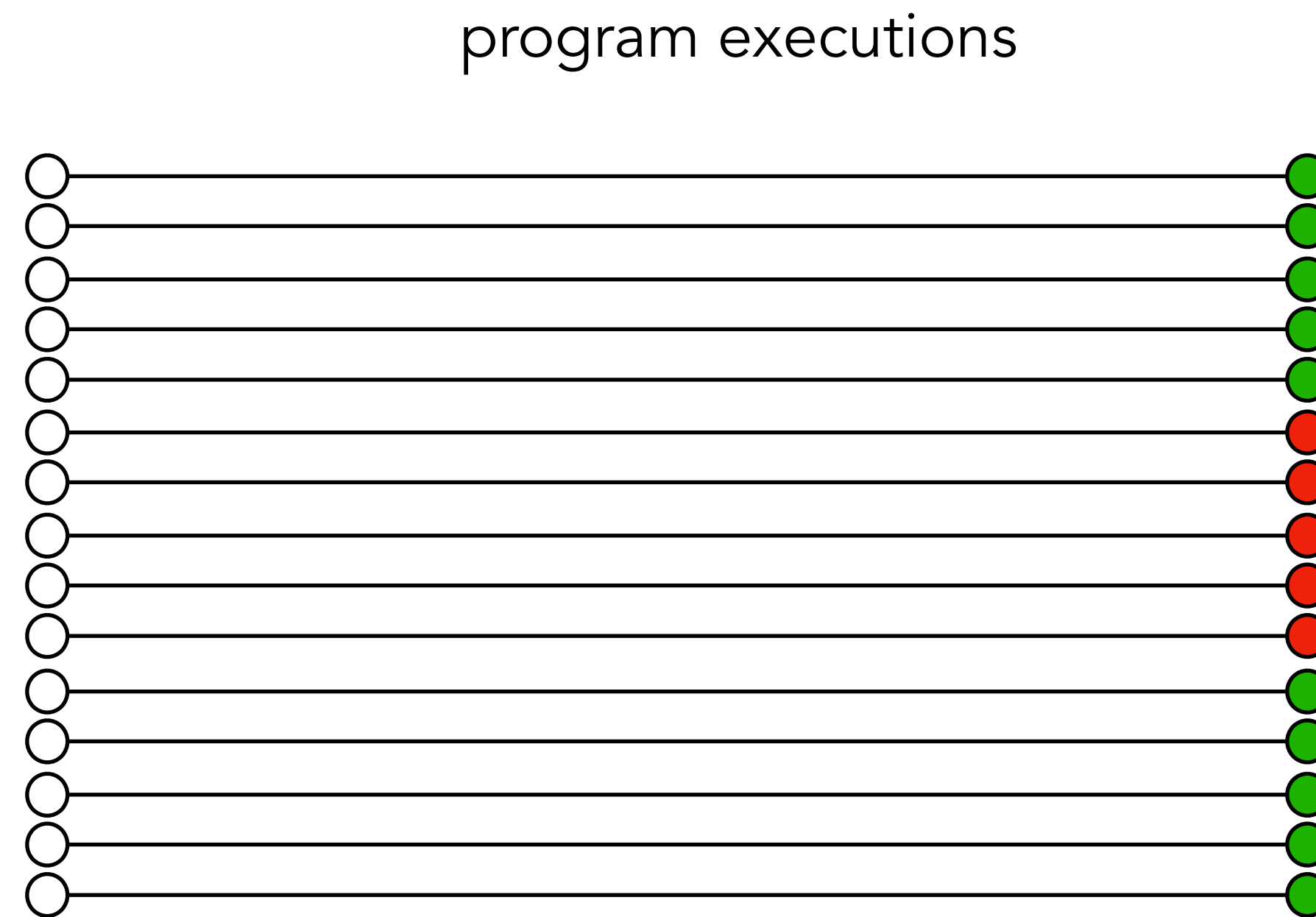
```
public class BST {  
    public void BST() {  
    }  
    public void insert(int i)  
    {  
        ...  
    }  
    public void remove(int i) {  
        ...  
    }  
    ...  
}
```



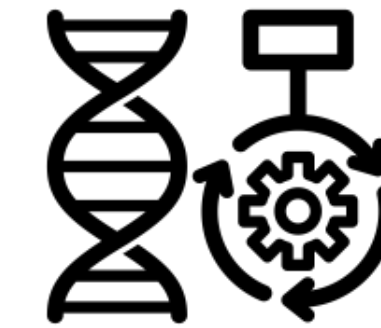
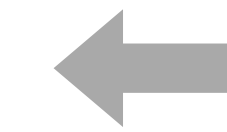


# Using Genetic Algorithms for Learning Postconditions

```
public class BST {  
    public void BST() {  
    }  
    public void insert(int i)  
    {  
        ...  
    }  
    public void remove(int i) {  
        ...  
    }  
    ...  
}
```

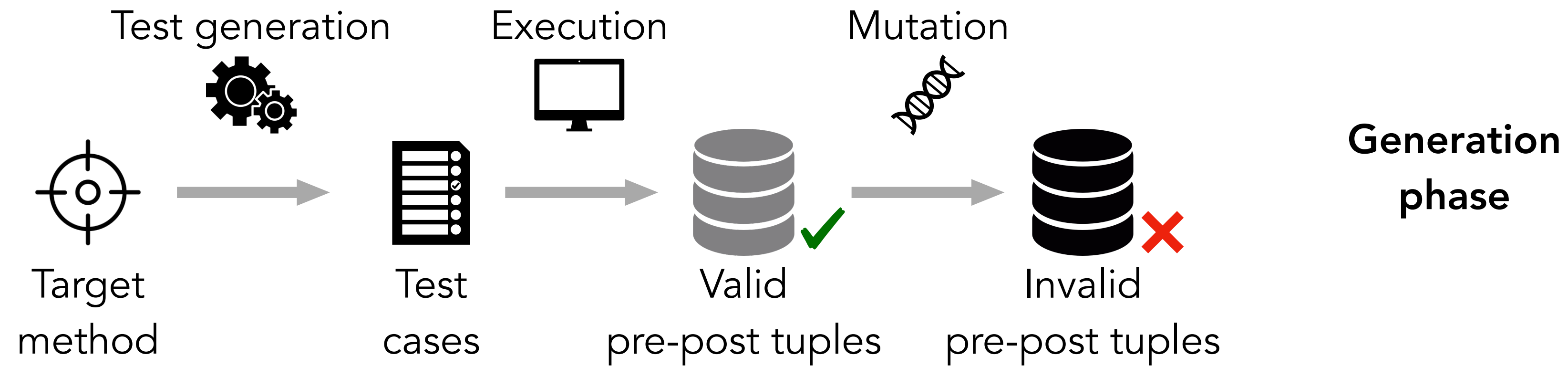


$\emptyset$

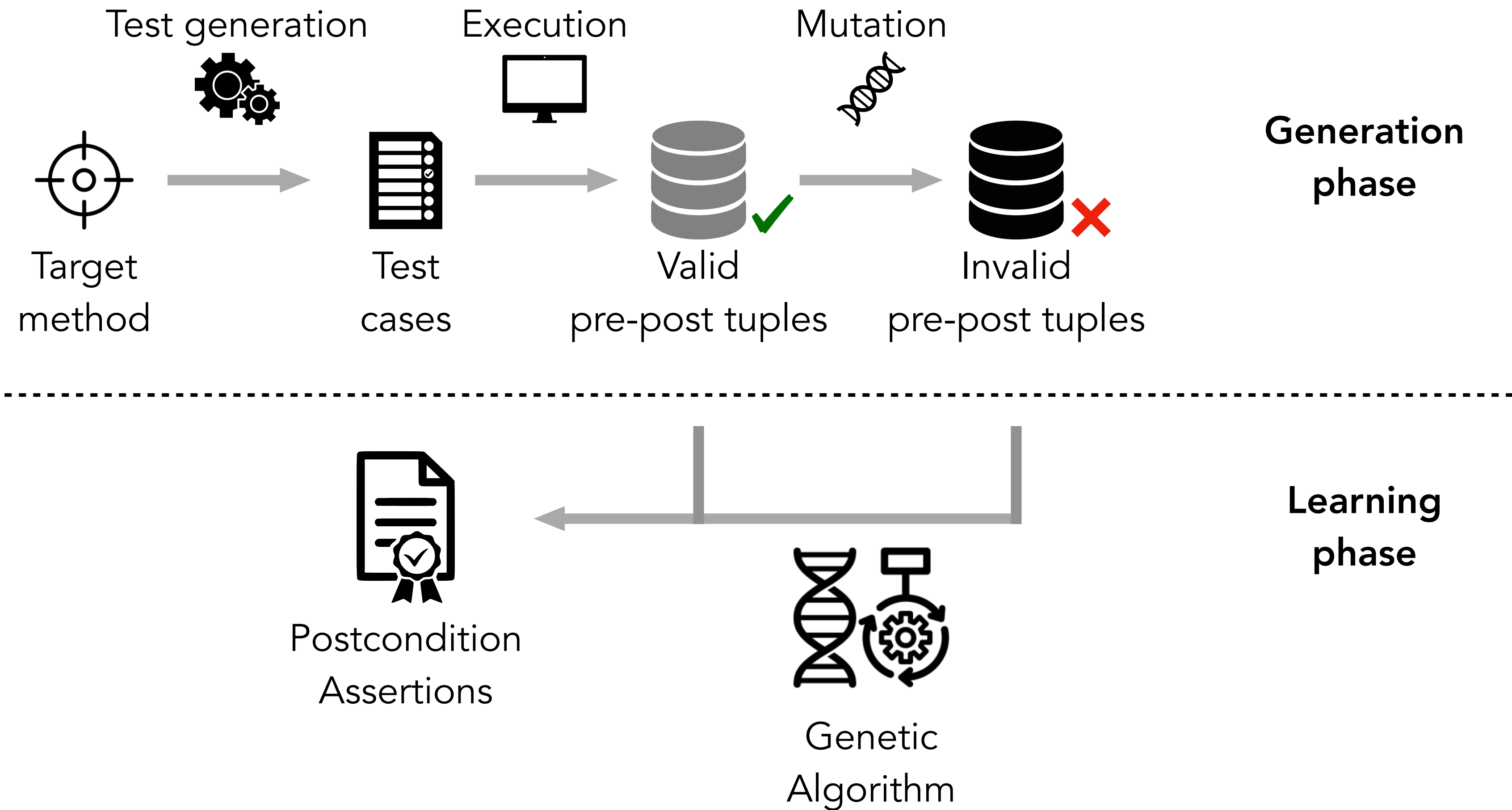


Genetic  
Algorithm

# Using Genetic Algorithms for Learning Postconditions



# Using Genetic Algorithms for Learning Postconditions



# Using Genetic Algorithms for Learning Postconditions

## Remarks

- + The technique is able to capture sophisticated properties of reference based implementations.



# Using Genetic Algorithms for Learning Postconditions

## Remarks

- + The technique is able to capture sophisticated properties of reference based implementations.
- + Postconditions are directly interpretable (captured as standard assertions).

# Using Genetic Algorithms for Learning Postconditions

## Remarks

- + The technique is able to capture sophisticated properties of reference based implementations.
- + Postconditions are directly interpretable (captured as standard assertions).
- The generation phase is quite time consuming (test generation + mutation)

# Conclusions

- ◆ The Oracle Problem is a relevant problem in Software Engineering.
  - Oracles are seldom found in practice, but they can significantly improve bug finding.

# Conclusions

- ◆ The Oracle Problem is a relevant problem in Software Engineering.
  - Oracles are seldom found in practice, but they can significantly improve bug finding.
- ◆ Derived oracles are specifications that can be obtained from existing software elements.

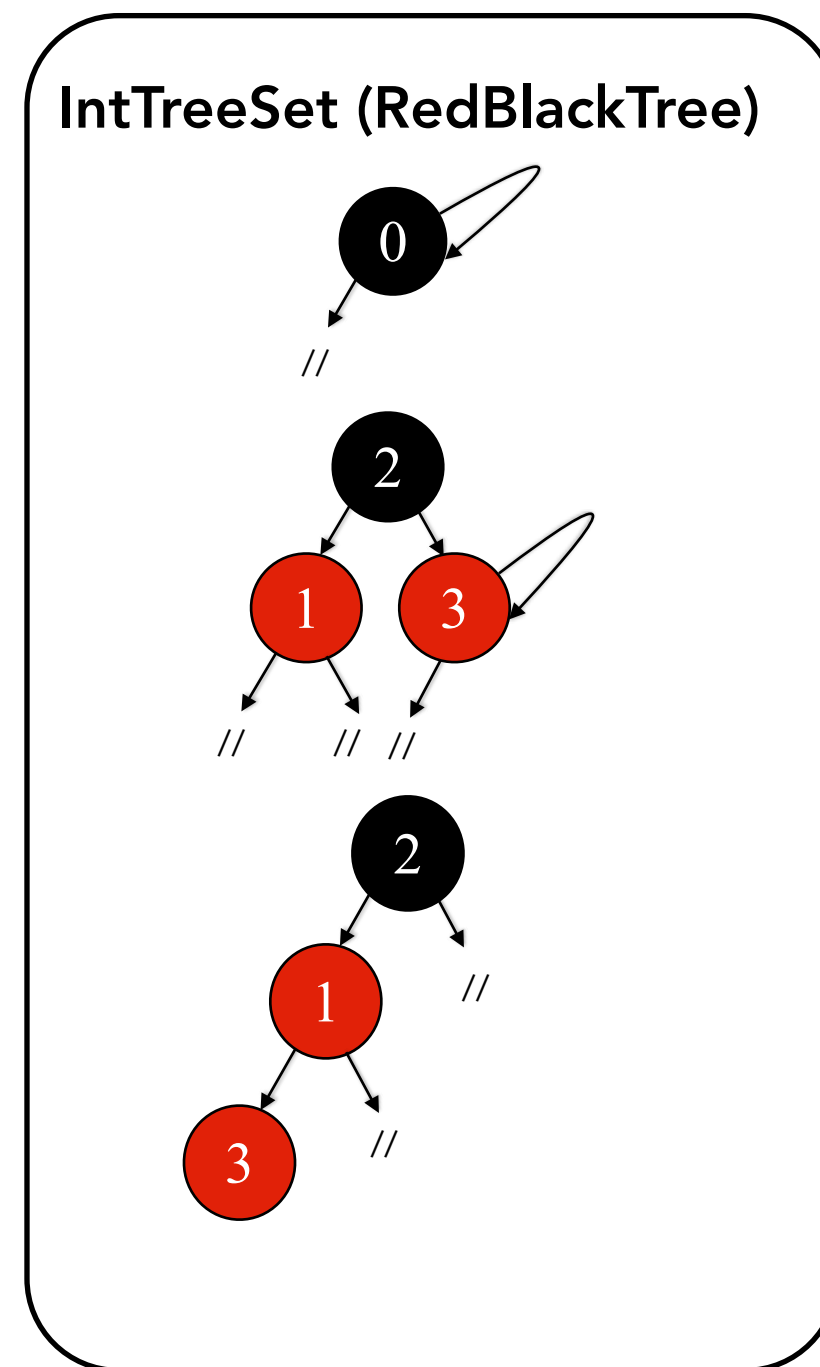
# Conclusions

- ◆ The Oracle Problem is a relevant problem in Software Engineering.
  - Oracles are seldom found in practice, but they can significantly improve bug finding.
- ◆ Derived oracles are specifications that can be obtained from existing software elements.
- ◆ Learning techniques can effectively derive oracles from program behaviors
  - Neural networks as data structure invariants
  - Genetic algorithms for learning postconditions

**Thank you :)**

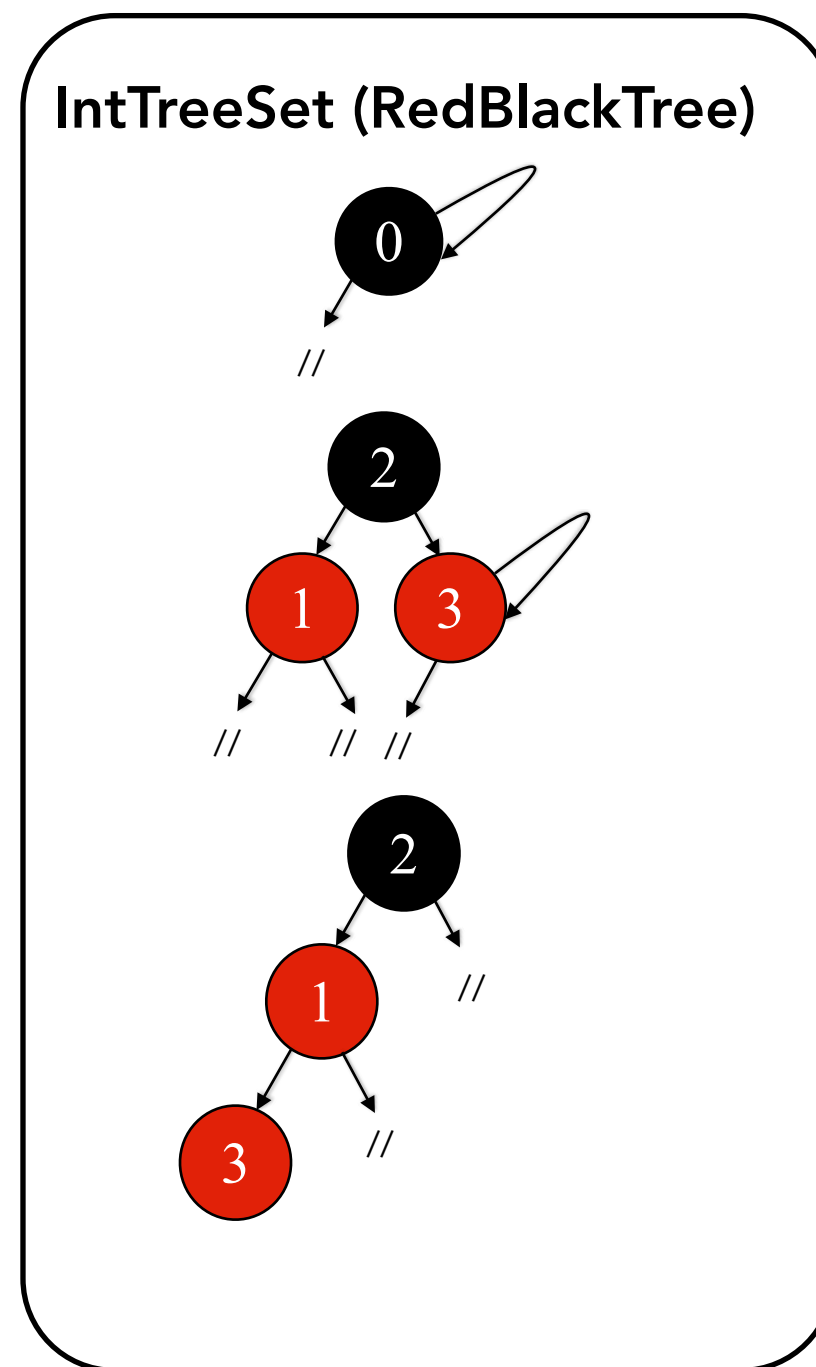
**Questions?**

# Bug Detection with Neural Networks as Test Oracles

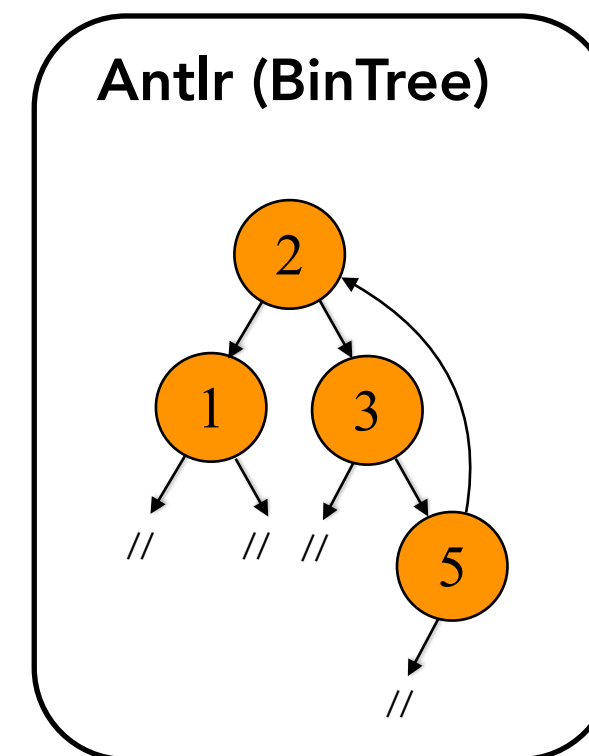


- Acyclicity
- Balancedness

# Bug Detection with Neural Networks as Test Oracles



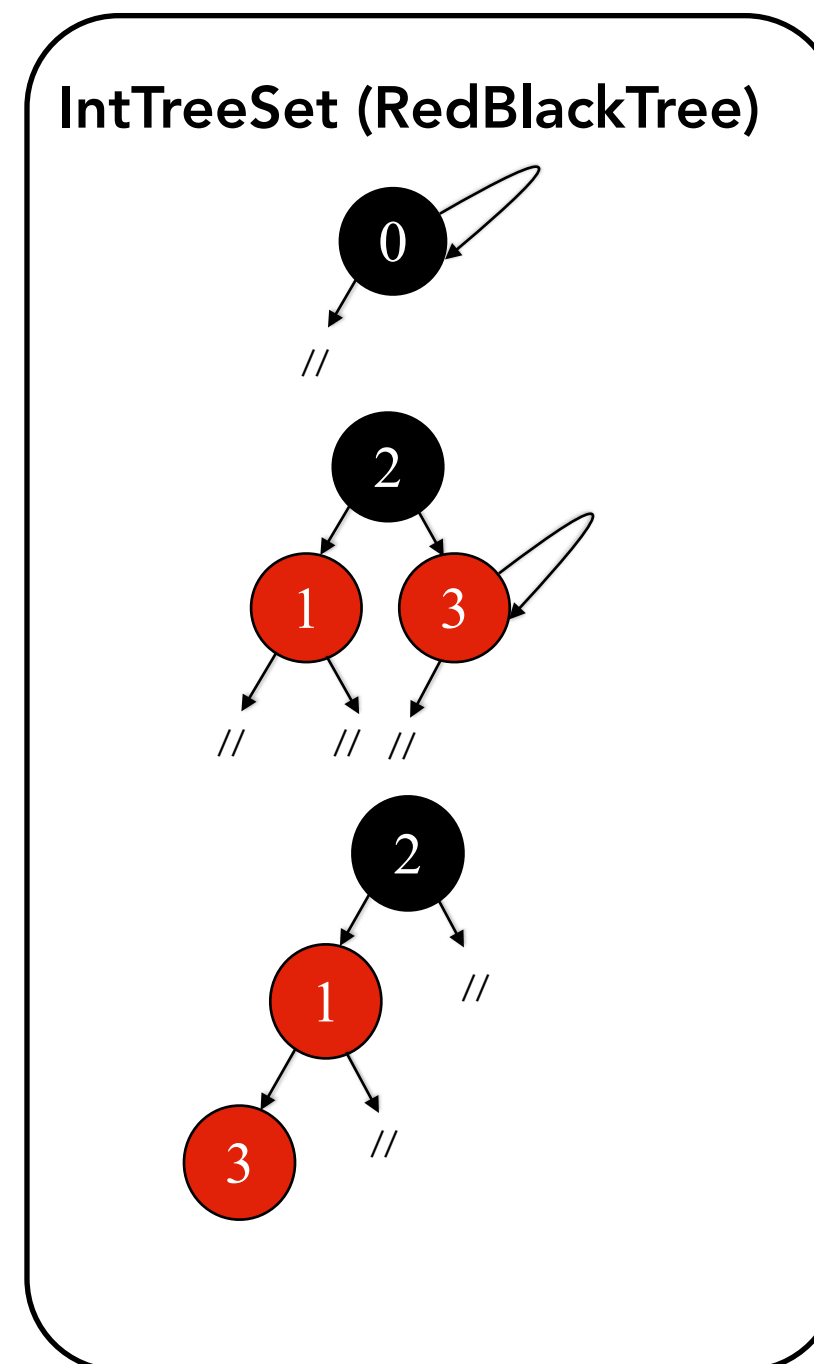
- Acyclicity
- Balancedness



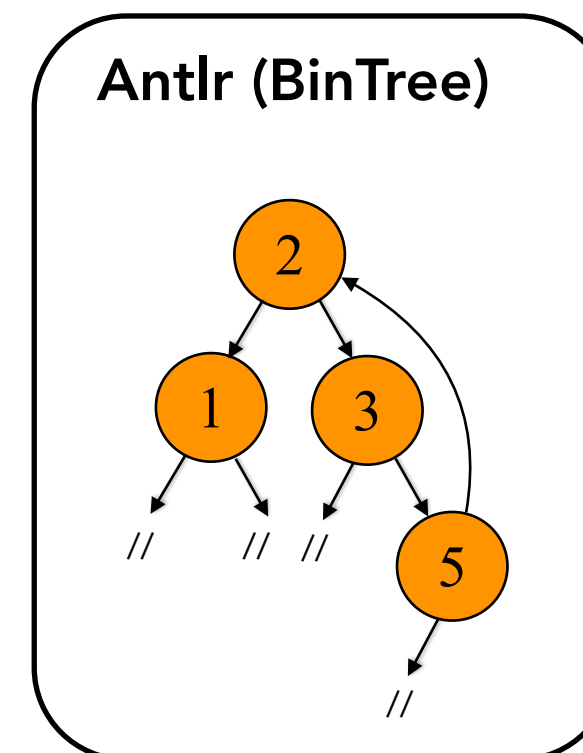
- Acyclicity



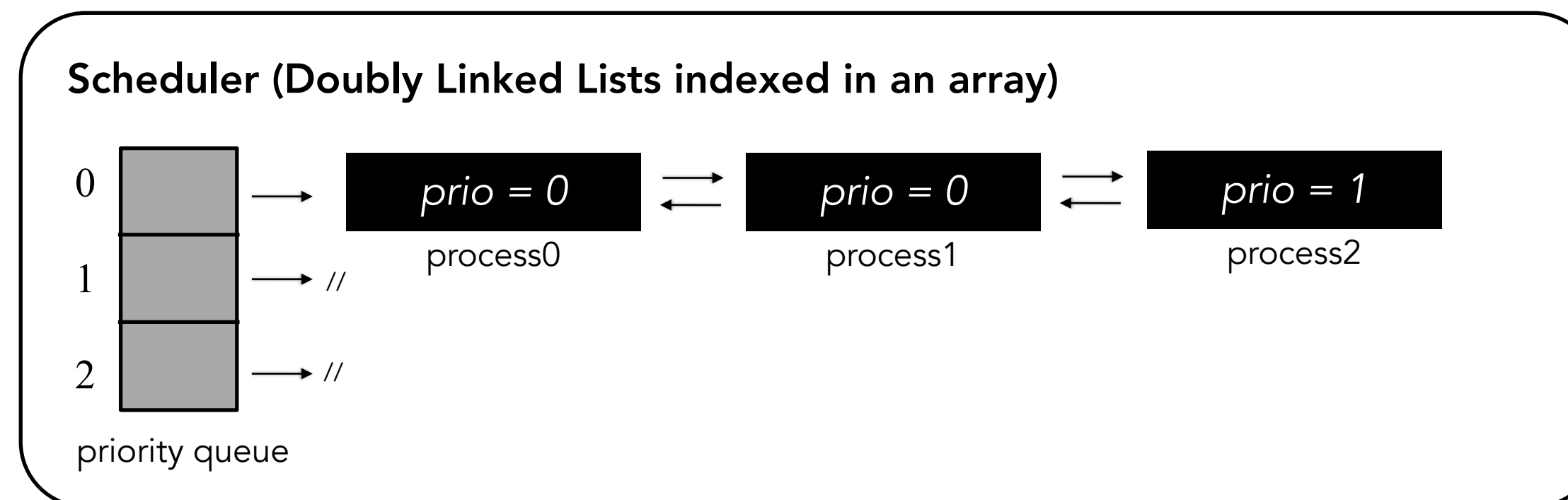
# Bug Detection with Neural Networks as Test Oracles



- Acyclicity
- Balancedness



- Acyclicity

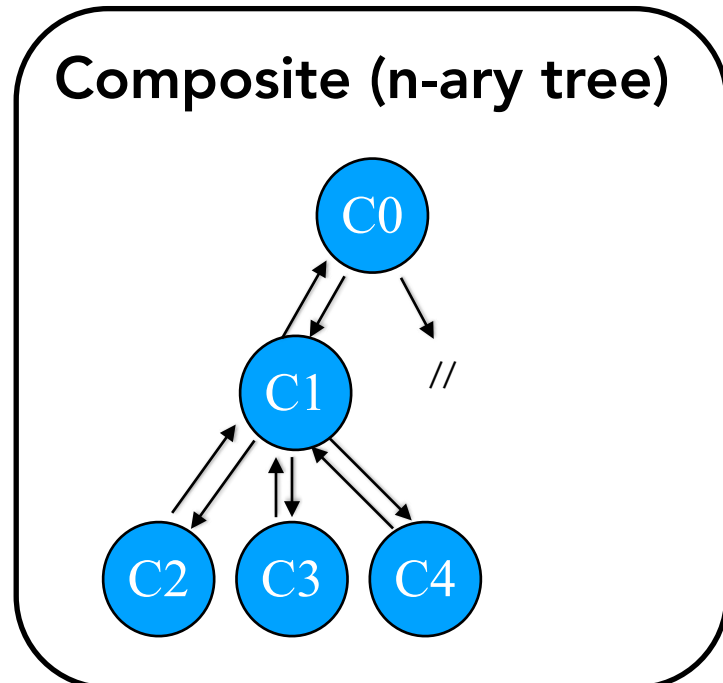


- Indexing

# Inferring manually written postconditions

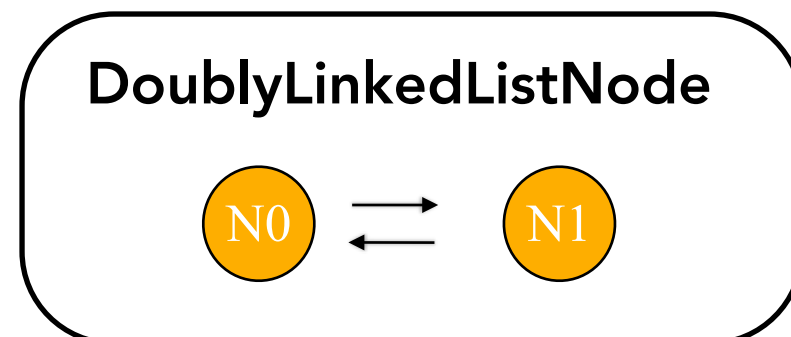
## Method

## Postcondition



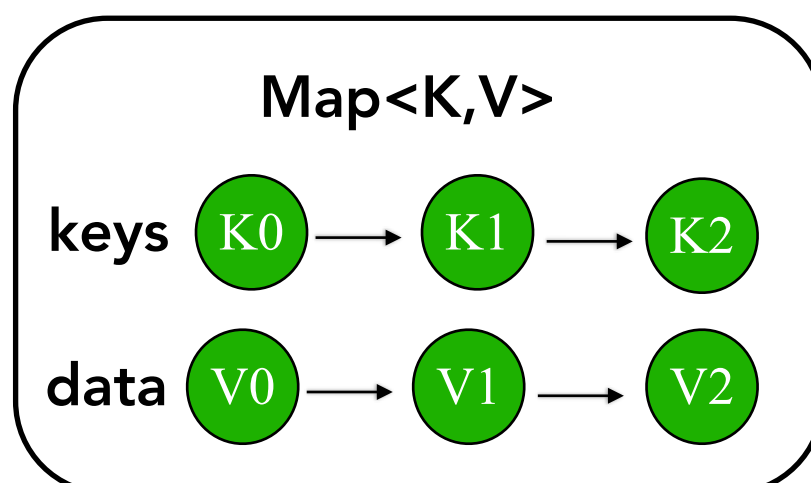
*add\_child(Composite c) : void*

*child added  
c value unchanged  
c children unchanged  
ancestors unchanged*



*remove(): void*

*singleton  
neighbors connected*



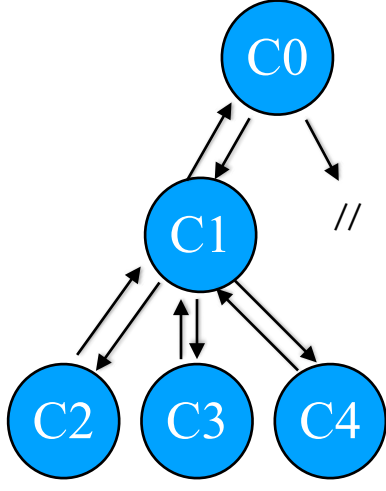
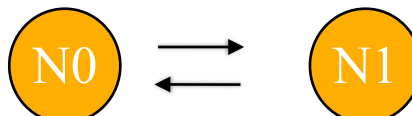
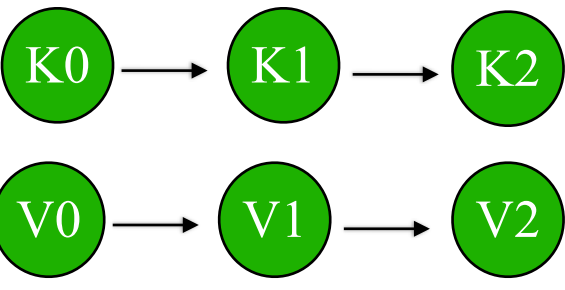
*count(): int*

*result is key set size*

*extend(K k , V v): int*

*key set  
data set  
other keys unchanged  
other data unchanged  
result is index*

# Inferring manually written postconditions

	Method	Postcondition	Inferred?
<p>Composite (n-ary tree)</p> 	<code>add_child(Composite c) : void</code>	<code>child added</code> <code>c value unchanged</code> <code>c children unchanged</code> <code>ancestors unchanged</code>	<p>✓</p> <p>✓</p>
<p>DoublyLinkedListNode</p> 	<code>remove(): void</code>	<code>singleton</code> <code>neighbors connected</code>	<p>✓</p>
<p>Map&lt;K,V&gt;</p> 	<code>count(): int</code> <code>extend(K k , V v): int</code>	<code>result is key set size</code> <code>key set</code> <code>data set</code> <code>other keys unchanged</code> <code>other data unchanged</code> <code>result is index</code>	<p>✓</p> <p>✓</p> <p>✓</p>