Generación Automática de Oráculos para Tests

Dr. Facundo Molina
IMDEA Software Institute
Madrid, España

Jornadas de Ciencias de la Computación - Rosario, Octubre 2023

Introducción al Problema del Oráculo

Software Testing - Conceptos básicos

Los *tests* son secuencias de estímulos y observaciones.

Inputs? ***enteros***

Rango? ***enteros positivos***

Abordando el Problema del Oráculo

Oráculos Implícitos

Chequear propiedades esperadas de muchos programas

Oráculos Especificados

Los desarrolladores especifican formalmente propiedades que el comportamiento correcto debería cumplir

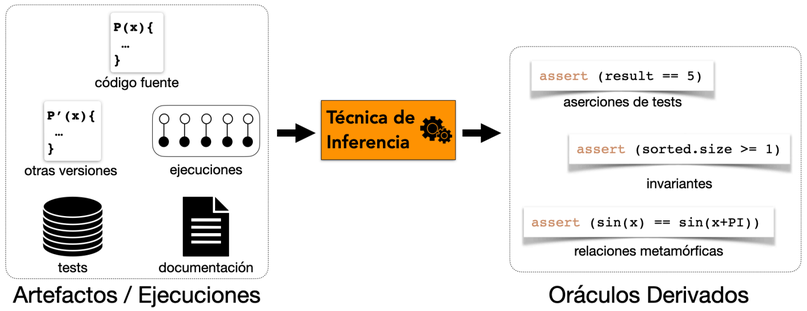
Oráculos Derivados

Derivados a partir de artefactos del software o ejecuciones del sistema

Oráculos Humanos

Cómo manejamos la falta de oráculos?

Generación Automática de Oráculos



Conclusiones Finales

	Costo	Precisión	Completitud
Oráculos Implícitos	🟢	🟢	🔴
Especificación Manual	🔴	🟢	🔴
Aserciones de Tests	🟢	🟢	🟡
Detección de Invariantes	🟡🟢	🟡	🟡
Testing Metamórfico	🟢	🟢	🟡

Software Testing - Conceptos básicos

Los *tests* son secuencias de estímulos y observaciones.

1
5
1000
1000999999
0
-10

Es suficiente?

Inputs? ***enteros***

Rango? ***enteros positivos***

Software Testing - Conceptos básicos

Los *tests* son secuencias de estímulos y observaciones.

1
5
1000
1000999999
0
-10

Función

La implementación es correcta?

oráculo

✓ ✗

Inputs? ***enteros***

Rango? ***enteros positivos***

De qué estamos hablando?

```java
import org.junit.Test;
import static org.junit.Assert.assertTrue;

public class TestAssertions {

  @Test
  public void testSubString() {
    String str1 = "abc";
    String str2 = "abc";
    boolean b = isSubString(str1, str2);
    assertTrue(b);
  }
```

test inputs

función bajo test

oráculo

Cómo definimos un oráculo de prueba?

```
int[] input = {1, 12, 4, 5, 32, 15};
int x = getMaxValue(input);
// ?
```

```
assertEquals(32, x)
```

```
int[] input = {1, 12, 4, 5, 32, 15};
int[] x = sort(input);
// ?
```

```
assertEquals(sortedArray,x)

assertEquals(x[0], 1)

assertEquals(x[5], 32)
```

Cómo definimos un oráculo de prueba?

Attitude and Orbit Control Systems (AOCS) are required for all space missions. The general scope of a satellites trajectory is set by the launcher that hauls it skyward – selected by orbital dynamics experts long in advance of the satellite being built – after which smaller thrusters manoeuvre it into its operational orbit. After that the onboard closed-loop control is in charge of controlling the spacecrafts pointing direction – known as its attitude – as it proceeds along its orbital path.

```
assert..(x está en un rango de valores correctos conocidos)
```

Science

Big Chinese rocket segment set to fall to Earth

By Jonathan Amos
BBC Science Correspondent

2 days ago

The rocket was launched to carry a Chinese space station section into orbit

Oráculo de Prueba - definición

Si un test es una **secuencia de actividades** (estímulos y observaciones), un oráculo es un predicado que determina si una secuencia dada es **aceptable o no**.

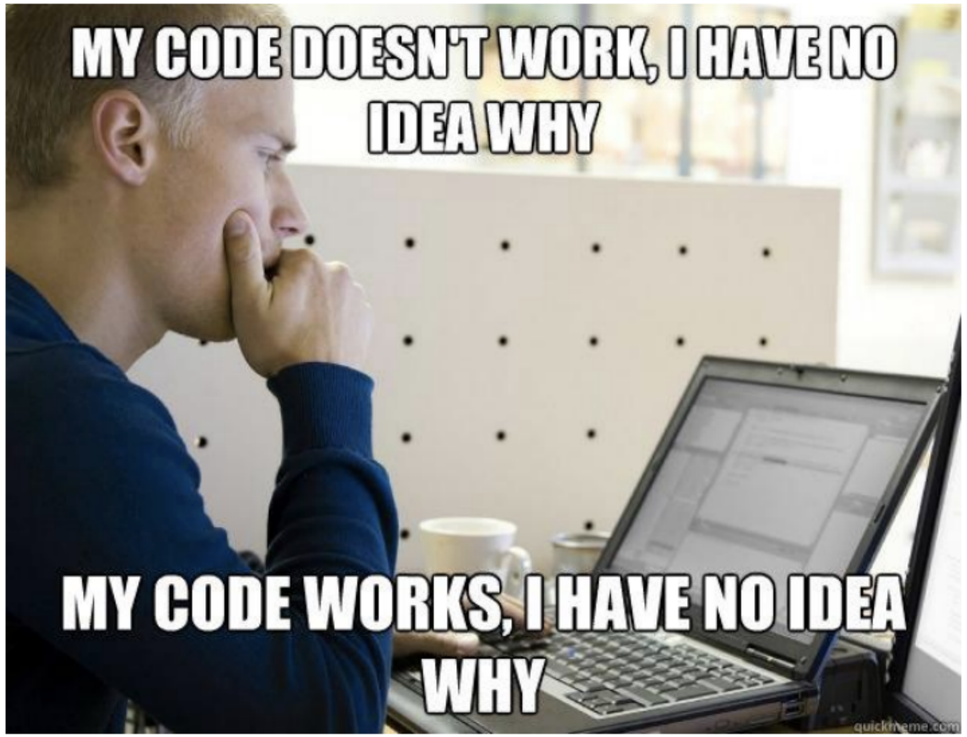
Los Oráculos son Programas

- Un oráculo es una implementación de una especificación.
- Los oráculos deben ser desarrollados.
  - Al igual que el proyecto, un oráculo se construye a partir de la especificación de requisitos.
    - … y está sujeto a la interpretación del desarrollador
    - … y podría contener fallas
- Un oráculo erróneo puede generar problemas.
  - Puede resultar en **falsos negativos** - “pasar” cuando en realidad hay una falla en el sistema.
  - Puede resultar en **falsos positivos** - “fallar” cuando en realidad no había ninguna falla en el sistema.

De dónde sacamos los oráculos?

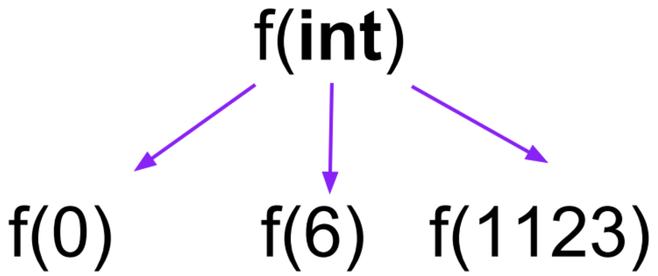
Comúnmente: los desarrolladores escriben oráculos asociados a tests particulares.

- Gran cantidad de esfuerzo manual y tiempo destinado a crear tests.



El problema del oráculo

Generar nuevos es inputs es “fácil”



Pero es mucho mas difícil chequear los resultados para múltiples tests.

f(0) = **?**

f(6) = **?**

f(1123) = **?**

Andreas Zeller et al. **The Oracle Problem in Software Testing.** 2023

Abordando el Problema del Oráculo

Oráculos Implícitos

Chequear propiedades esperadas de muchos programas

Oráculos Especificados

Los desarrolladores especifican formalmente propiedades que el comportamiento correcto debería cumplir

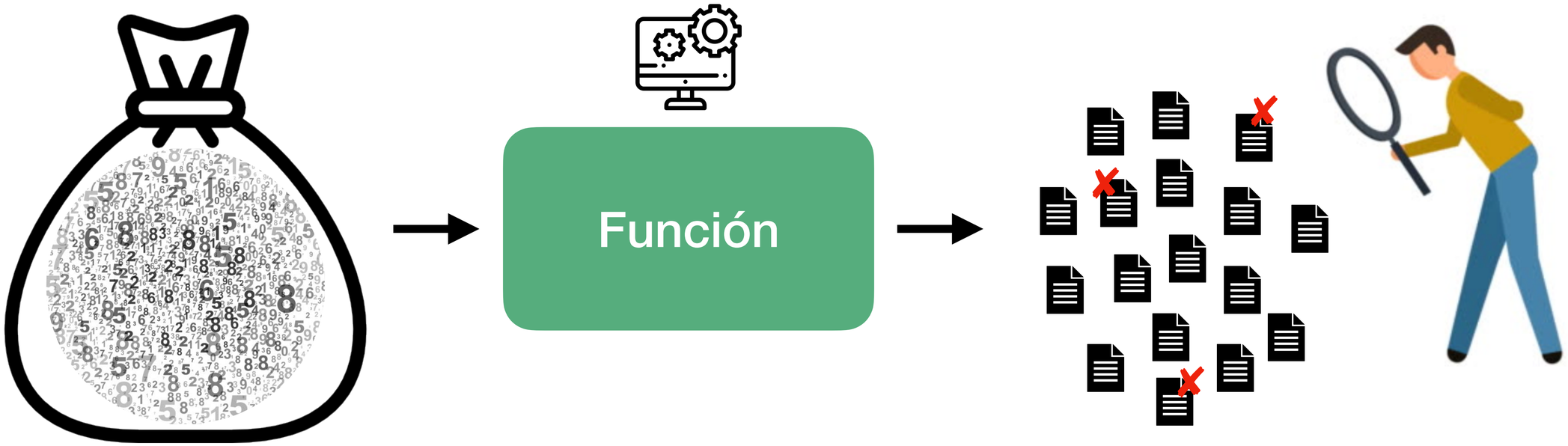
Oráculos Derivados

Derivados a partir de artefactos del software o ejecuciones del sistema

Oráculos Humanos

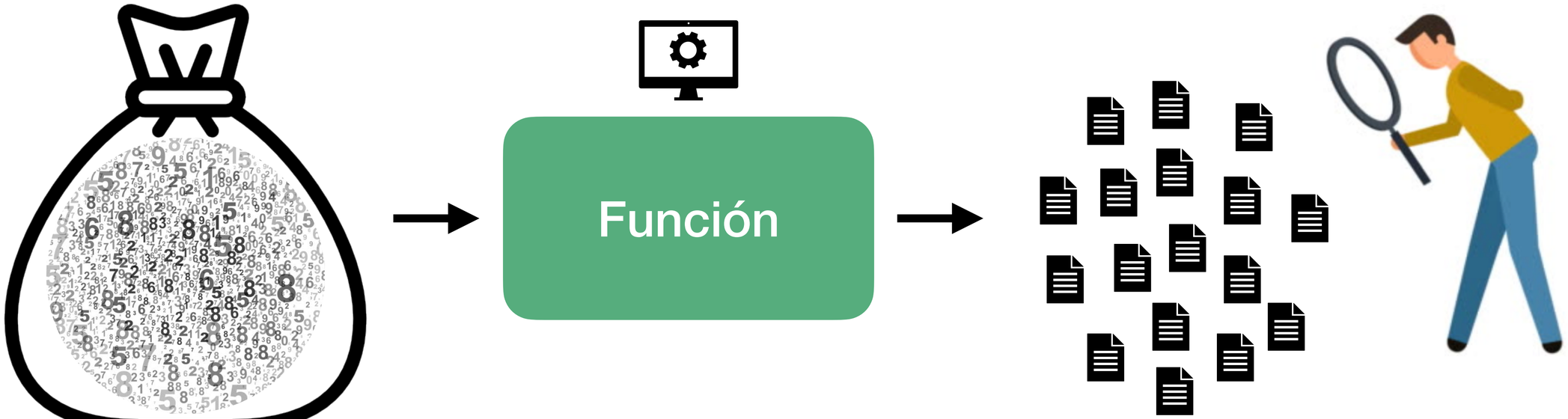
Cómo manejamos la falta de oráculos?

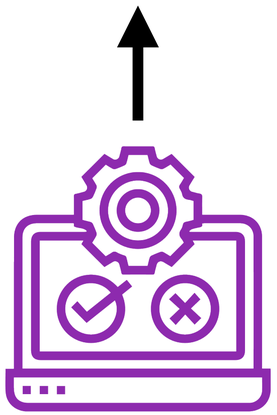
Oráculos Implícitos



- No requieren conocimiento del dominio o especificación alguna
- Permiten chequear propiedades esperadas de cualquier programa
- Puede detectar anomalías particulares, como irregularidades de red o deadlocks

Oráculos Implícitos





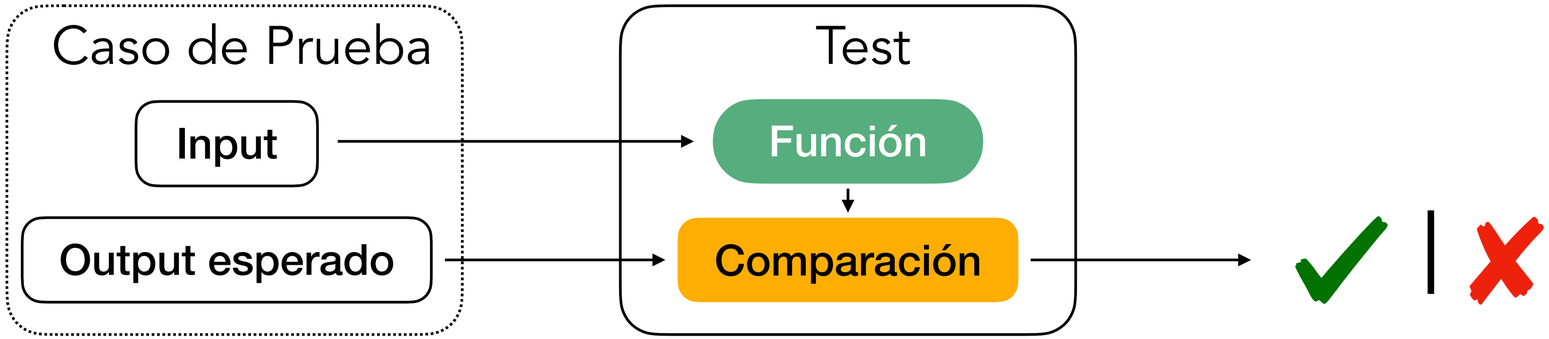
Fuzzing

- Generación de inputs random eficiente y efectiva.
- Atacar al sistema con estos inputs.
  - Proceso guiado por “perfiles de ataque” que reflejan ciertos escenarios maliciosos.
  - Detección de vulnerabilidades de seguridad.

Andreas Zeller et al. **The Fuzzing Book.** 2023

Oráculos Especificados

Juzgan el comportamiento usando una especificación creada manualmente.

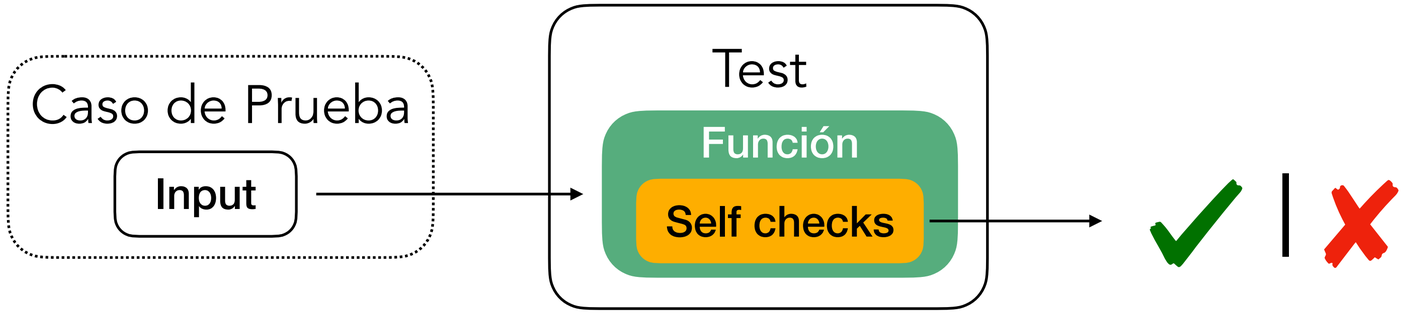



```
@Test
public void testSubString() {
  String str1 = "abc";
  String str2 = "abc";
  boolean b = isSubString(str1, str2);
  assertTrue(b);
}
```

Cómo podemos extender esto a múltiples tests?

Oráculos Especificados - Self checks

En lugar de comparar valores obtenidos, usar propiedades sobre los resultados para juzgar las secuencias.



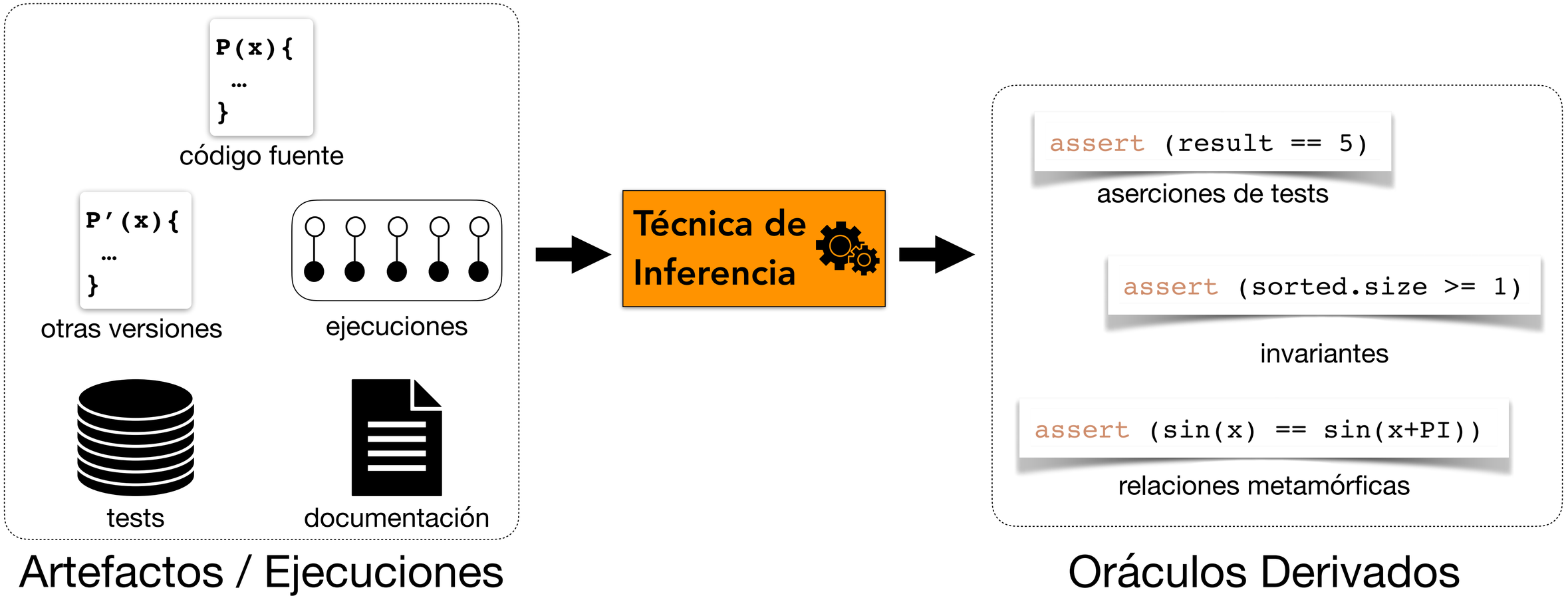
Toman la forma de aserciones, contratos, y otras propiedades lógicas

```
@Test
public void testQuickSort(String[] input) {
  String[] sorted = quickSort(input);
  assert (sorted.size >= 1);
}
```

- Escritos a nivel de función
- Funcionan para cualquier input
- Preciso solo para esas propiedades

Oráculos Derivados (Automáticamente)

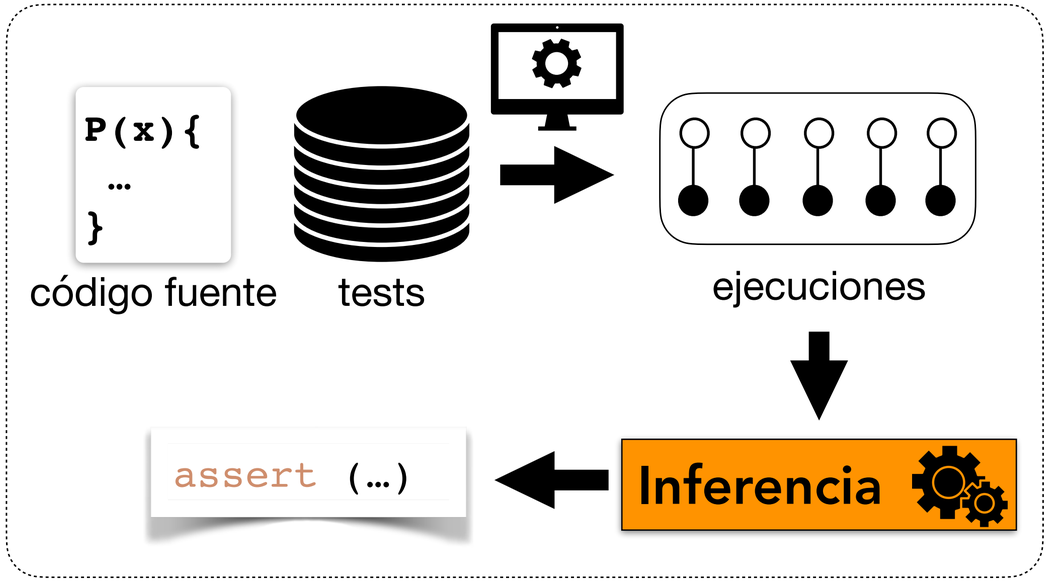
Derivar los oráculos a partir de fuentes de información existentes.



Oráculos Derivados (Automáticamente)

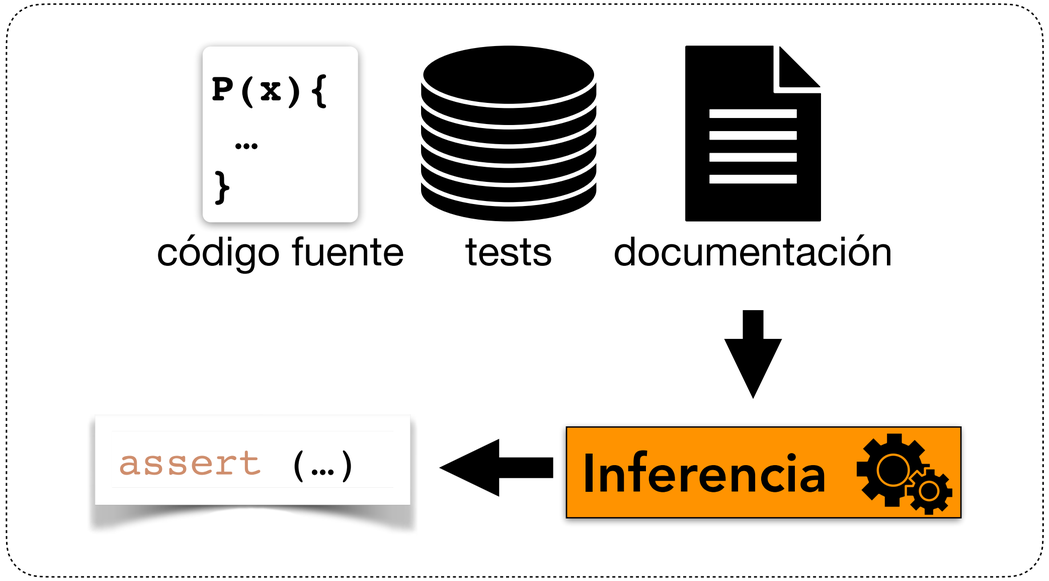
Técnicas Dinámicas

- **Ejecutan** el programa bajo análisis
- Producen oráculos de **regresión**
  - Capturan el comportamiento **actual** del programa



Técnicas Estáticas

- Basadas en procesamiento de lenguaje (natural o código)
- Pueden producir oráculos de **comportamiento esperado**



Generación Automática de Oráculos

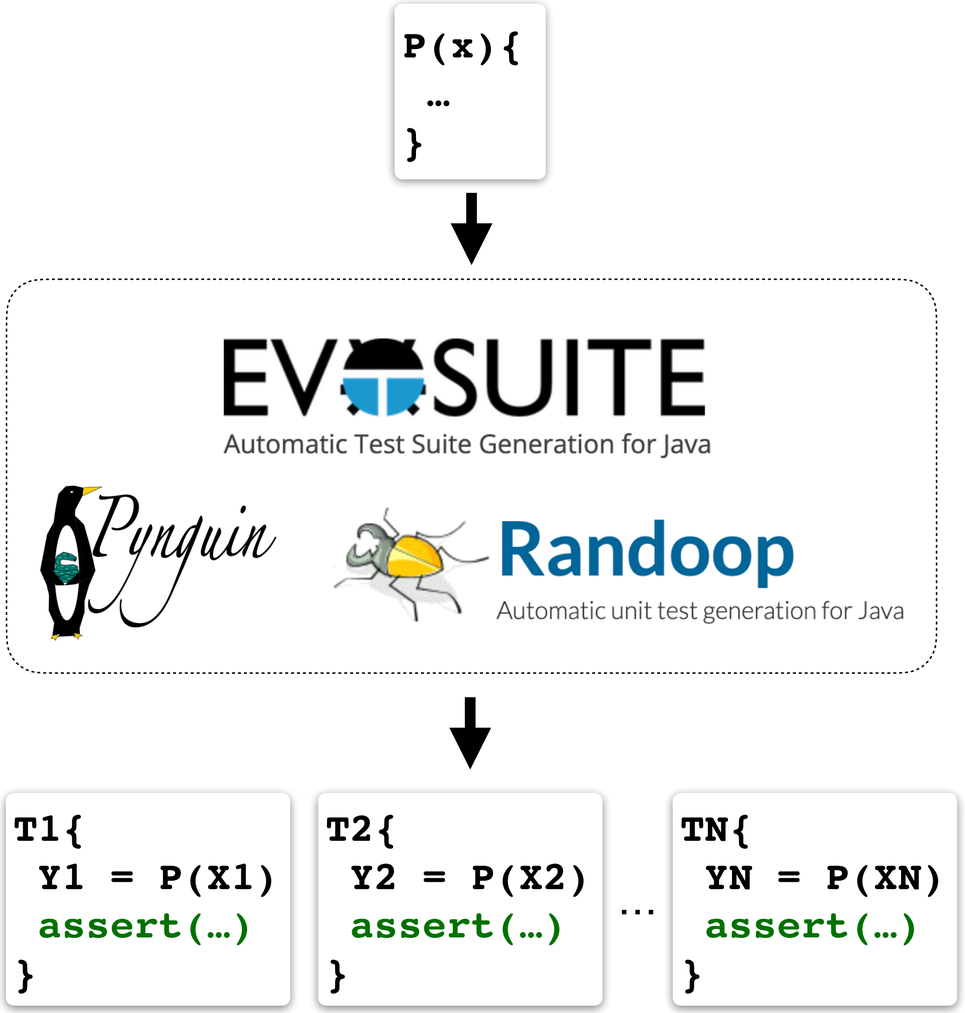
Aserciones de Tests

```
public class StackTests {

  @Test
  public void testPop() {
    Stack s = new Stack();
    int a = 2;

    s.push(a);
    s.pop();

    assert( ??? );
  }
  …
```

Generación Automática de Oráculos

Aserciones de Tests - Técnicas Dinámicas

- Por lo general, la tarea principal es generación automática de tests
- Para generar oráculos, los tests son **ejecutados**
- Oráculos basados en **comparaciones** a partir del comportamiento observado



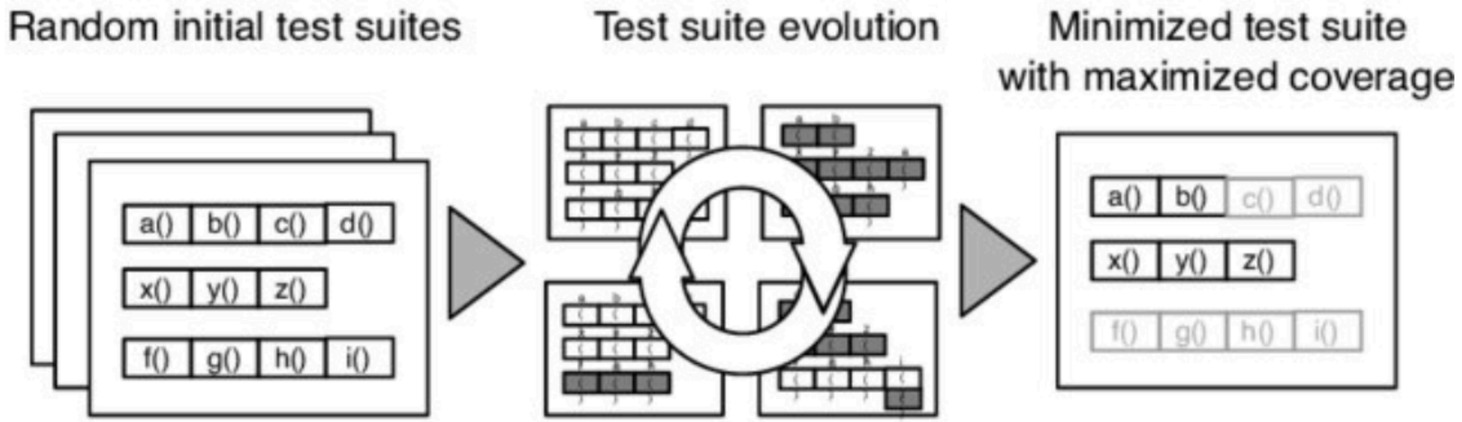
Generación Automática de Oráculos

Aserciones de Tests - Técnicas Dinámicas: EvoSuite

- Generación automática de tests para clases Java
- Herramienta estado del arte, +10 años dev.
- Enfoque evolutivo: **algoritmo genético**
- Optimiza criterios de cobertura: **branch coverage**
- Incorpora aserciones de **regresión** en los tests

EVOSUITE
Automatic Test Suite Generation for Java

https://www.evosuite.org/

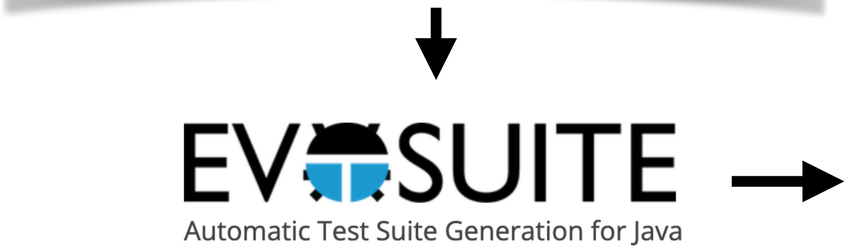


Gordon Fraser and Andrea Arcuri. **EvoSuite: Automatic Test Suite Generation for Object-Oriented Software.** FSE 2011

Generación Automática de Oráculos

Aserciones de Tests - Técnicas Dinámicas: EvoSuite

```
public class Stack {

  public void push(int n) {
    …
  }

  public void pop() {
    …
  }
```



```
public class StackTests {

  @Test
  public void testPop() {
    Stack s = new Stack();

    s.push(2);
    s.pop();

    boolean empty = s.isEmpty();
    assertTrue(empty);
  }
  …
```

Gordon Fraser and Andrea Arcuri. **EvoSuite: Automatic Test Suite Generation for Object-Oriented Software.** FSE 2011

Generación Automática de Oráculos

Aserciones de Tests - Técnicas Dinámicas

- Útiles para equipar con aserciones código existente “correcto”
- Testing más efectivo en el futuro
- Las aserciones pueden ser incorrectas si la implementación es incorrecta.

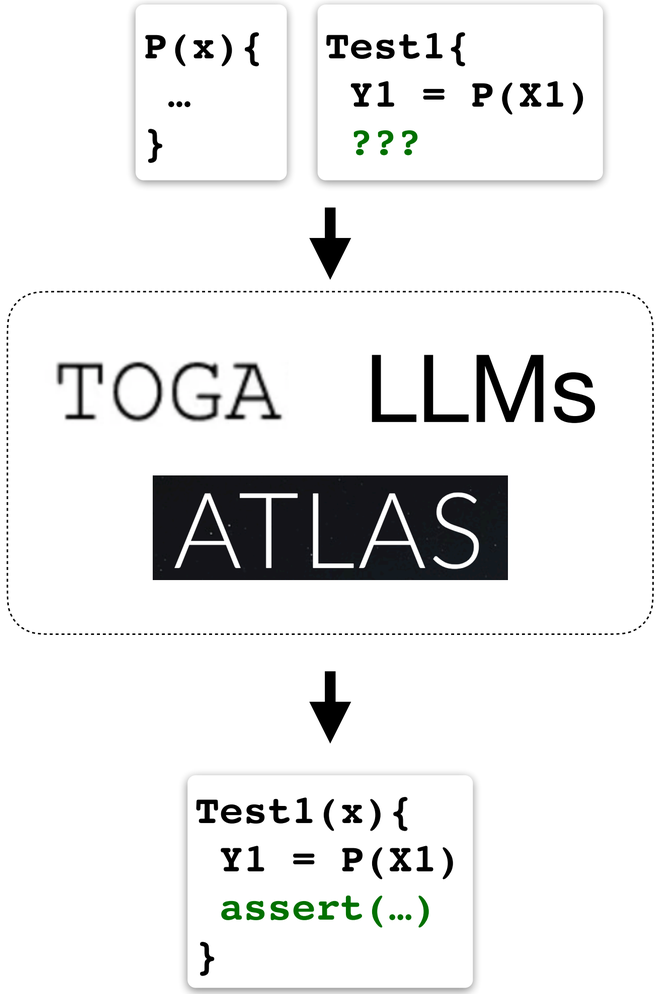
```
public class Stack {

  public void pop() {
     // NO-OP
   }

  …
```

```
  @Test
  public void testPop() {
    Stack s = new Stack();

    s.push(2);
    s.pop();

    boolean empty = s.isEmpty();
    assertFalse(empty);
  }
```

Generación Automática de Oráculos

Aserciones de Tests - Técnicas Estáticas

- **No ejecutan** los programas bajo análisis
- Por lo general generan aserciones para un test específico (usando también el contexto)
- Los oráculos intentan *predecir* el comportamiento **esperado**



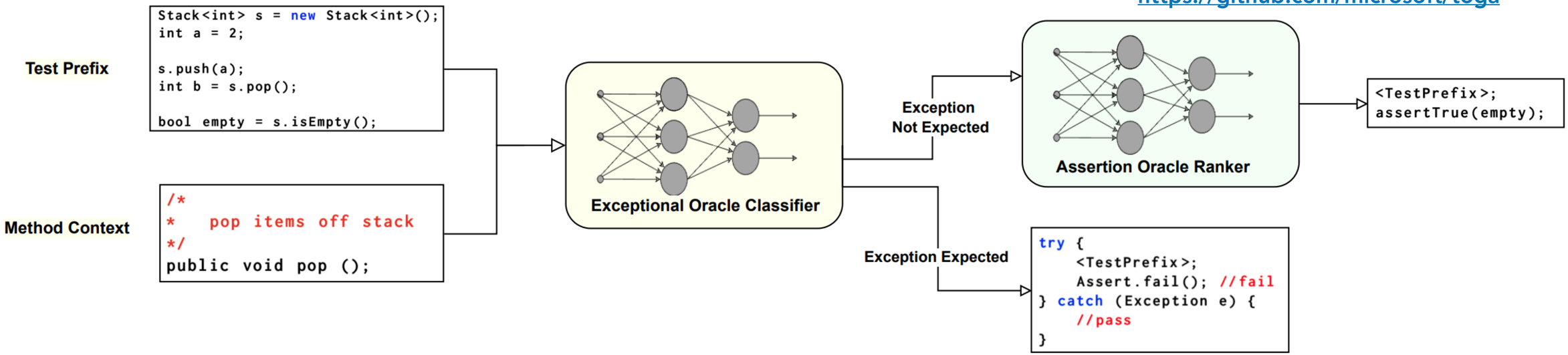
Generación Automática de Oráculos

Aserciones de Tests - Técnicas Estáticas: TOGA

- Herramienta prototípica
- Enfoque basado en **modelos neuronales generativos**
- Oráculos para excepciones y aserciones

TOGA

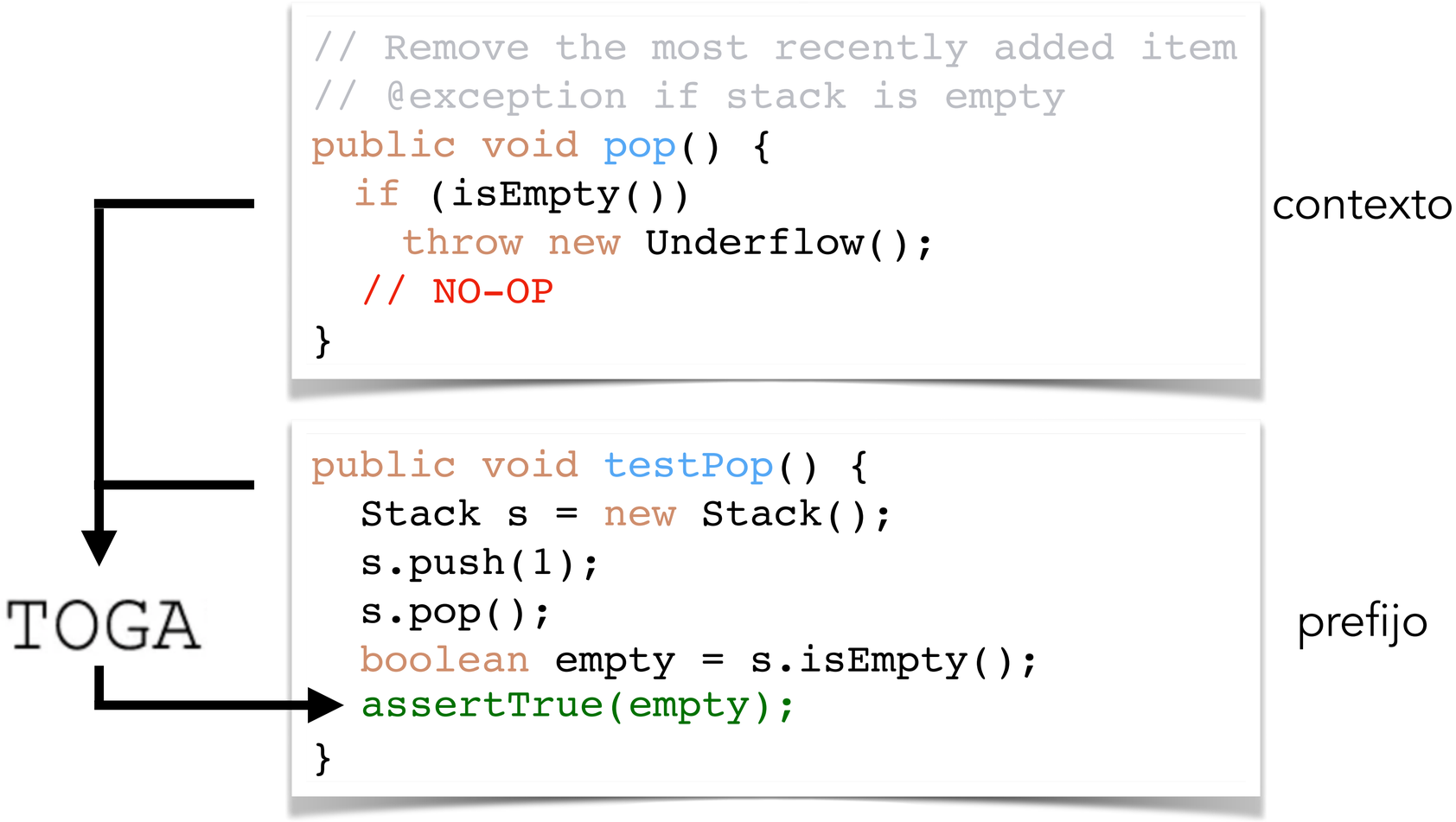
https://github.com/microsoft/toga



Elizabeth Dinella et al. **TOGA: A Neural Method for Test Oracle Generation.** ICSE 2022

Generación Automática de Oráculos

Aserciones de Tests - Técnicas Estáticas: TOGA



Elizabeth Dinella et al. **TOGA: A Neural Method for Test Oracle Generation.** ICSE 2022

Generación Automática de Oráculos

Aserciones de Tests - Técnicas Estáticas: TOGA

```
public void test1()  throws Throwable  {
Locale locale0 = new Locale("TZea6h)b", "LE$&{r\f+E=b+Uz}rR",
      "TZea6h)b");
Locale locale1 = Locale.KOREAN;
List<Locale> list0 = LocaleUtils.localeLookupList(locale0,
      locale1);
assertEquals(4, list0.size()); /*EvoSuite Assertion*/
///AssertionError: expected:<1> but was:<4>*
assertEquals(1,list0.size());///false positive assertion by TOGA
}
```

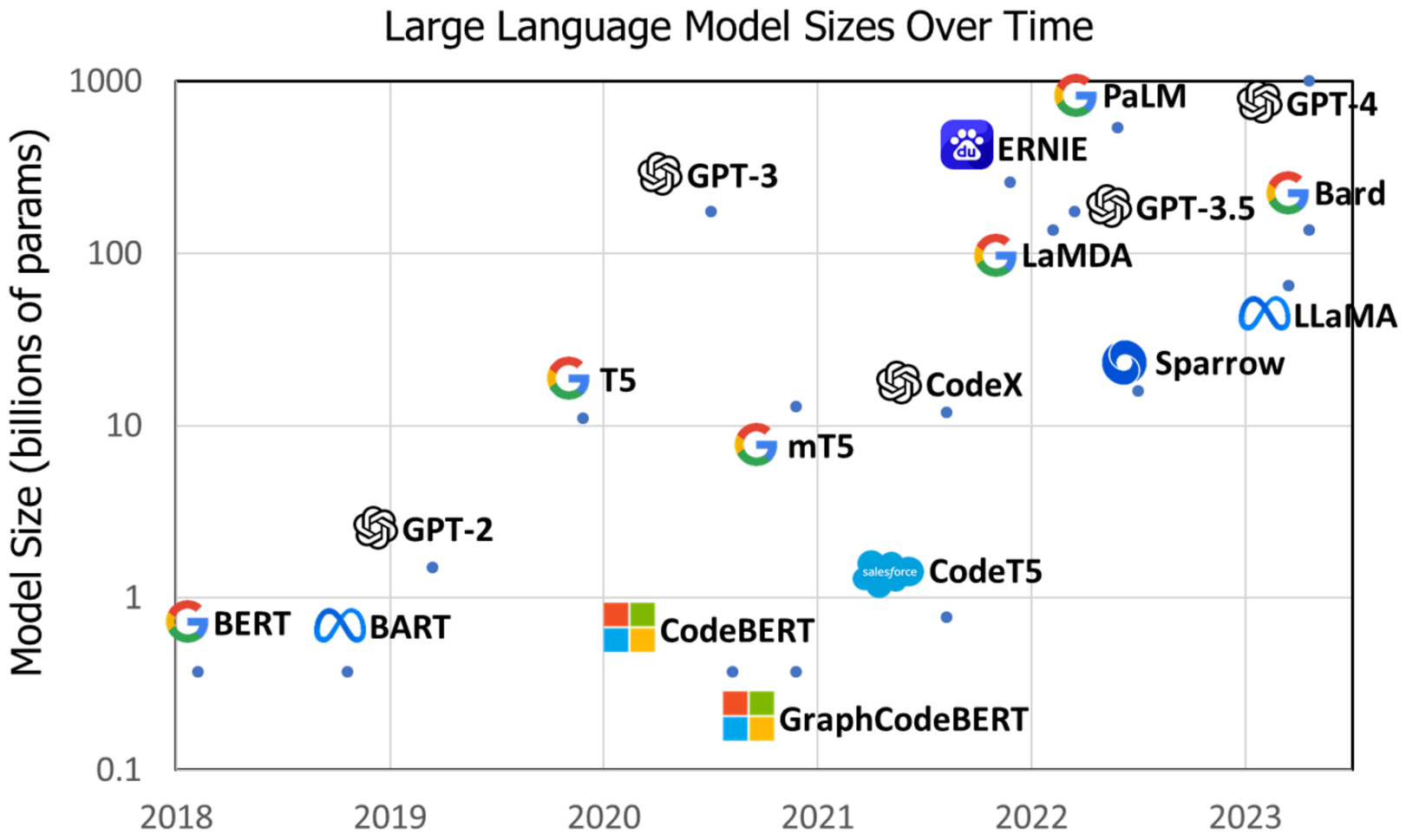
```
public void test1() throws Throwable  {
MutableLong mutableLong0 = new MutableLong();
MutableLong mutableLong1 = new MutableLong();
mutableLong1.incrementAndGet();
boolean boolean0 = mutableLong0.equals(mutableLong1);
assertEquals((short)1, mutableLong1.shortValue()); /*EvoSuite
      Assertion*/
///incompatible types: short cannot be converted to boolean
assertTrue(mutableLong1.shortValue()); ///TOGA assertion with
      type error"
}
```

- (+) Los oráculos de TOGA ayudan a detectar bugs (19 de 57, 33%)
- (-) TOGA produce una gran cantidad de falsos positivos (47%)

Soneya Hossain et al. **Neural-Based Test Oracle Generation: A Large-Scale Evaluation and Lessons Learned.** FSE 2023

Generación Automática de Oráculos

Aserciones de Tests - Técnicas Estáticas: LLMs



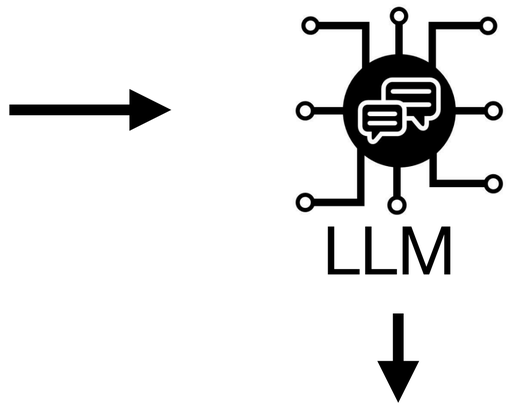
Generación Automática de Oráculos

Aserciones de Tests - Técnicas Estáticas: LLMs

```
@Test
public void testPop() {
  Stack<int> s = new Stack<int>();
  s.push(2);
  s.push(3);
  s.push(5);
  s.pop();
  ?
```

+

Prompt: “Generate as many assertions as possible for this Java test”




```
.....
.....
assertFalse(s.isEmpty());
assertTrue(s.size() == 2);
```

Generación Automática de Oráculos

Aserciones de Tests - Técnicas Estáticas: LLMs

ChatGPT(GPT-3.5)

```
@Test
public void testPop() {
    Stack<Integer> s = new Stack<>();

    s.push(2);
    s.push(3);
    s.push(5);
    s.pop();

    // Assertion 1: Check if the stack size is correct after poppi
    assertEquals(2, s.size());

    // Assertion 2: Check if the top element of the stack is corre
    assertEquals(3, (int) s.peek());

    // Assertion 3: Check if the popped element matches the expecte
    assertEquals(5, (int) s.pop());

    // Assertion 4: Check if the stack is not empty after pushing
    assertFalse(s.isEmpty());

    // Assertion 5: Check if the stack does not contain the popped
    assertFalse(s.contains(5));
```

Bard

```
@Test
public void testPop() {
  Stack<int> s = new Stack<int>();

  s.push(2);
  s.push(3);
  s.push(5);

  // Assert that the stack is not empty
  assertFalse(s.isEmpty());

  // Assert that the top element of the stack is 5
  assertEquals(5, s.pop());

  // Assert that the stack is now smaller by one element
  assertEquals(2, s.size());

  // Assert that the top element of the stack is now 3
  assertEquals(3, s.pop());

  // Assert that the top element of the stack is now 2
  assertEquals(2, s.pop());

  // Assert that the stack is now empty
  assertTrue(s.isEmpty());
```

Generación Automática de Oráculos

Invariantes

```
public class Stack {

  private int[] array;
  private int topOfStack;

  public Stack(int capacity) {
    if (capacity <= 0) throw new Illegal…
    array = new int[capacity];
    topOfStack = -1;
  }

  public void pop() {
    if (isEmpty())
      throw new Underflow();
    array[topOfStack--] = null;
  }
  …
}
```

Invariantes de clase

```
size() >= 0
```

Precondiciones

```
capacity > 0
```

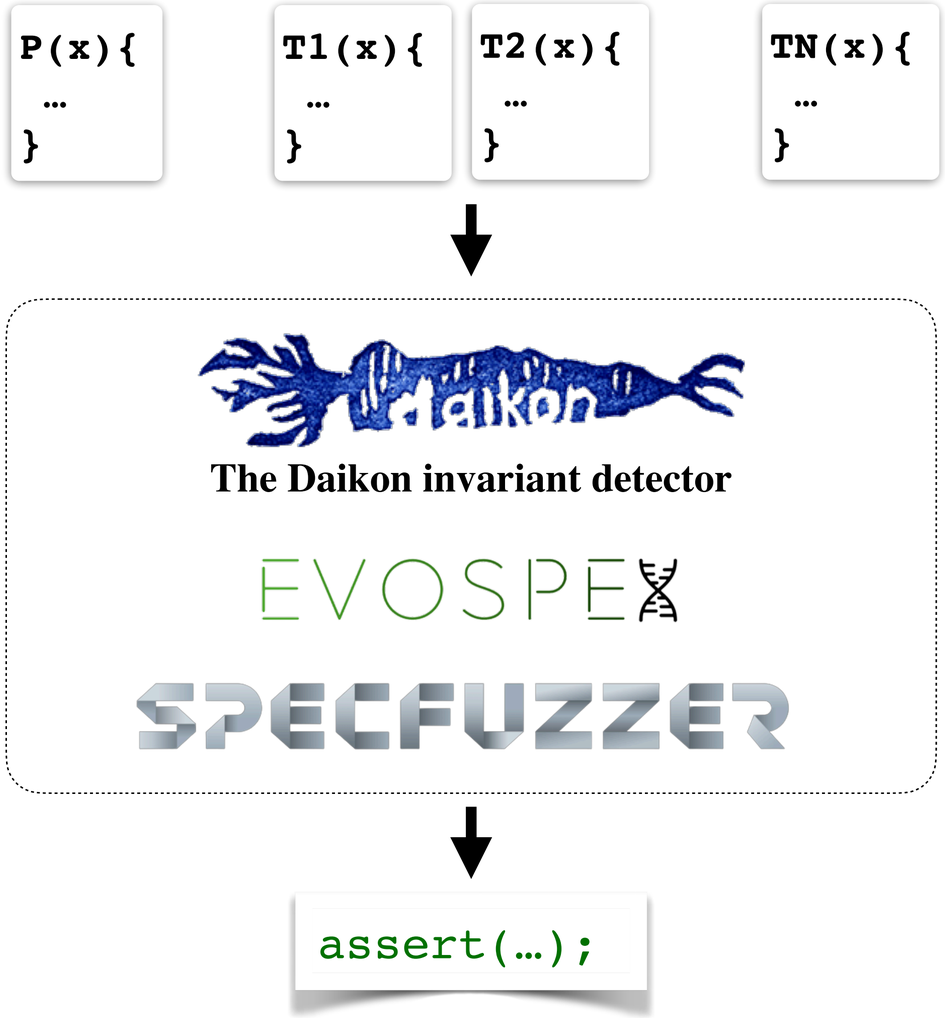
Postcondiciones

```
topOfStack+1 == old(topOfStack)
```

Generación Automática de Oráculos

Invariantes - Técnicas Dinámicas

- Requieren **tests** como inputs
- Los invariantes se derivan a partir de **ejecutar** los tests
- Los oráculos son **self-checks**
  - Capturan el comportamiento **actual**
  - Basados en propiedades que se cumplen en **todos** los tests observados



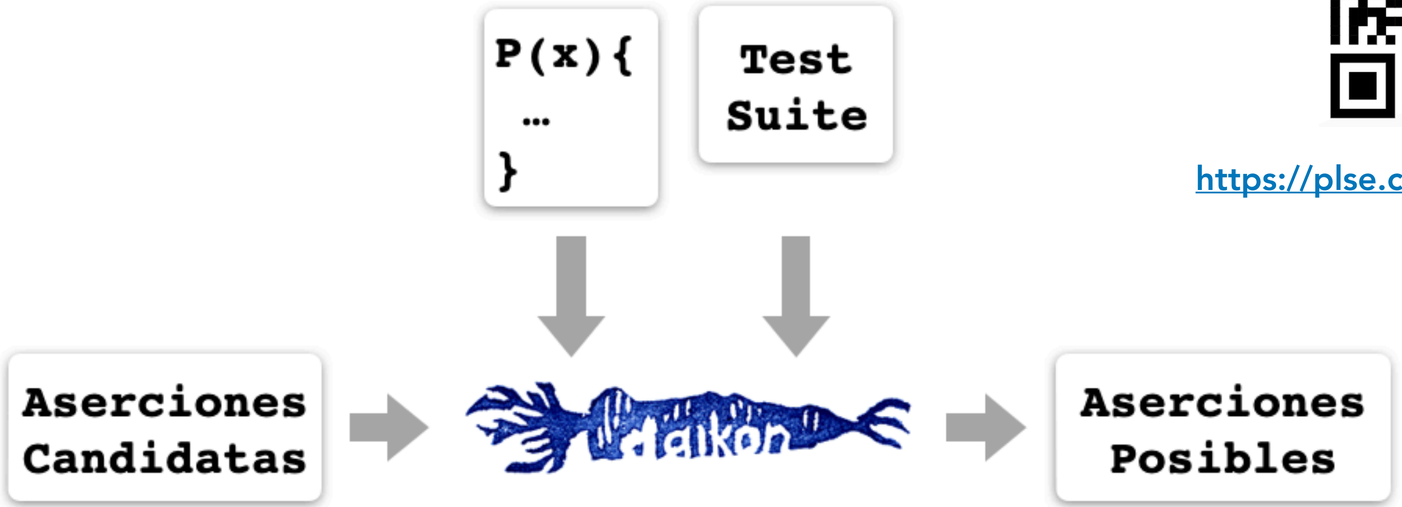
Generación Automática de Oráculos

Invariantes - Técnicas Dinámicas - Daikon

- Detección **dinámica** de **invariantes probables**
- Herramienta con +20 años de desarrollo
- Funciona para C, C++, C#, Eiffel, Java…
- Conjunto fijo de propiedades (todas las que no fueron inválidadas son reportadas)

The Daikon invariant detector

https://plse.cs.washington.edu/daikon/



Michael D. Ernst et al. **The Daikon System for Dynamic Detection of Likely Invariants** SCP 2007

Generación Automática de Oráculos

Invariantes - Técnicas Dinámicas - Daikon

```
public class Stack {

  int[] array, int topOfStack;

  public Stack(int capacity) {
    if (capacity <= 0)
      throw new Illegal…
    array = new int[capacity];
    topOfStack = -1;
  }

  public void pop() {
    if (isEmpty())
      throw new Underflow();
    array[topOfStack--] = null;
  }
  …
```

```
invariants:
topOfStack >= -1

push - preconditions: true
push - postconditions:
array[topOfStack] == x

pop - postconditions
topOfStack+1 == old(topOfStack)
```

- Propiedades lógicas/aritméticas simples (extendidas manualmente)
- Invariantes reportados “débiles”

Michael D. Ernst et al. **The Daikon System for Dynamic Detection of Likely Invariants** SCP 2007

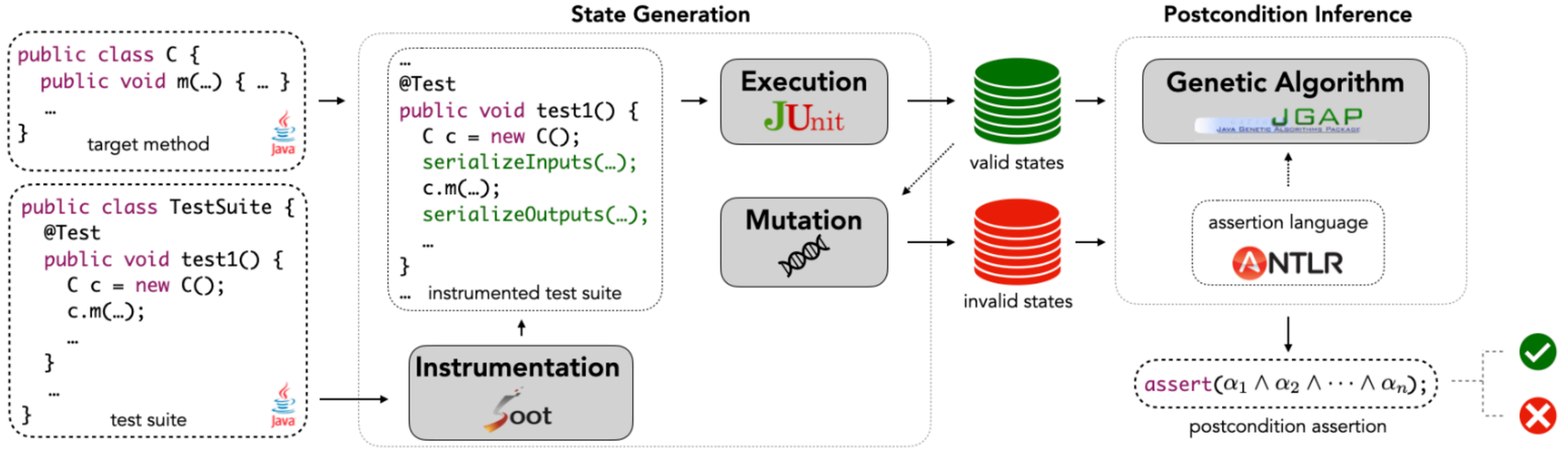
Generación Automática de Oráculos

Invariantes - Técnicas Dinámicas - EvoSpex

- Inferencia de **postcondiciones** de métodos Java
- Computación evolutiva: **algoritmo genético**
- Además de usar el comportamiento actual, usa comportamiento potencialmente inválido

EVOSPEX

https://github.com/facumolina/evospex



Facundo Molina et al. **EvoSpex: An Evolutionary Algorithm for Learning Postconditions.** ICSE 2021

Generación Automática de Oráculos

Invariantes - Técnicas Dinámicas - EvoSpex

```
public class TreeList<E> extends AbstractList<E> {
  private AVLNode<E> root;
  private int size;

  // Adds a new element to the list.
  public void add(int index, E obj) {
    modCount++;
    checkInterval(index, 0, size()); // Checks whether the index is valid.
    if (root == null) {
      root = new AVLNode<>(index, obj, null, null);
    } else {
      root = root.insert(index, obj);
    }
    size++;
  }

  private static class AVLNode<E> {
    private E value;
    private int height, relativePosition;
    private boolean leftIsPrevious, rightIsNext;
    private AVLNode<E> left, right;

    // Inserts a node at the position index.
    public AVLNode<E> insert(int index, E obj) {
      int indexRelativeToMe = index - relativePosition;
      if (indexRelativeToMe <= 0) {
        return insertOnLeft(indexRelativeToMe, obj);
      }
      return insertOnRight(indexRelativeToMe, obj);
    }
    ...
  }
}
```

```
obj in this.root.*(left+right).value
```

El elemento *obj* fue insertado en el árbol

```
#(old(this).root.*(left+right))=this.size-1
```

El valor del campo *size* fue incrementado y coincide con la cantidad de elementos del árbol

```
index in this.root.*(left).height
```

el valor de index corresponde a un índice válido

- Lenguaje más enfocado en propiedades estructurales (que incluyen cuantificación, pertenencia, etc)

Facundo Molina et al. **EvoSpex: An Evolutionary Algorithm for Learning Postconditions.** ICSE 2021

Generación Automática de Oráculos

Invariantes - Técnicas Estáticas - JDoctor

- Única técnica estática para inferencia de invariantes
- Genera **precondiciones/postcondiciones** a partir de comentarios en Javadoc
- Basada en métodos de NLP

JDoctor

https://github.com/albertogoffi/toradocu

```
// Remove the most recently
added item
// @exception if stack is empty
public void pop() {
  …
}
```

→ JDoctor →

```
isEmpty() implies
IllegalStateException
```

Arianna Blassi et al. **Translating Code Comments to Procedure Specifications.** ISSTA 2018

Generación Automática de Oráculos

Invariantes - Técnicas Estáticas - JDoctor

Precondiciones

Postcondiciones Excepcionales

```
1  /**
2   * @param funnel the funnel of T's that the constructed {@code
3   *     BloomFilter<T>} will use
4   * @param expectedInsertions the number of expected insertions to the
5   *     constructed {@code BloomFilter<T>}; must be positive
6   * @param fpp the desired false positive probability (must be positive
7   *     and less than 1.0)
8   */
9  public static <T> BloomFilter<T> create(
10   Funnel<? super T> funnel,
11   int expectedInsertions,
12   double fpp) { ... }
```

```
expectedInsertions > 0
```

```
fpp > 0 && fpp < 1.0
```

```
1  /**
2   * @throws NullPointerException if either collection or the comparator is null
3   */
4  public static <O> List<O> collate(
5    Iterable<? extends O> a,
6    Iterable<? extends O> b,
7    Comparator<? super O> c) { ··· }
```

```
(a == null || b == null || c == null)
                implies
    java.lang.NullPointerException
```

Arianna Blassi et al. **Translating Code Comments to Procedure Specifications.** ISSTA 2018

Generación Automática de Oráculos

Invariantes - Técnicas Estáticas - JDoctor

Postcondiciones Normales

```
1 | /** @return an empty Bag */
2 | Bag emptyBag()
```

```
result.equals(Bag.EMPTY_BAG)
```

- En ocasiones, los comentarios estructurados pueden aportar información valiosa
- Las propiedades pueden usarse tanto para detectar bugs como para detectar inconsistencias en los comentarios

Arianna Blassi et al. **Translating Code Comments to Procedure Specifications.** ISSTA 2018

Generación Automática de Oráculos

Relaciones Metamórficas

- Una relación metamórfica es una propiedad necesaria el software bajo análisis:
  - Una propiedad de la función *sin* es que **sin(x) = sin(pi - x)**
  - Entonces, sin(x) y sin(pi - x) tienen la misma salida esperada.

- Si este tipo de relaciones no se cumplen, entonces hay un bug.

Generación Automática de Oráculos

Relaciones Metamórficas

- En cualquier implementación de **Stack** la operación pop debería deshacer la operación push:

```
@Test
public void testPop() {
  Stack s = new Stack();
  s.push(1);
  s.pop();
}
```

```
@Test
public void testPop() {
  Stack s = new Stack();
  s.push(2);
  s.push(3);
}
```

```
@Test
public void testPop() {
  Stack s = new Stack();
  s.push(1);
  s.clear();
 }
```

```
s = s.push();s.pop();
```

Generación Automática de Oráculos

Relaciones Metamórficas - Técnicas Dinámicas - SBES

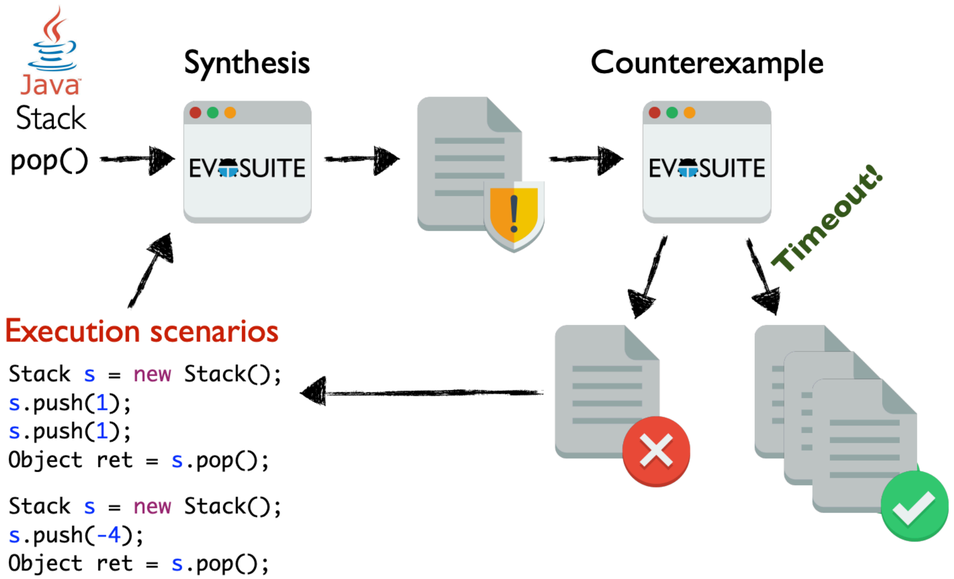
- Detección **dinámica** de **relaciones metamórficas**
- **Secuencias de métodos equivalentes a un método**
- Basada en un método evolutivo (EvoSuite)

SBES

Search-**B**ased synthesis
of **E**quivalent method **S**equences

https://github.com/andreamattavelli/sbes

Search-based Synthesis of Equivalences



Alberto Goffi et al. **Search-based Synthesis of Equivalent Method Sequences.** FSE 2014

Generación Automática de Oráculos

Relaciones Metamórficas - Técnicas Dinámicas - SBES

```
public class Stack<E>
  extends AbstractList<E>
  implements List<E> {

  // Stack methods
  public Stack(int capacity) { … }
  public void push(E item) { … }
  public E pop() { … }
  public E peek() { … }
  …

  // List methods
  public void add(E item) { … }
  public E addLast() { … }
  public E firstElement() { … }
  …
}
```

Alberto Goffi et al. **Search-based Synthesis of Equivalent Method Sequences.** FSE 2014

Generación Automática de Oráculos

Relaciones Metamórficas - Técnicas Dinámicas - SBES

Método objetivo

```
public void pop() { … }
```

```
public E clear() { … }
```

```
public void push(E item) { … }
```

Secuencias equivalentes

```
remove(size()-1)
```

```
removeAllElements()
```

```
add(item)
```

```
add(size(),item)
```

Alberto Goffi et al. **Search-based Synthesis of Equivalent Method Sequences.** FSE 2014

Generación Automática de Oráculos

Relaciones Metamórficas - Técnicas Estáticas - MeMo

- Similar a JDoctor, pero para **relaciones metamórficas**
- Funciona a partir de comentarios en Javadoc
- Basada en NLP

MeMo

https://github.com/ariannab/MeMo

```
// Like add(I, size())
public void addLast(int i) {
  …
}
```

→ MeMo →

```
addLast(i) = add(i, size())
```

Arianna Blassi et al. **MeMo: Automatically identifying metamorphic relations in Javadoc comments for test automation.** SCP 2021

Generación Automática de Oráculos

Relaciones Metamórficas - Técnicas Estáticas - MeMo

Método documentado

```
/** Equivalent to newReentrantLock(lockName, false). */
public ReentrantLock newReentrantLock(String lockName) { ... }
```

Output de MeMo

```
newReentrantLock(lockName) =
newReentrantLock(lockName, false)
```

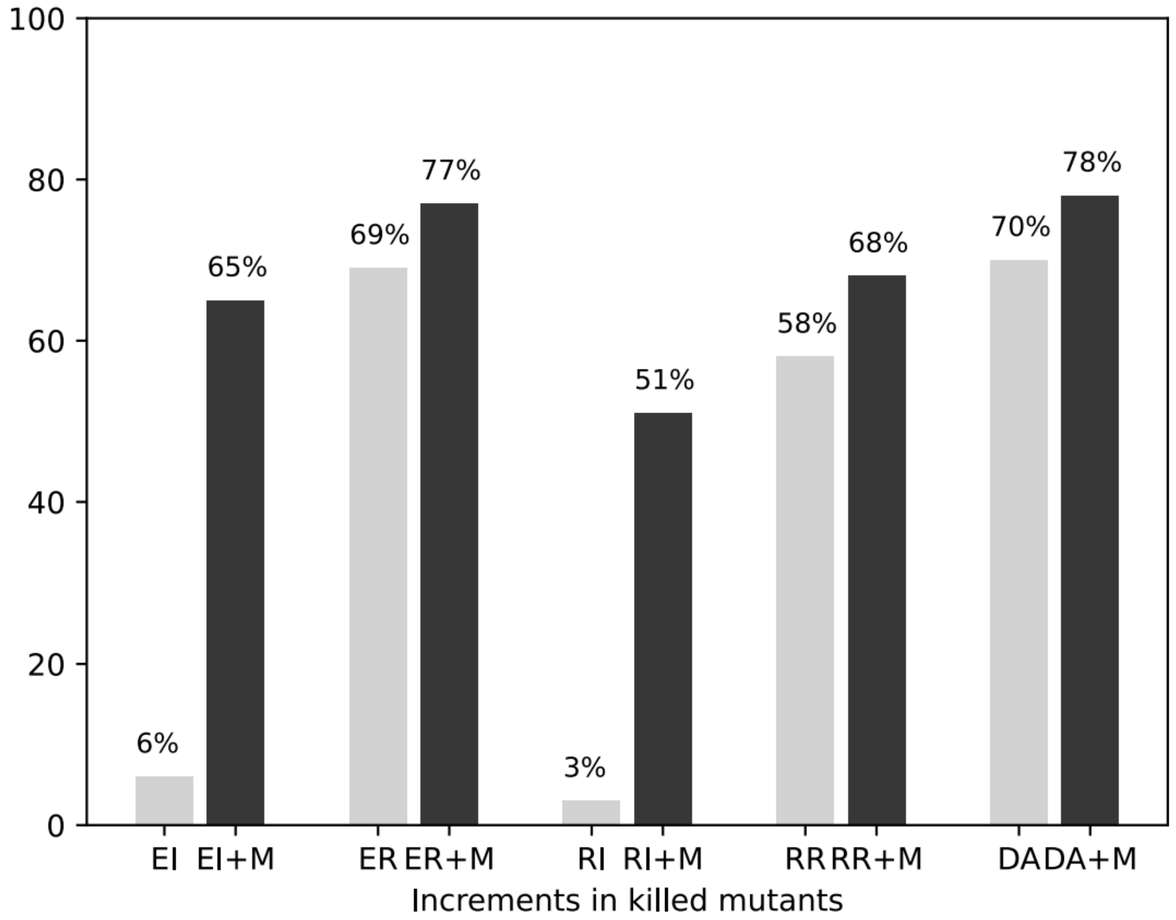
```
/** Calling this method is equivalent to call composeInverse(r,
RotationConvention.VECTOR_OPERATOR). */
public
applyInverseTo(org.apache.commons.math3.geometry.euclidean.
        threed.FieldRotation<T>
 r) { ...
```

```
applyInverseTo(r) = composeInverse(r,
                    VECTOR_OPERATOR)
```

Arianna Blassi et al. **MeMo: Automatically identifying metamorphic relations in Javadoc comments for test automation.** SCP 2021

Generación Automática de Oráculos

Relaciones Metamórficas - Técnicas Estáticas - MeMo



- La detección de fallas puede ser mejorada al incorporar las relaciones metamórficas a tests equipados con aserciones de regresión (EvoSuite)

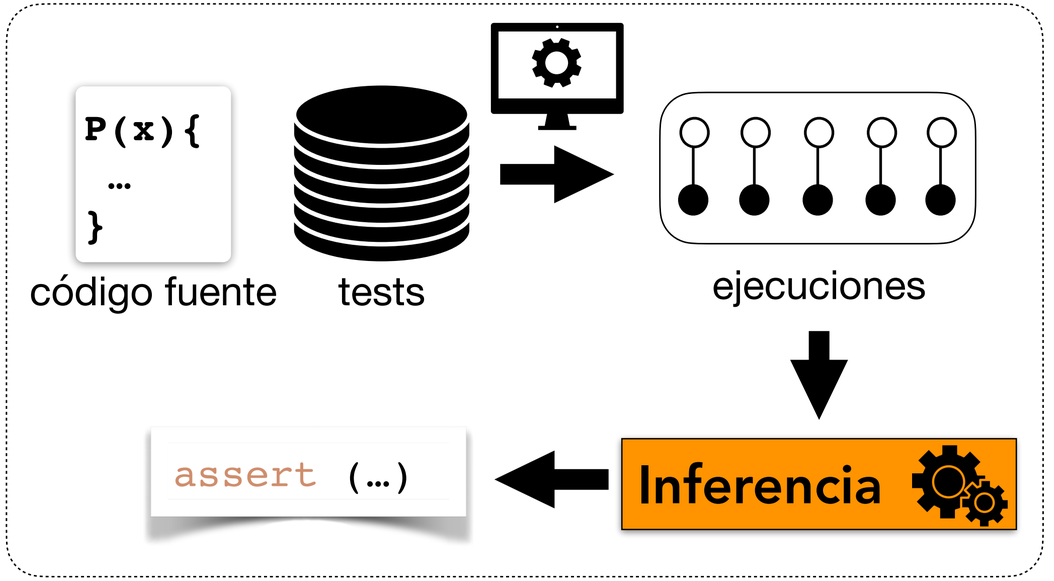
Arianna Blassi et al. **MeMo: Automatically identifying metamorphic relations in Javadoc comments for test automation.** SCP 2021

Oráculos Derivados

Conclusión: Técnicas Dinámicas o Estáticas?

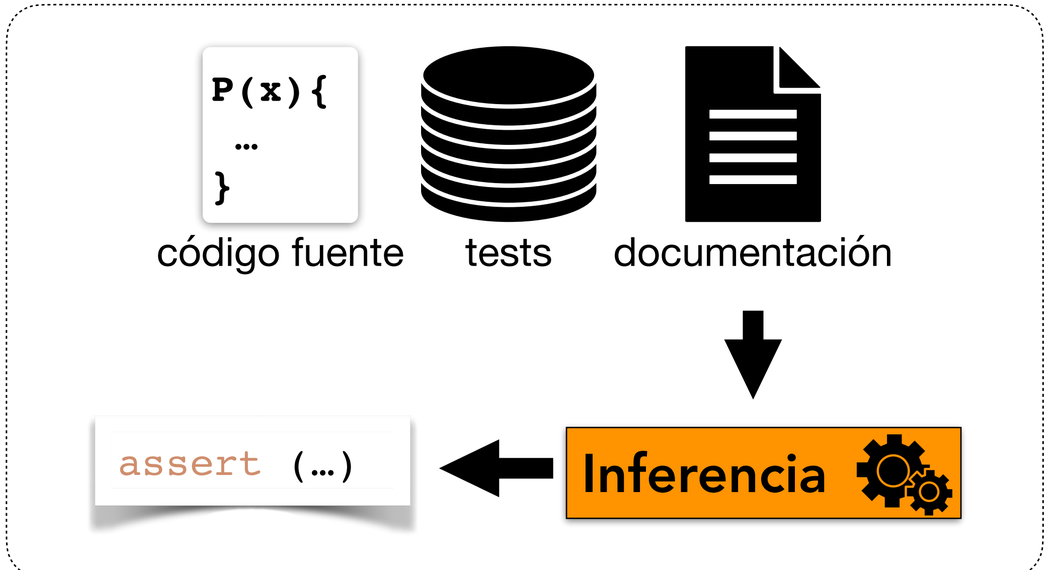
Técnicas Dinámicas

- La precisión (falsos positivos) es "tratable"
- Detección de bugs:
  - Propiedades "sospechosas"
  - Testing en versiones futuras



Técnicas Estáticas

- Por lo general, menos costosas
- Problemas de precisión menos tratables
- Pueden tener mayor efectividad en detección de bugs



Cómo manejamos la falta de oráculos?

Oráculos Humanos

- Si la automatización no es posible o no existe una especificación, un humano siempre puede juzgar la salida manualmente.

- No es lo ideal, pero es muy común en la práctica.

Mathematical genius Shakuntala Devi, nicknamed as “human computer”

Oráculos Humanos - Crowdsourcing

- Desarrollo reciente - subcontratar muchos oráculos humanos distintos distribuyendo el problema.
  - Amazon Mechanical Turk, Mob4Hire, MobTest, uTest.

- No se puede esperar que los usuarios tengan mucho conocimiento del dominio, por lo que la compresión de los inputs y la documentación son muy importantes.

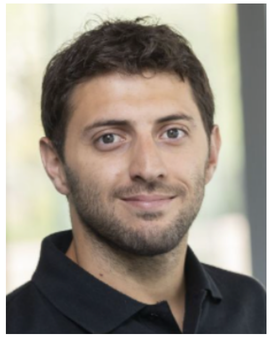
Oráculos para Tests - Resumen

	Costo	Precisión	Completitud
Oráculos Implícitos			
Especificación Manual			
Aserciones de Tests			
Detección de Invariantes			
Testing Metamórfico			

IMDEA Software Institute

Software Testing and Analysis

Alessandra Gorla
Associate Research Professor



Facundo Molina
Postdoctoral Researcher

Juan Manuel Copia
PhD Student

Agustín Nolasco
Researcher Intern (incoming)

~20 Faculty
~50 investigadores
& estudiantes

Oportunidades:

- PhD Students
- Internships

Gracias!