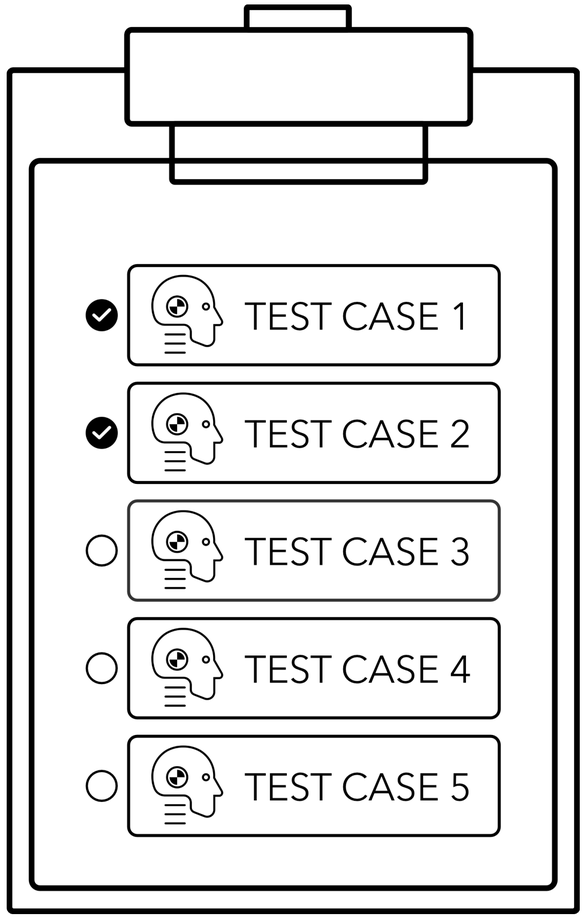
Davide Molinelli
Alberto Martín López
Elliott Zackrone
Beyza Eken
Michael D. Ernst
Mauro Pezzè

Tratto: A Neuro-Symbolic Approach to Deriving Axiomatic Test Oracles

ISSTA 2025

Università della Svizzera italiana
UNIVERSITY of WASHINGTON
CONSTRUCTOR UNIVERSITY

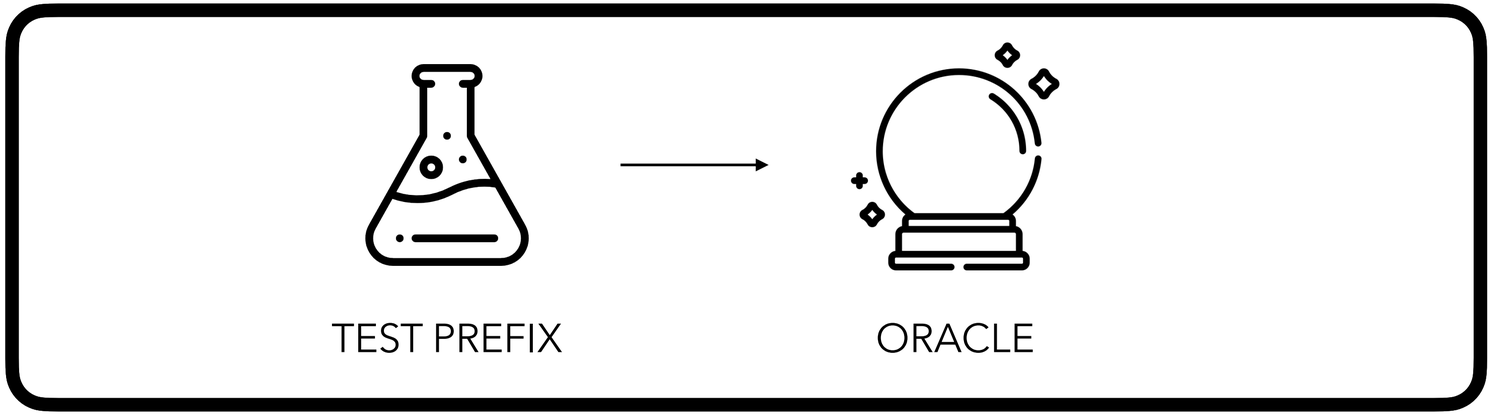
Unit Test = Prefix + Oracle



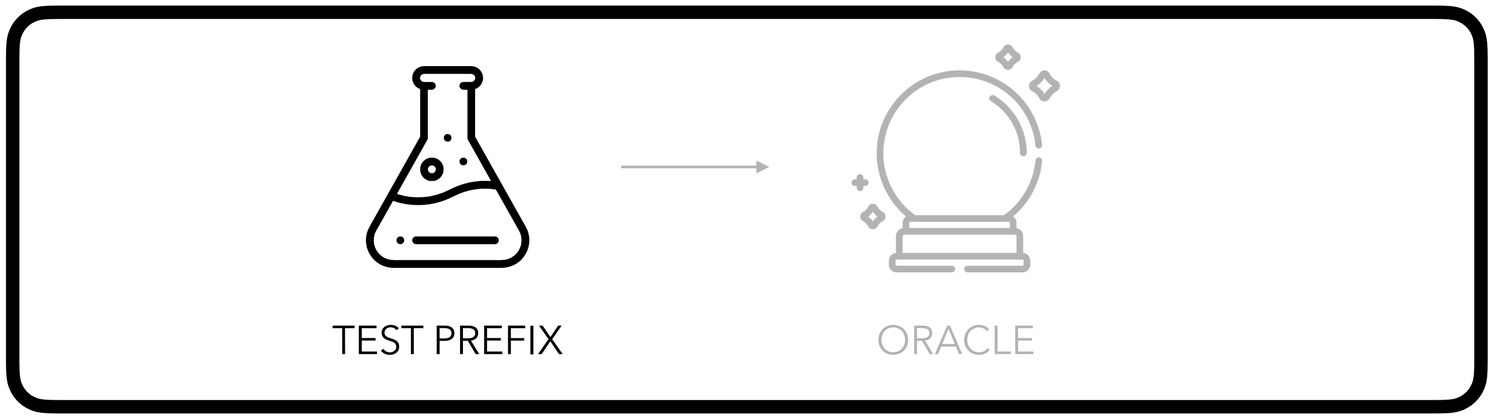
TEST SUITE

Unit Test = Prefix + Oracle

Unit Test = Prefix + Oracle

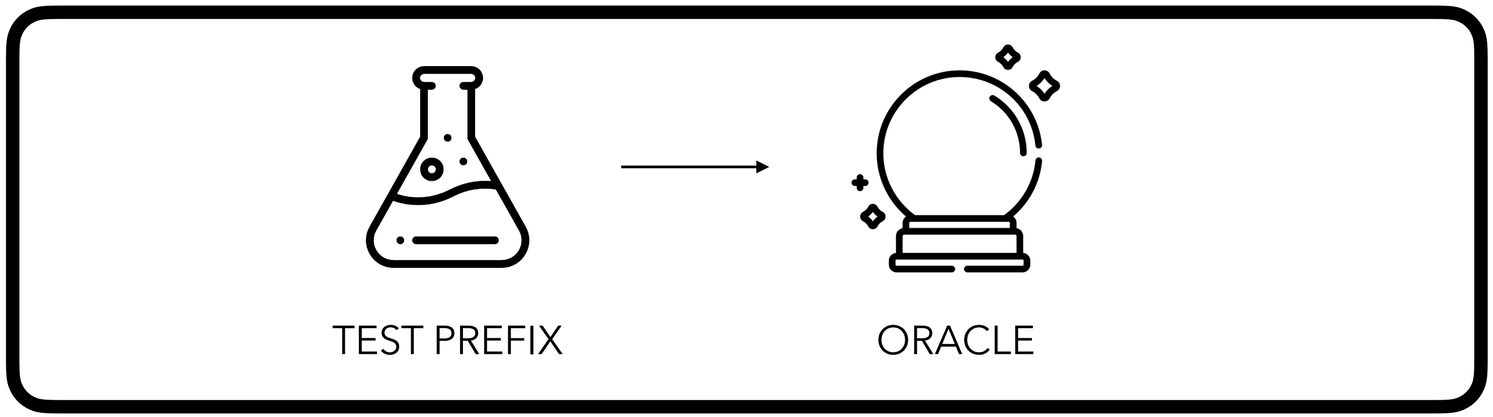


Unit Test = Prefix + Oracle



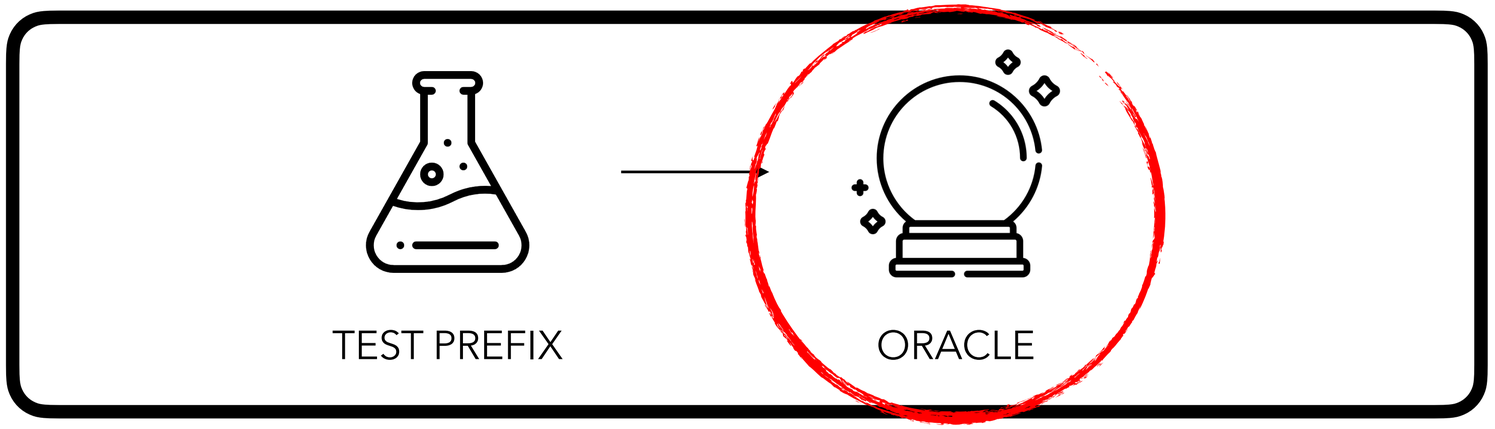

```
@Test(timeout = 4000)
public void testCase() throws Throwable  {
    Car car1 = new Car();
    Car car2 = new Car();
    boolean same = car1.equals(car2);
    assertFalse(same);
}
```

Unit Test = Prefix + Oracle




```
@Test(timeout = 4000)
public void testCase() throws Throwable  {
    Car car1 = new Car();
    Car car2 = new Car();
    boolean same = car1.equals(car2);
    assertFalse(same);
}
```

The Oracle Problem




```
@Test(timeout = 4000)
public void testCase() throws Throwable  {
    Car car1 = new Car();
    Car car2 = new Car();
    boolean same = car1.equals(car2);
    assertFalse(same);
}
```

GENERATING ORACLE IS AN OPEN CHALLENGE

Axiomatic Oracle

```
assert sum(num1, num2) >= num1 && sum(num1, num2) >= num2
```

Program assertions that predicate on input or output parameters of the program

Axiomatic Oracle

```
assert sum(num1, num2) >= num1 && sum(num1, num2) >= num2
```

Program assertions that predicate on input or output parameters of the program

CONCRETE ORACLE

```
assert sum(20, 22) == 42
```

Axiomatic Oracle

```
assert sum(num1, num2) >= num1 && sum(num1, num2) >= num2
```

Program assertions that predicate on input or output parameters of the program

~~CONCRETE ORACLE~~

```
assert sum(20, 22) == 42
```

Axiomatic Oracles

Preconditions

Validity of the test input

Postconditions

Regular behavior

Exceptional postconditions

Exceptional behavior

```
/**
 * The method returns the square root if the argument is greater
 * than or equal to 0.
 *
 * @param num the number to take the square root of. Must be
 *            greater than or equal to 0
 * @return the square root of the number passed to the function
 * @throws IllegalArgumentException if the number is
 *         negative
 */
public int method(int num) throws IllegalArgumentException  {
    if (num < 0) {
        throw new IllegalArgumentException("Negative number");
    }
    return Math.sqrt(num);
}
```

Axiomatic Oracles

Preconditions

Validity of the test input

Postconditions

Regular behavior

Exceptional postconditions

Exceptional behavior

```
/**
 * The method returns the square root if the argument is greater
 * than or equal to 0.
 *
 * @param num the number to take the square root of. Must be
 *            greater than or equal to 0
 * @return the square root of the number passed to the function
 * @throws IllegalArgumentException if the number is
 *         negative
 */
public int method(int num) throws IllegalArgumentException  {
    if (num < 0) {
        throw new IllegalArgumentException("Negative number");
    }
    return Math.sqrt(num);
}
```

```
assert num >= 0
```

Axiomatic Oracles

Preconditions

Validity of the test input

Postconditions

Regular behavior

Exceptional postconditions

Exceptional behavior

```
/**
 * The method returns the square root if the argument is greater
 * than or equal to 0.
 *
 * @param num the number to take the square root of. Must be
 *            greater than or equal to 0
 * @return the square root of the number passed to the function
 * @throws IllegalArgumentException if the number is
 *         negative
 */
public int method(int num) throws IllegalArgumentException  {
    if (num < 0) {
        throw new IllegalArgumentException("Negative number");
    }
    return Math.sqrt(num);
}
```

NO AXIOMATIC ORACLE

Axiomatic Oracles

Preconditions

Validity of the test input

Postconditions

Regular behavior

Exceptional postconditions

Exceptional behavior

```
/**
 * The method returns the square root if the argument is greater
 * than or equal to 0.
 *
 * @param num the number to take the square root of. Must be
 *            greater than or equal to 0
 * @return the square root of the number passed to the function
 * @throws IllegalArgumentException if the number is
 *         negative
 */
public int method(int num) throws IllegalArgumentException  {
    if (num < 0) {
        throw new IllegalArgumentException("Negative number");
    }
    return Math.sqrt(num);
}
```

```
assert num < 0
```

Precondition Example

```
/**
 * Sets the item label generator for a series and
 * sends a {@link RendererChangeEvent} to all
 * registered listeners.
 *
 * @param series the series index (zero based).
 * @param generator the generator
 * (null permitted).
 *
 * @see #getSeriesItemLabelGenerator(int)
 */
public void setSeriesItemLabelGenerator(
    int series,
    CategoryItemLabelGenerator generator
) {
    setSeriesItemLabelGenerator(series, generator, true);
}
```

Method setSeriesItemLabelGenerator from JFreeChart

Precondition Example

```
/**
 * Sets the item label generator for a series and
 * sends a {@link RendererChangeEvent} to all
 * registered listeners.
 *
 * @param series the series index (zero based).
 * @param generator the generator
 * (null permitted).
 *
 * @see #getSeriesItemLabelGenerator(int)
 */
public void setSeriesItemLabelGenerator(
    int series,
    CategoryItemLabelGenerator generator
) {
    setSeriesItemLabelGenerator(series, generator, true);
}
```

Precondition Example

```
/**
 * Sets the item label generator for a series and
 * sends a {@link RendererChangeEvent} to all
 * registered listeners.
 *
 * @param series the series index (zero based).
 * @param generator the generator
 * (null permitted).
 *
 * @see #getSeriesItemLabelGenerator(int)
 */
public void setSeriesItemLabelGenerator(
    int series,
    CategoryItemLabelGenerator generator
) {
    setSeriesItemLabelGenerator(series, generator, true);
}
```

series >= 0

Precondition Example

```
/**
 * Sets the item label generator for a series and
 * sends a {@link RendererChangeEvent} to all
 * registered listeners.
 *
 * @param series the series index (zero based).
 * @param generator the generator
 * (null permitted).
 *
 * @see #getSeriesItemLabelGenerator(int)
 */
public void setSeriesItemLabelGenerator(
    int series,
    CategoryItemLabelGenerator generator
) {
    setSeriesItemLabelGenerator(series, generator, true);
}
```

series >= 0

STATE-OF-THE-ART TECHNIQUES

EvoSuite and **Randoop** do not generate preconditions

Precondition Example

```
/**
 * Sets the item label generator for a series and
 * sends a {@link RendererChangeEvent} to all
 * registered listeners.
 *
 * @param series the series index (zero based).
 * @param generator the generator
 * (null permitted).
 *
 * @see #getSeriesItemLabelGenerator(int)
 */
public void setSeriesItemLabelGenerator(
    int series,
    CategoryItemLabelGenerator generator
) {
    setSeriesItemLabelGenerator(series, generator, true);
}
```

series >= 0

STATE-OF-THE-ART TECHNIQUES

EvoSuite and **Randoop** do not generate preconditions

SYMBOLIC APPROACHES

Jdoctor patterns do not match this text

Precondition Example

```
/**
 * Sets the item label generator for a series and
 * sends a {@link RendererChangeEvent} to all
 * registered listeners.
 *
 * @param series the series index (zero based).
 * @param generator the generator
 * (null permitted).
 *
 * @see #getSeriesItemLabelGenerator(int)
 */
public void setSeriesItemLabelGenerator(
    int series,
    CategoryItemLabelGenerator generator
) {
    setSeriesItemLabelGenerator(series, generator, true);
}
```

series >= 0

STATE-OF-THE-ART TECHNIQUES

EvoSuite and **Randoop** do not generate preconditions

SYMBOLIC APPROACHES

Jdoctor patterns do not match this text

NEURAL APPROACHES

TOGA, **TOGLL**, and **AugmentTest** do not generate preconditions

Precondition Example

```
/**
 * Sets the item label generator for a series and
 * sends a {@link RendererChangeEvent} to all
 * registered listeners.
 *
 * @param series the series index (zero based).
 * @param generator the generator
 * (null permitted).
 *
 * @see #getSeriesItemLabelGenerator(int)
 */
public void setSeriesItemLabelGenerator(
    int series,
    CategoryItemLabelGenerator generator
) {
    setSeriesItemLabelGenerator(series, generator, true);
}
```

series >= 0

STATE-OF-THE-ART TECHNIQUES

EvoSuite and **Randoop** do not generate preconditions

SYMBOLIC APPROACHES

Jdoctor patterns do not match this text

NEURAL APPROACHES

TOGA, **TOGLL**, and **AugmentTest** do not generate preconditions

GPT4 correctly generates the precondition, but also additional wrong, useless, or non-compilable oracles

Precondition Example

```
/**
 * Sets the item label generator for a series and
 * sends a {@link RendererChangeEvent} to all
 * registered listeners.
 *
 * @param series the series index (zero based).
 * @param generator the generator
 * (null permitted).
 *
 * @see #getSeriesItemLabelGenerator(int)
 */
public void setSeriesItemLabelGenerator(
    int series,
    CategoryItemLabelGenerator generator
) {
    setSeriesItemLabelGenerator(series, generator, true);
}
```

series >= 0

STATE-OF-THE-ART TECHNIQUES

EvoSuite and **Randoop** do not generate preconditions

SYMBOLIC APPROACHES

Jdoctor patterns do not match this text

NEURAL APPROACHES

TOGA, **TOGLL**, and **AugmentTest** do not generate preconditions

GPT4 correctly generates the precondition, but also additional wrong, useless, or non-compilable oracles

generator == null || generator is a valid CategoryItemLabelGenerator instance

Precondition Example

```
/**
 * Sets the item label generator for a series and
 * sends a {@link RendererChangeEvent} to all
 * registered listeners.
 *
 * @param series the series index (zero based).
 * @param generator the generator
 * (null permitted).
 *
 * @see #getSeriesItemLabelGenerator(int)
 */
public void setSeriesItemLabelGenerator(
    int series,
    CategoryItemLabelGenerator generator
) {
    setSeriesItemLabelGenerator(series, generator, true);
}
```

series >= 0

STATE-OF-THE-ART TECHNIQUES

EvoSuite and **Randoop** do not generate preconditions

SYMBOLIC APPROACHES

Jdoctor patterns do not match this text

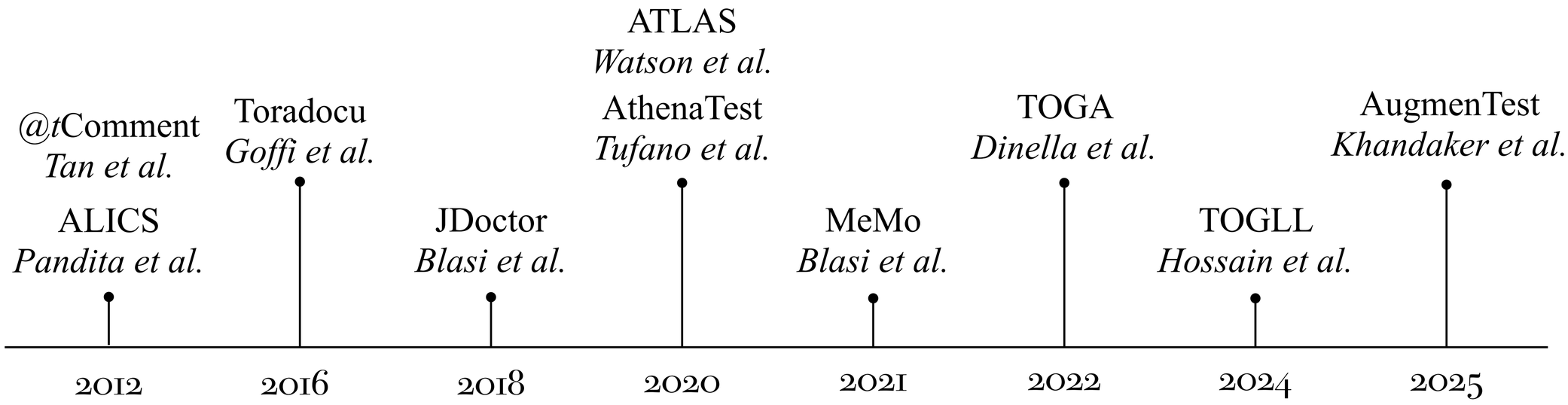
NEURAL APPROACHES

TOGA, **TOGLL**, and **AugmentTest** do not generate preconditions

GPT4 correctly generates the precondition, but also additional wrong, useless, or non-compilable oracles

TraTTO generates the precondition and does not generate wrong oracles like GPT4

Neuro & Symbolic Approaches



Symbolic Approaches

Technique

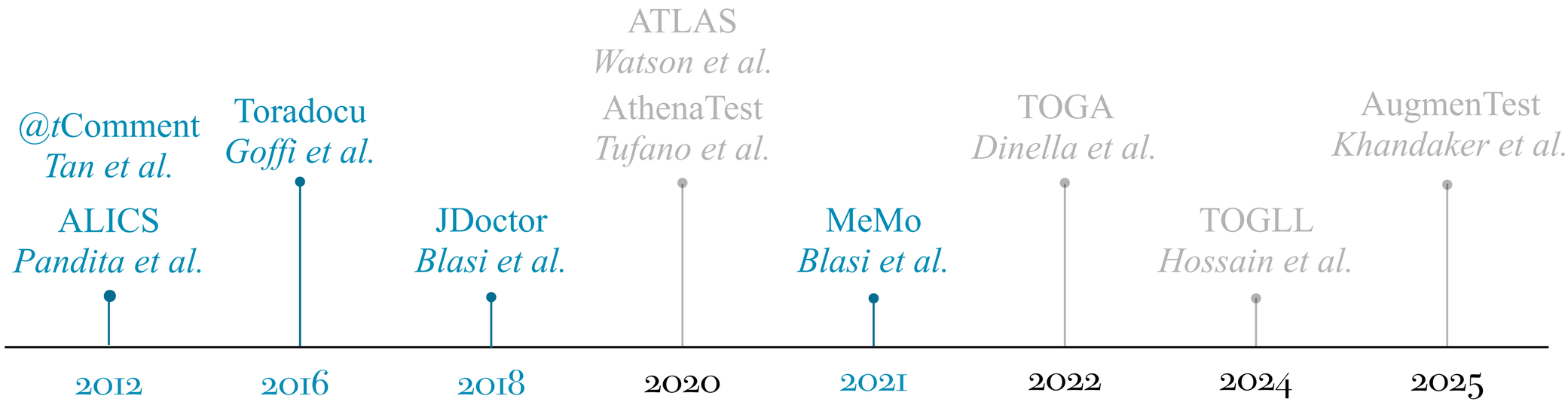
Fixed set of domain-specific rules, such as pattern and lexical matching

Pros

High precision and recall

Cons

Restricted applicability



Neural Approaches

Technique

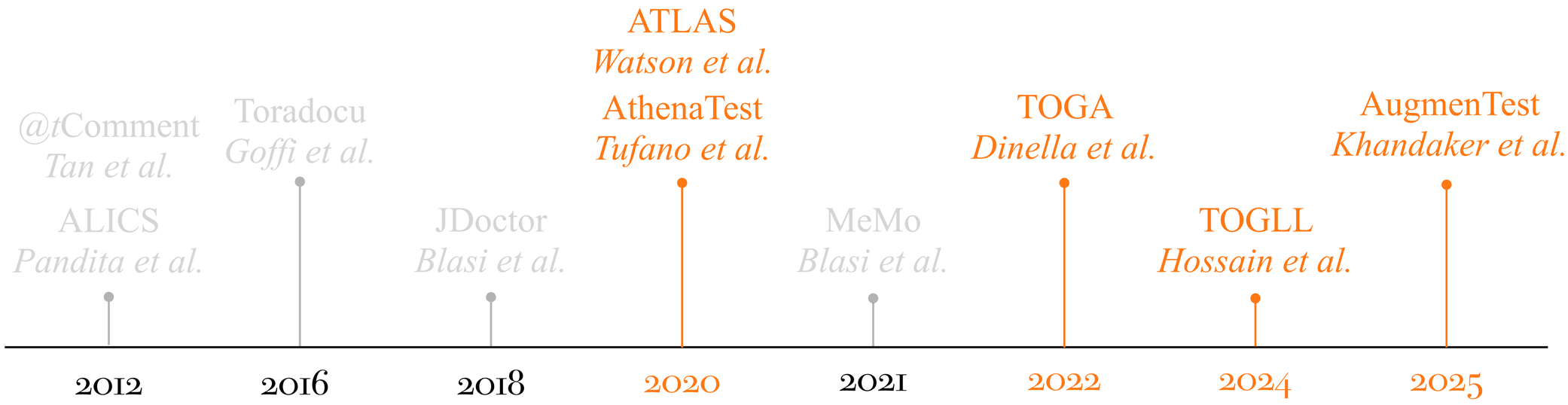
Deep Learning and transfer learning

Pros

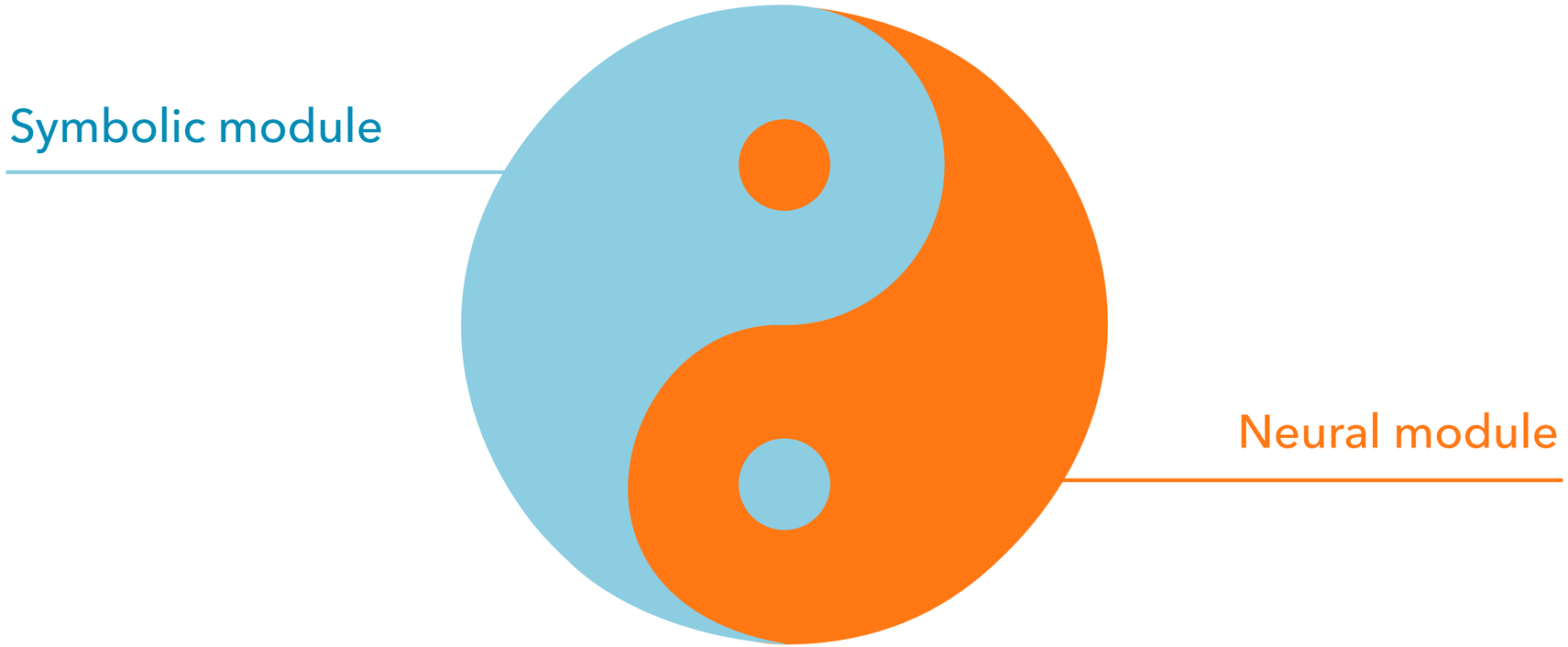
Well suited to handle informal text

Cons

High false positive rate



Our Approach: Neuro-Symbolic



NEURO-SYMBOLIC APPROACHES

TraTTO

TRANSFORMER-BASED **T**OKEN BY **T**OKEN **O**RACLE GENERATION

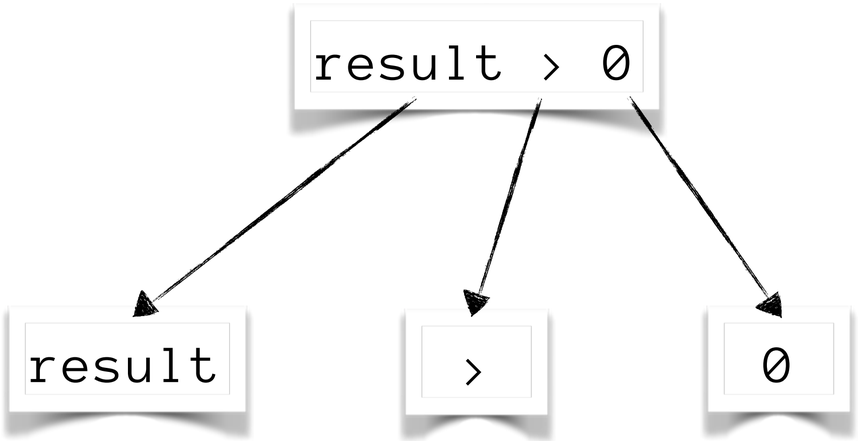
Reformulates the oracle generation problem as a token generation problem

```
result > 0
```

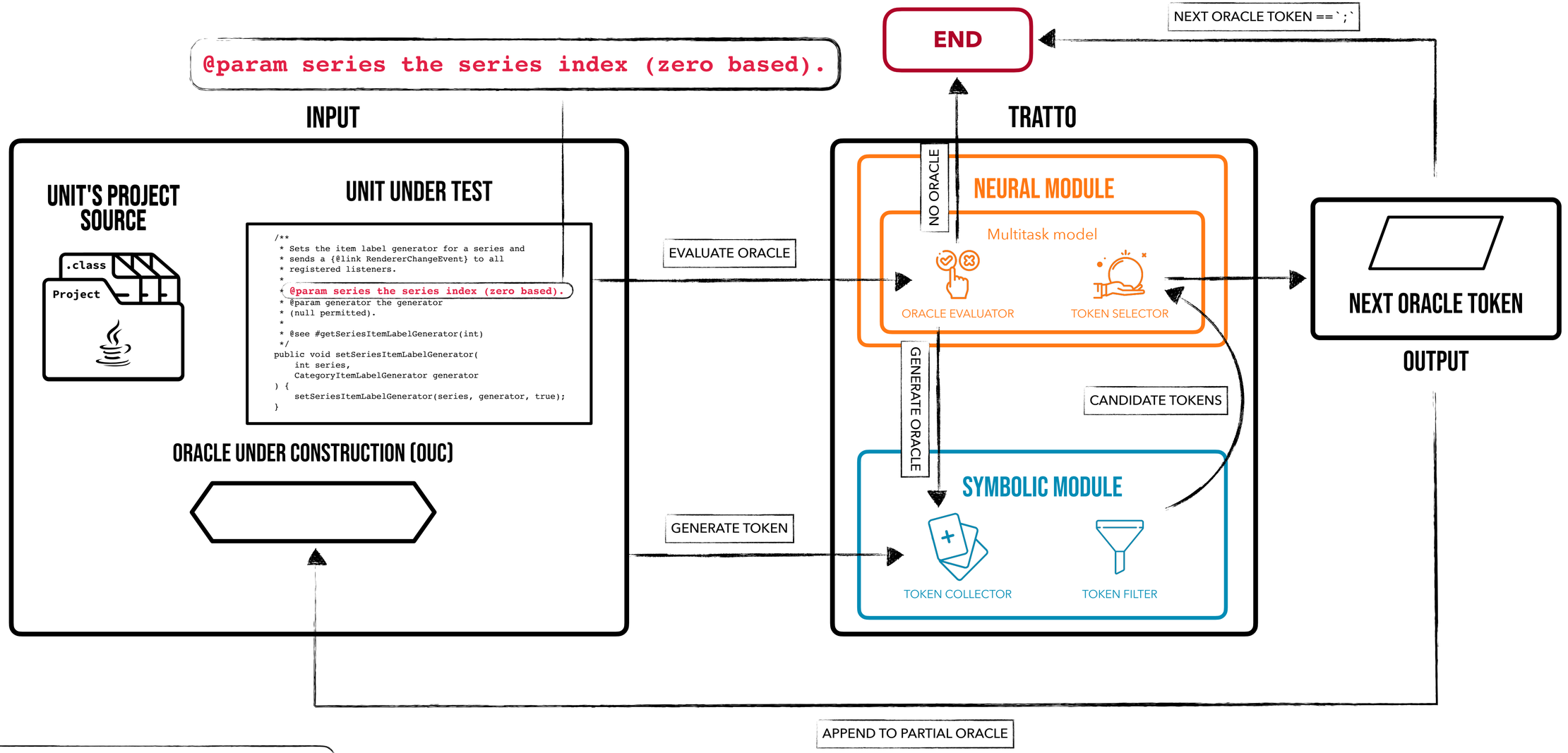
TraTTO

TRANSFORMER-BASED **T**OKEN BY **T**OKEN **O**RACLE GENERATION

Reformulates the oracle generation problem as a token generation problem

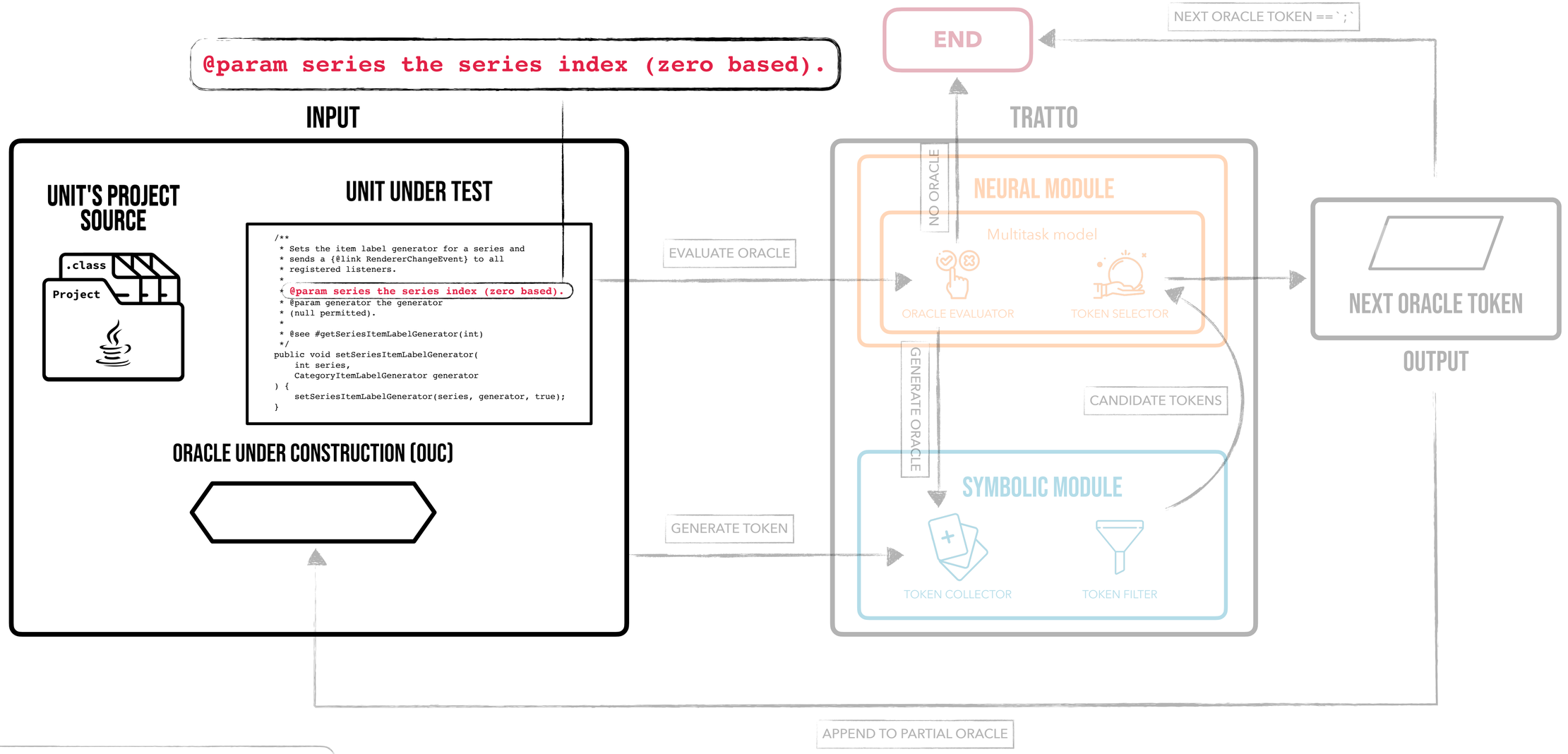


Workflow



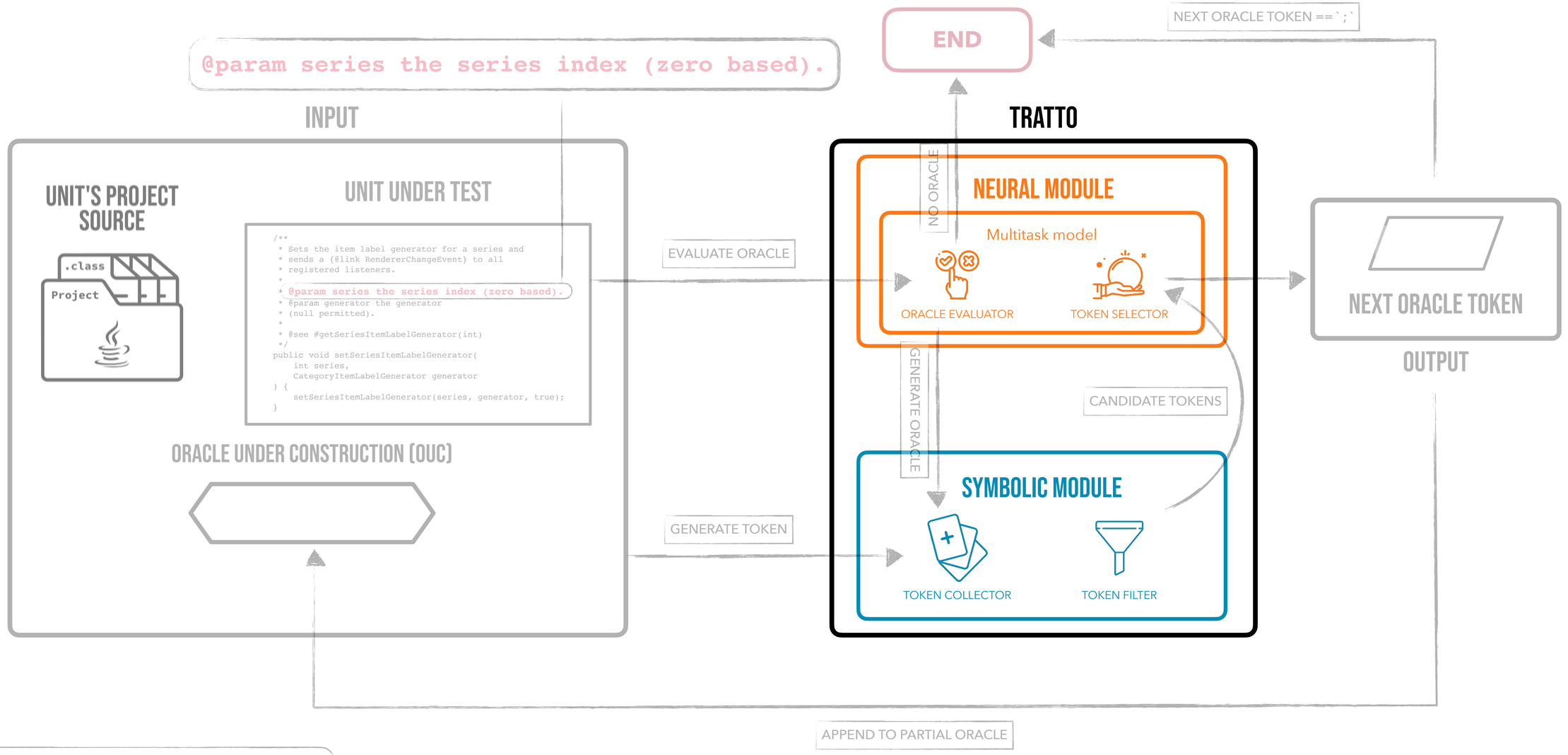
Final oracle: `series >= 0`

Input



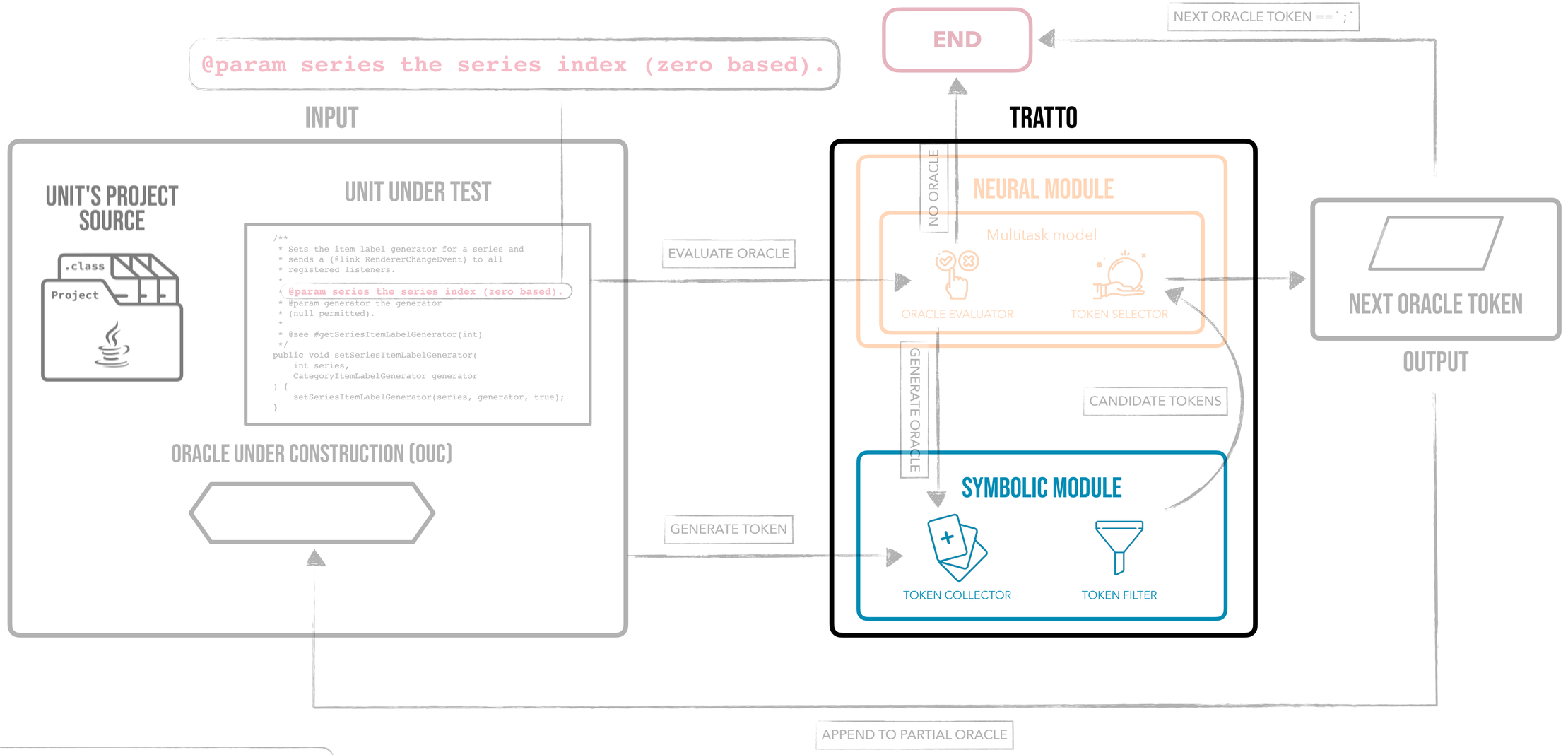
Final oracle: `series >= 0`

TraTTO



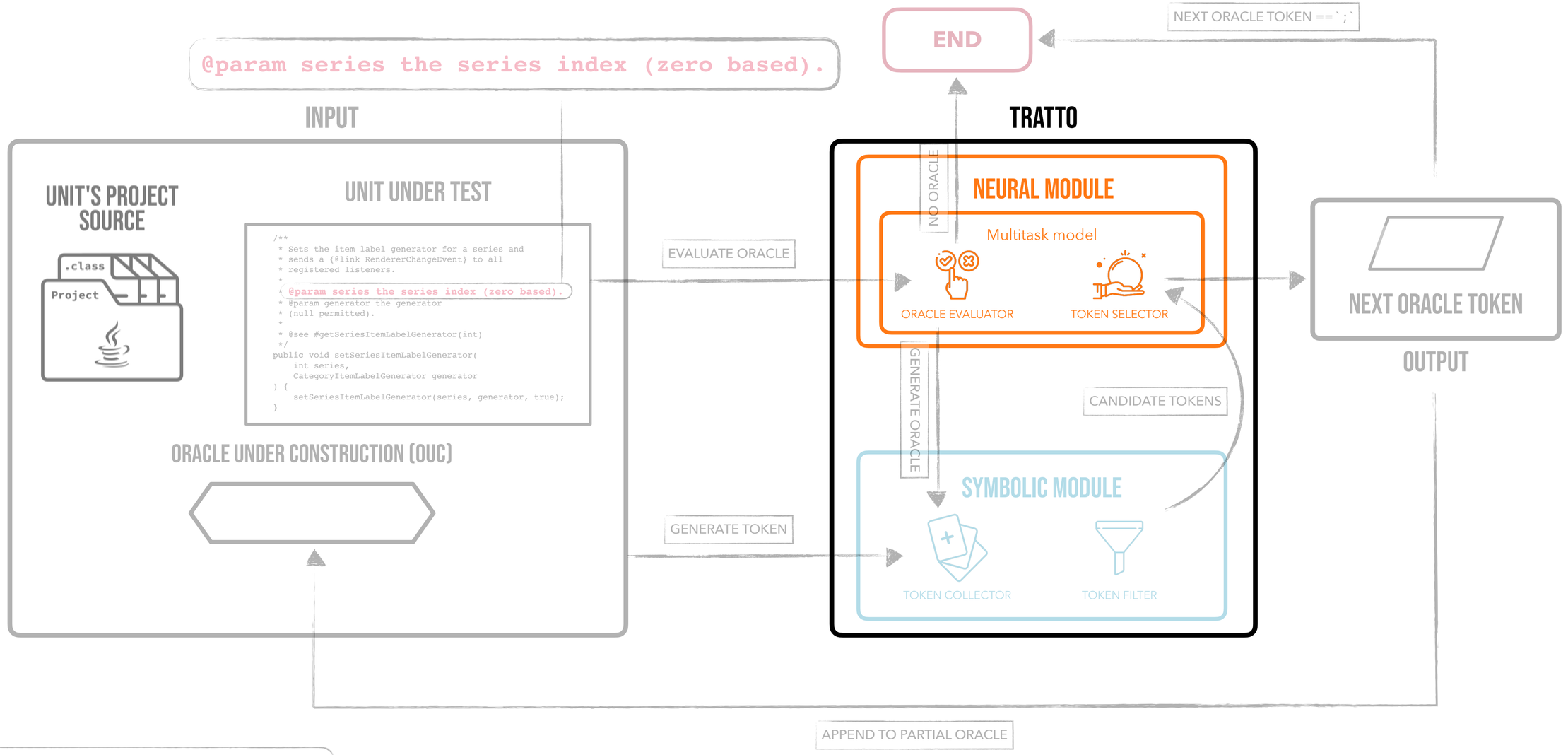
Final oracle: `series >= 0`

Symbolic Module



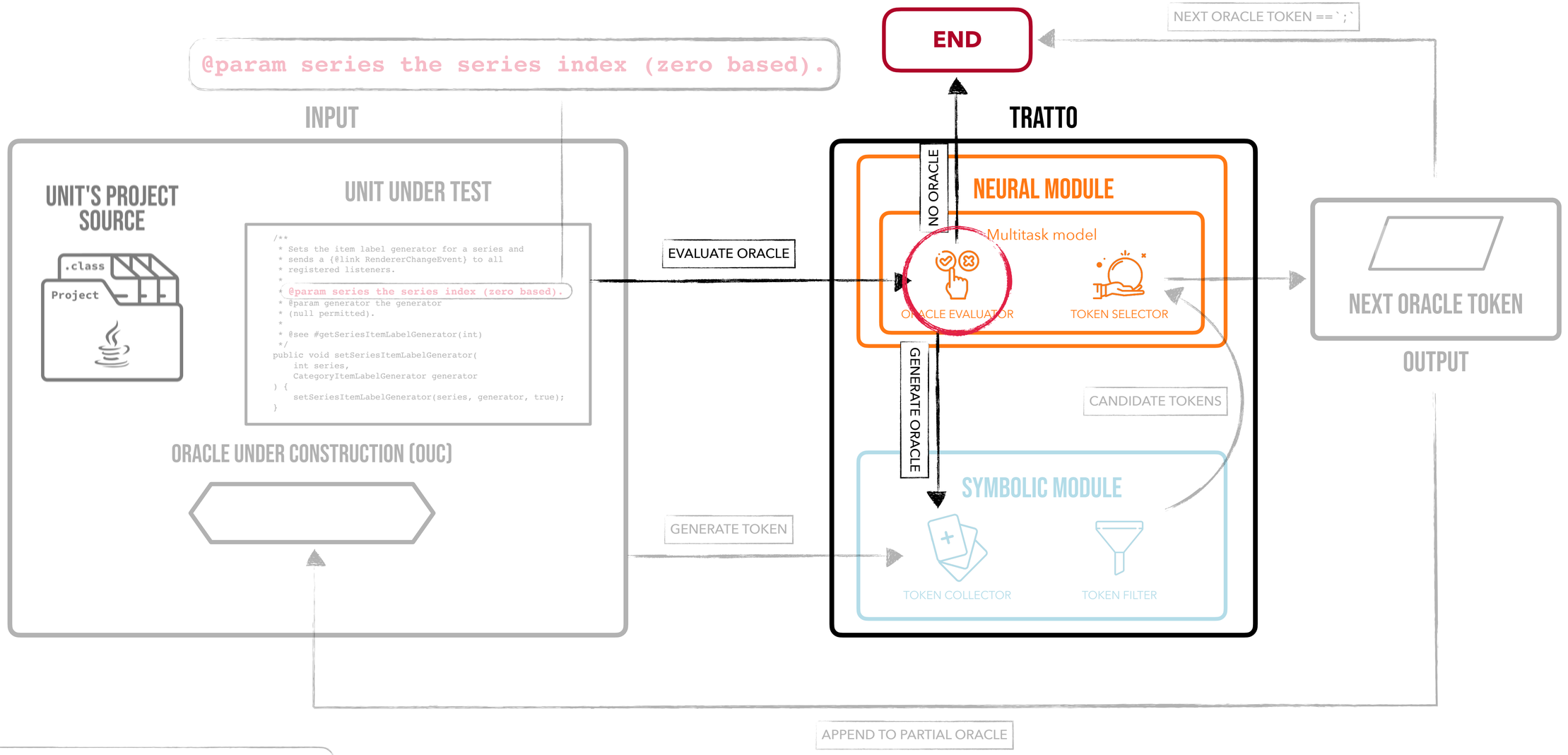
Final oracle: `series >= 0`

Neural Module



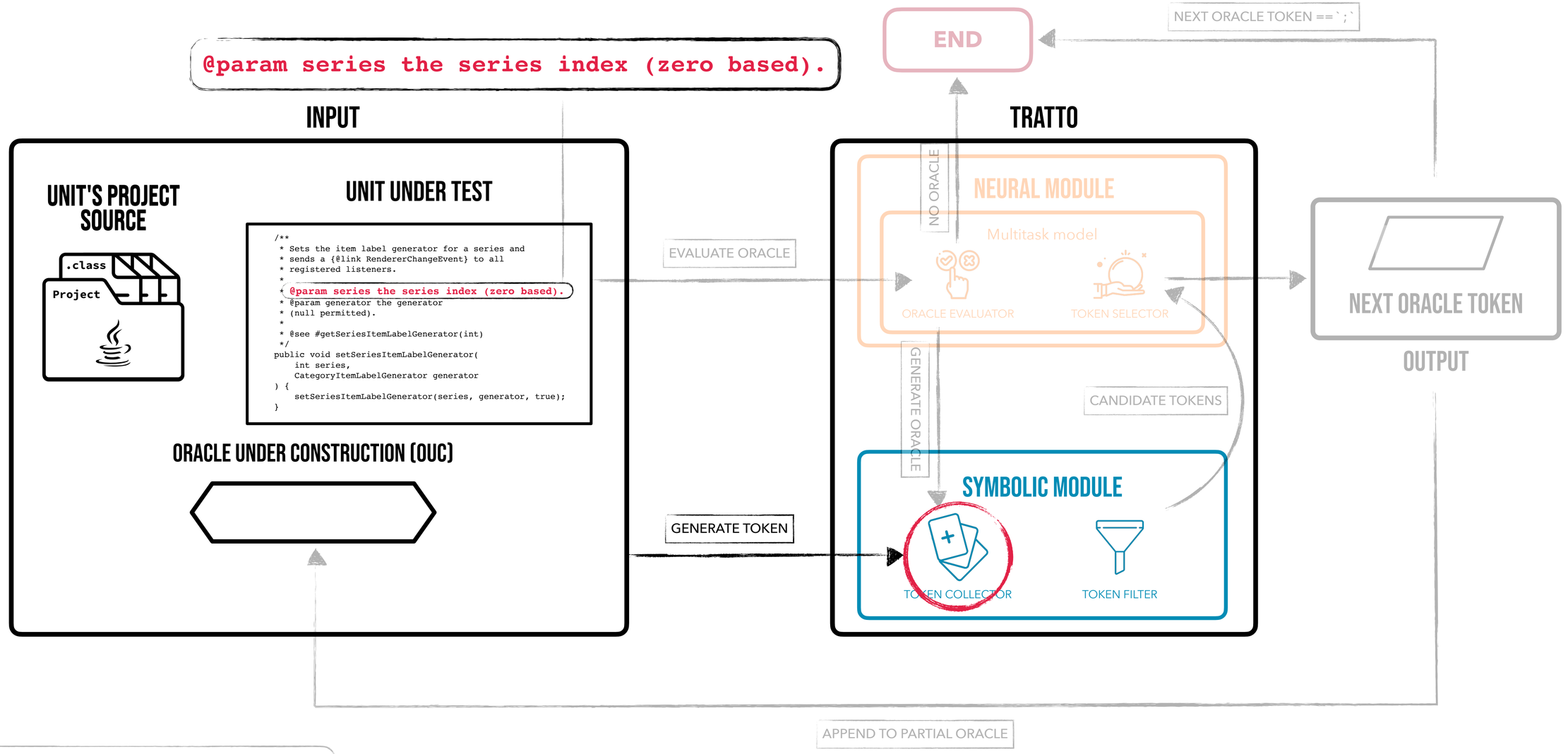
Final oracle: `series >= 0`

Should an oracle be generated?



Final oracle: `series >= 0`

Token Collector



Final oracle: `series >= 0`

Token Collector

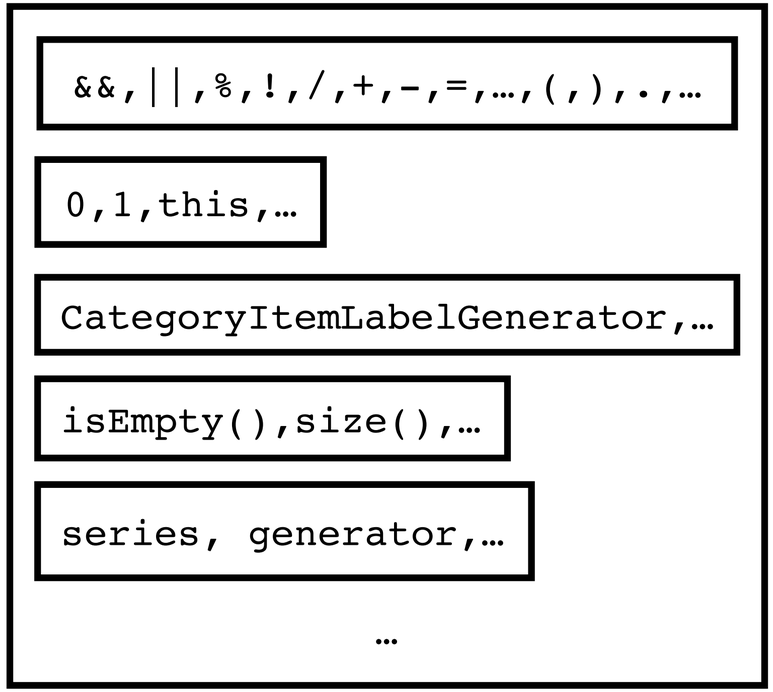
OPERATORS

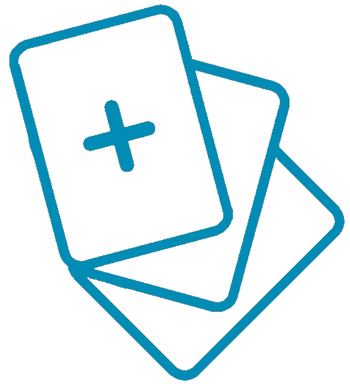
COMMON CONSTANTS AND VARIABLES

PROJECT CLASS NAMES

CLASS METHODS AND FIELDS

PARAMETER NAMES

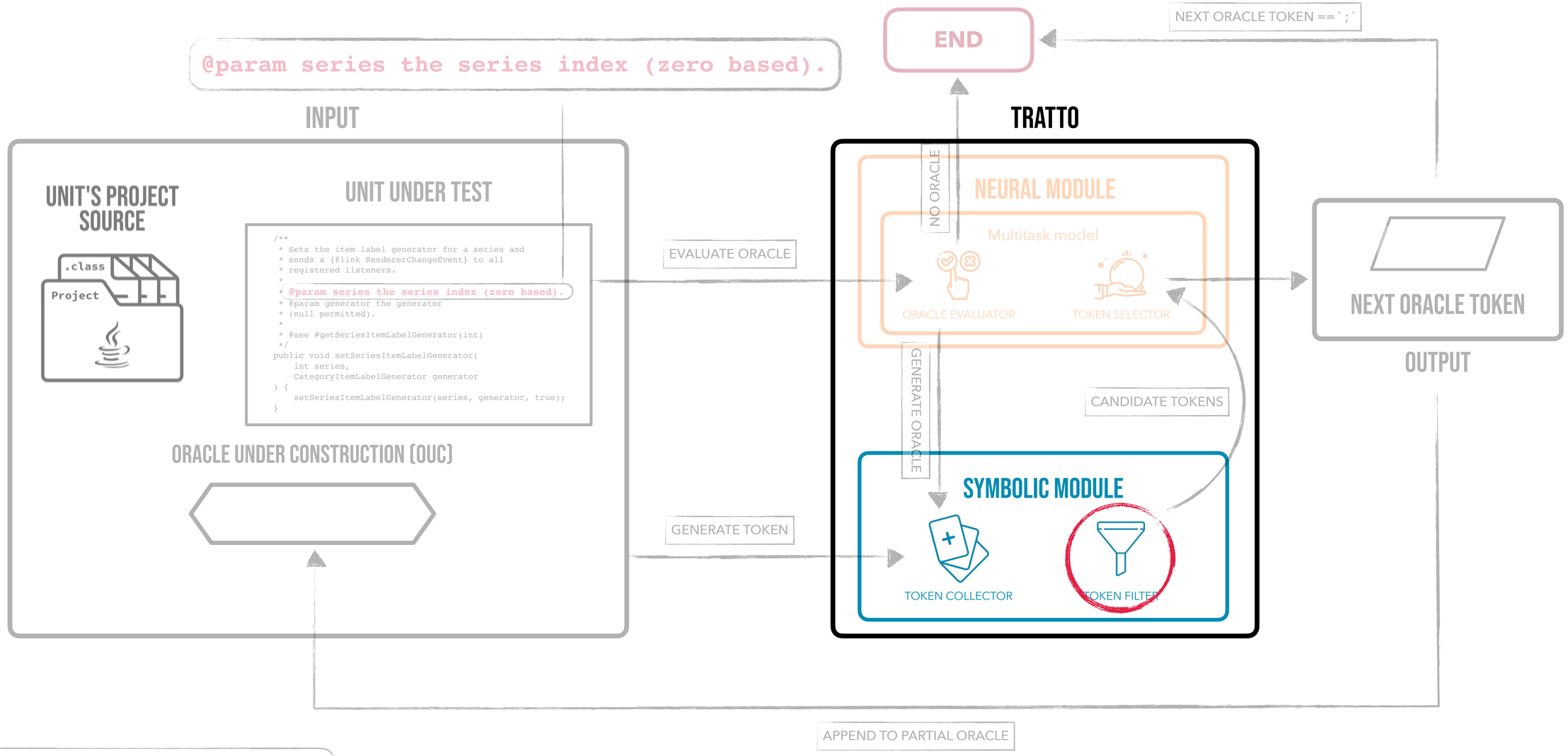




TOKEN COLLECTOR

Symbolic Module

Token Filter



Final oracle: `series >= 0`

Token Filter

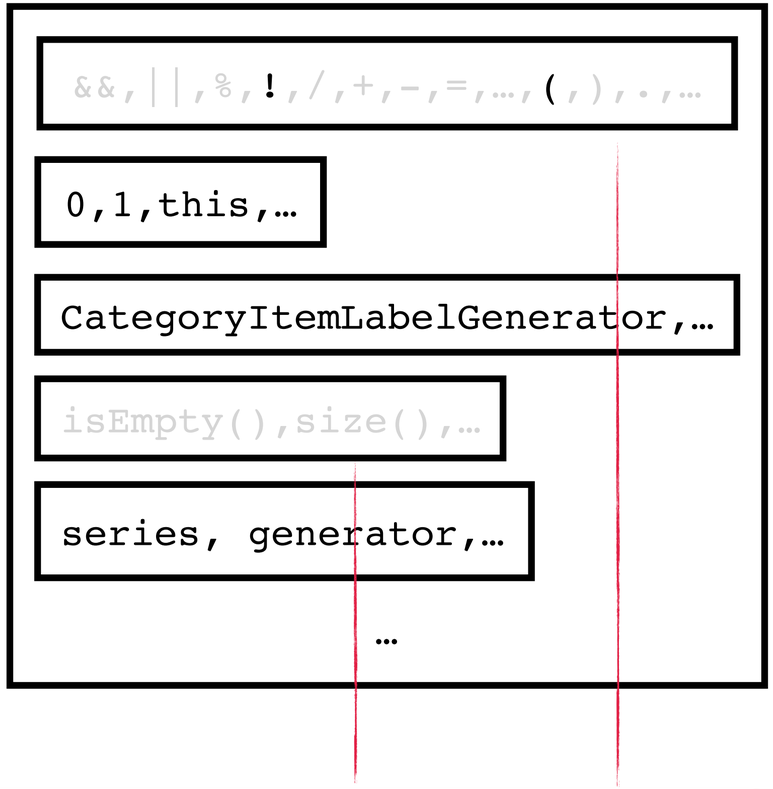
OPERATORS

COMMON CONSTANTS AND VARIABLES

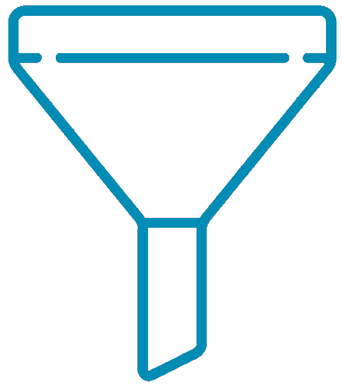
PROJECT CLASS NAMES

CLASS METHODS AND FIELDS

PARAMETER NAMES



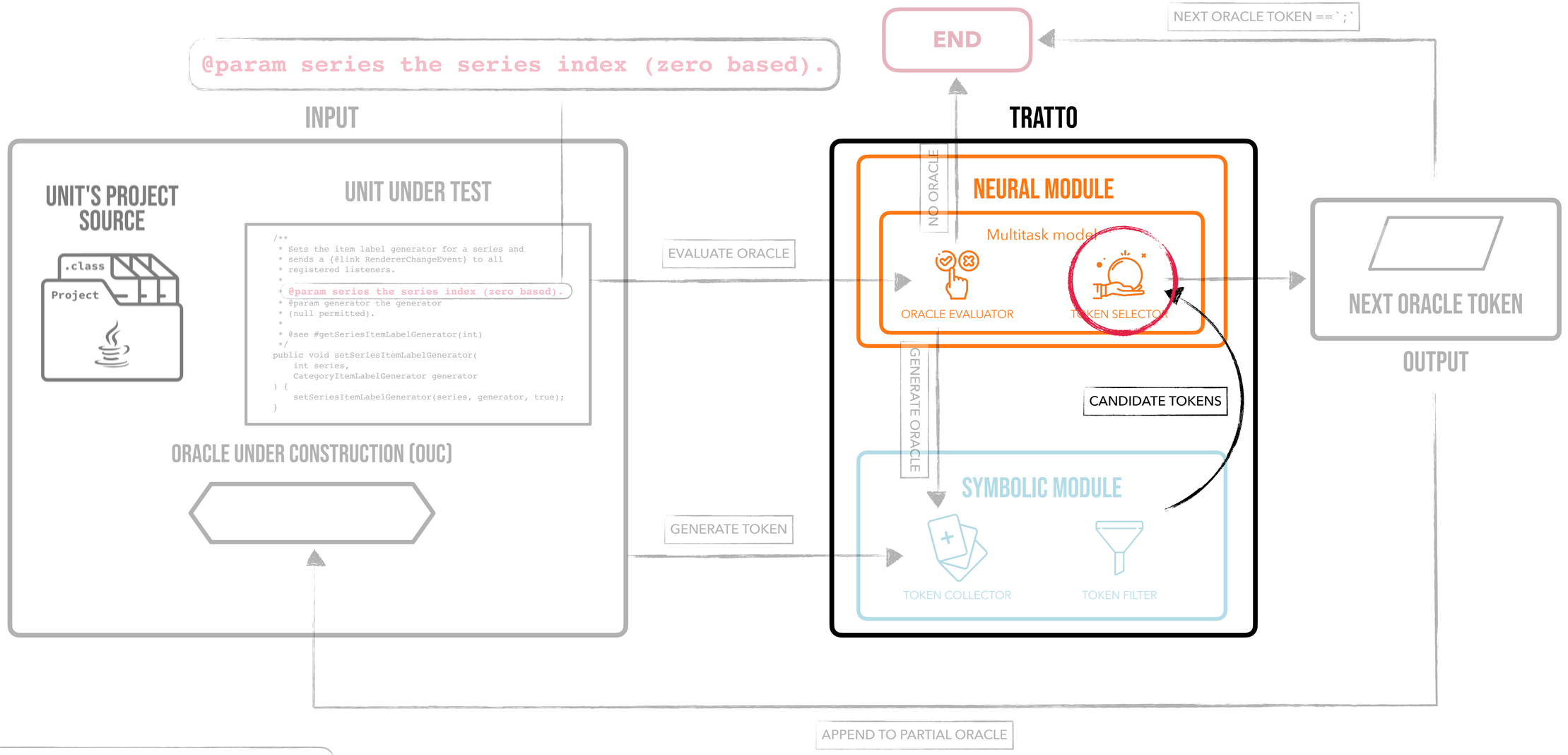
Rule out invalid tokens, given the current partial oracle



TOKEN FILTER

Symbolic Module

Token Selector



Final oracle: `series >= 0`

Token Selector

INPUT

```
// PRE-CONDITION: "@param series the series index (zero based)."
// Next possible tokens: ['!','(','0','1','this','series','generator',…]
// Assertion:
assertTrue(<FILL_ME>

// Method under test:
/**
 * Sets the item label generator for a series and
 * sends a {@link RendererChangeEvent} to all
 * registered listeners.
 * …
 */
public void setSeriesItemLabelGenerator(
    int series,
    CategoryItemLabelGenerator generator
) {…}

// Additional context
public int size() { … }
public int isEmpty() { … }
…
```

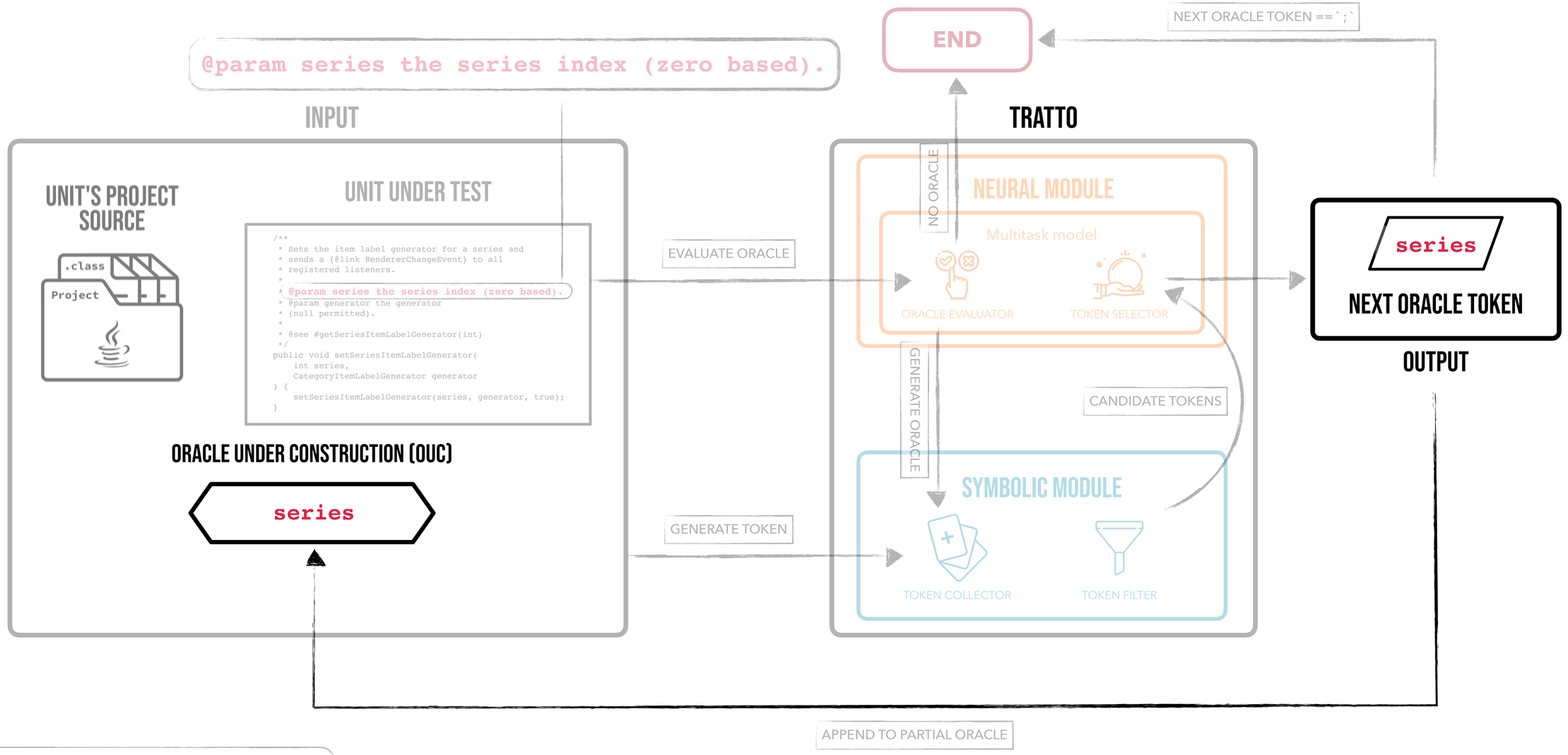
OUTPUT

series

TOKEN SELECTOR

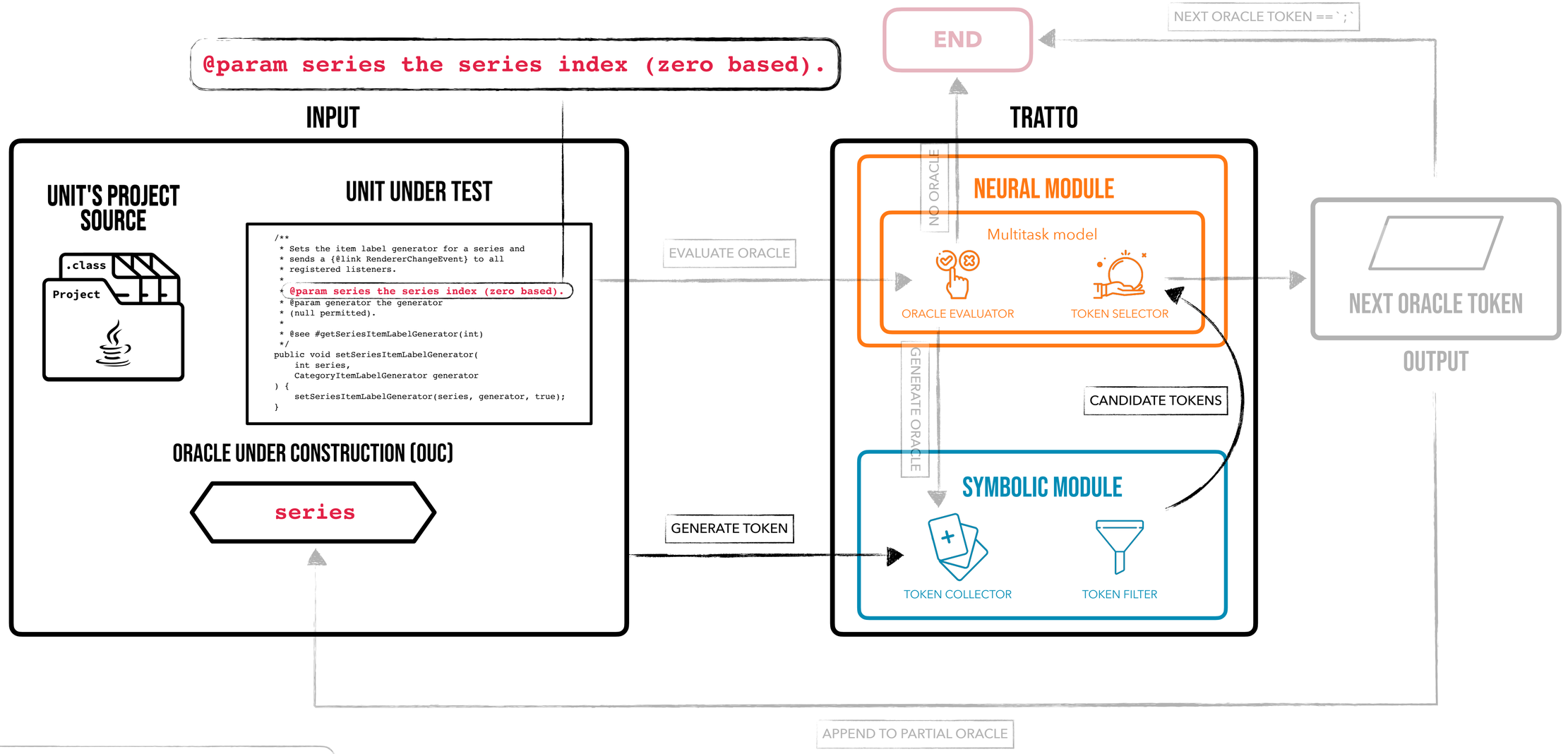
Neural Module

Token Generation



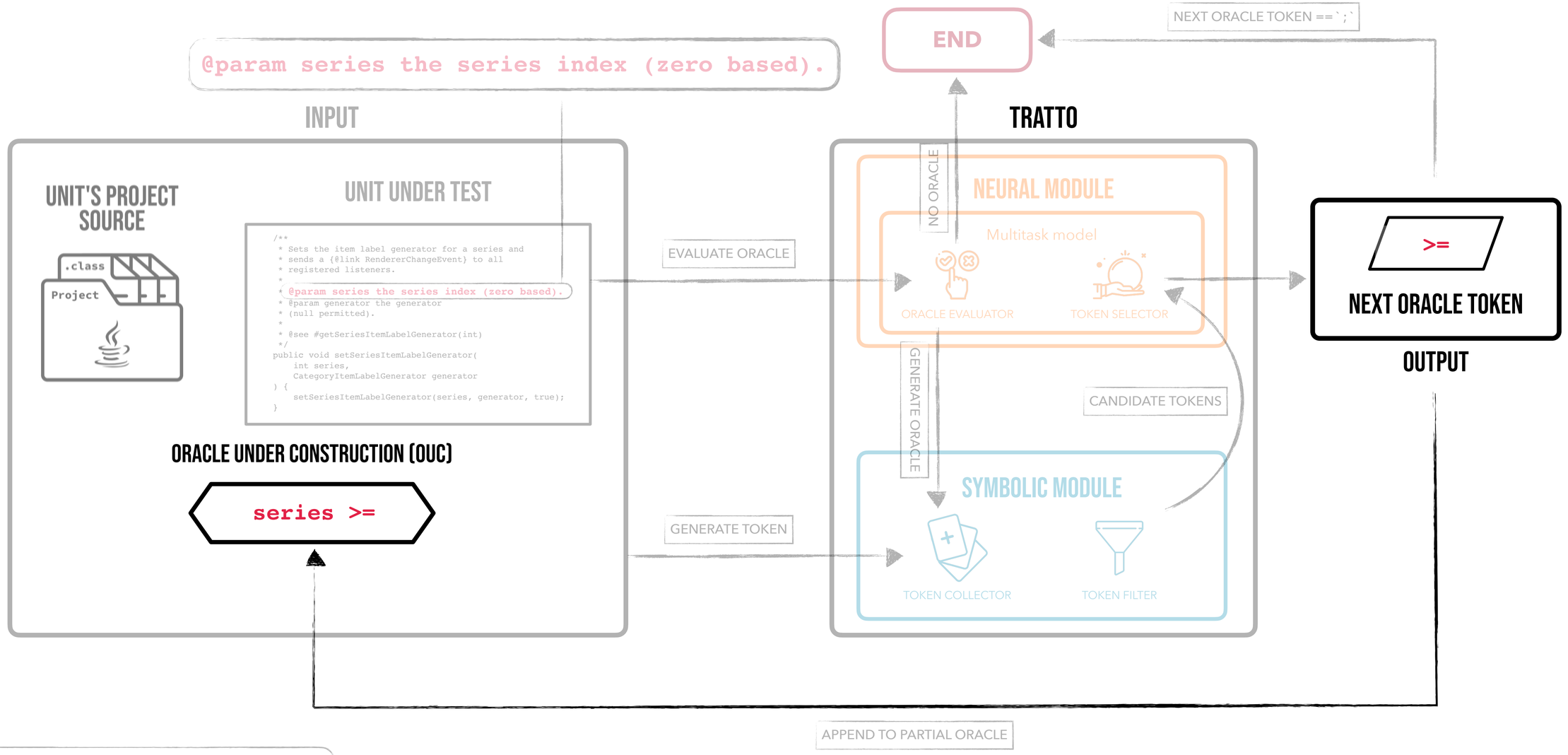
Final oracle: `series >= 0`

2° Iteration



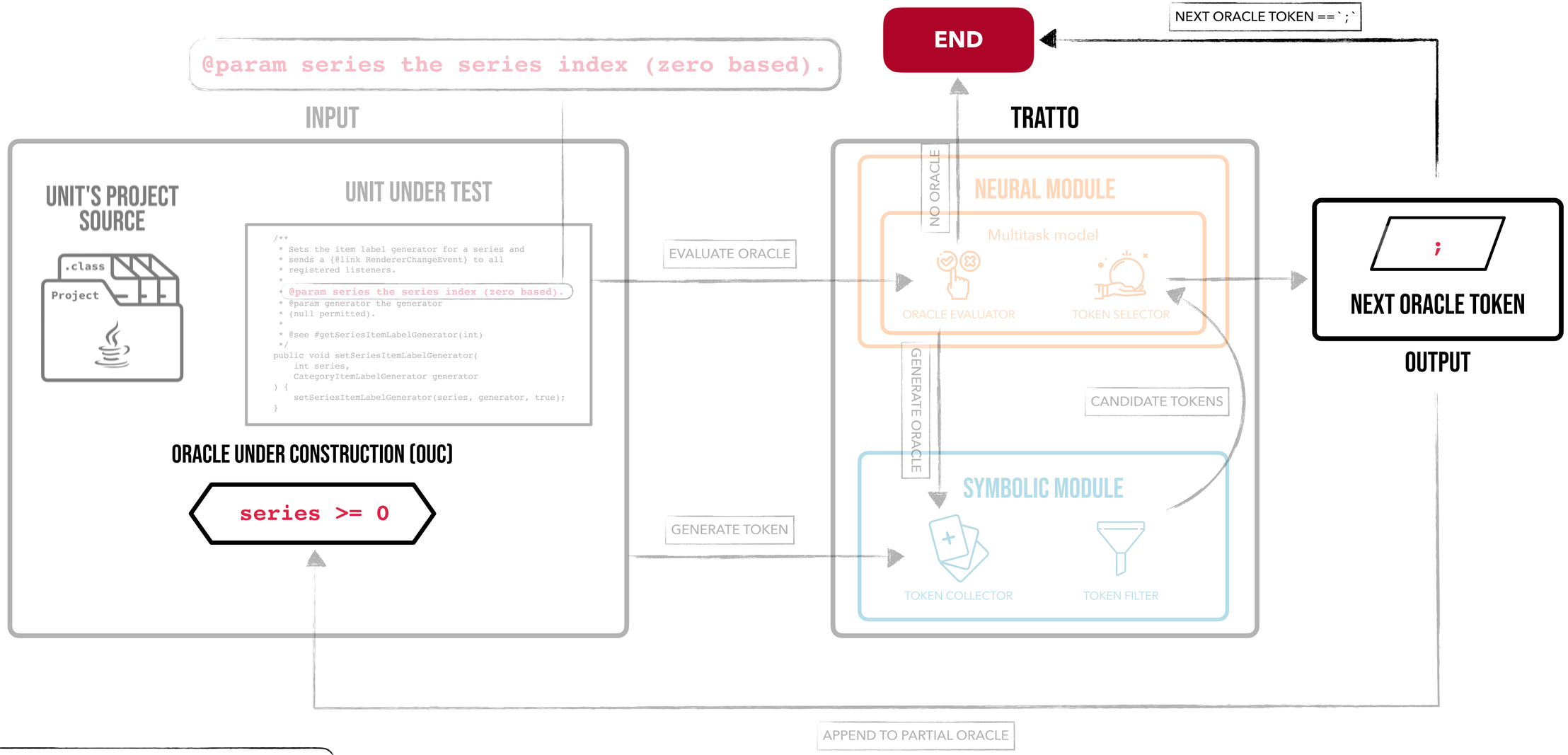
Final oracle: `series >= 0`

2° Iteration



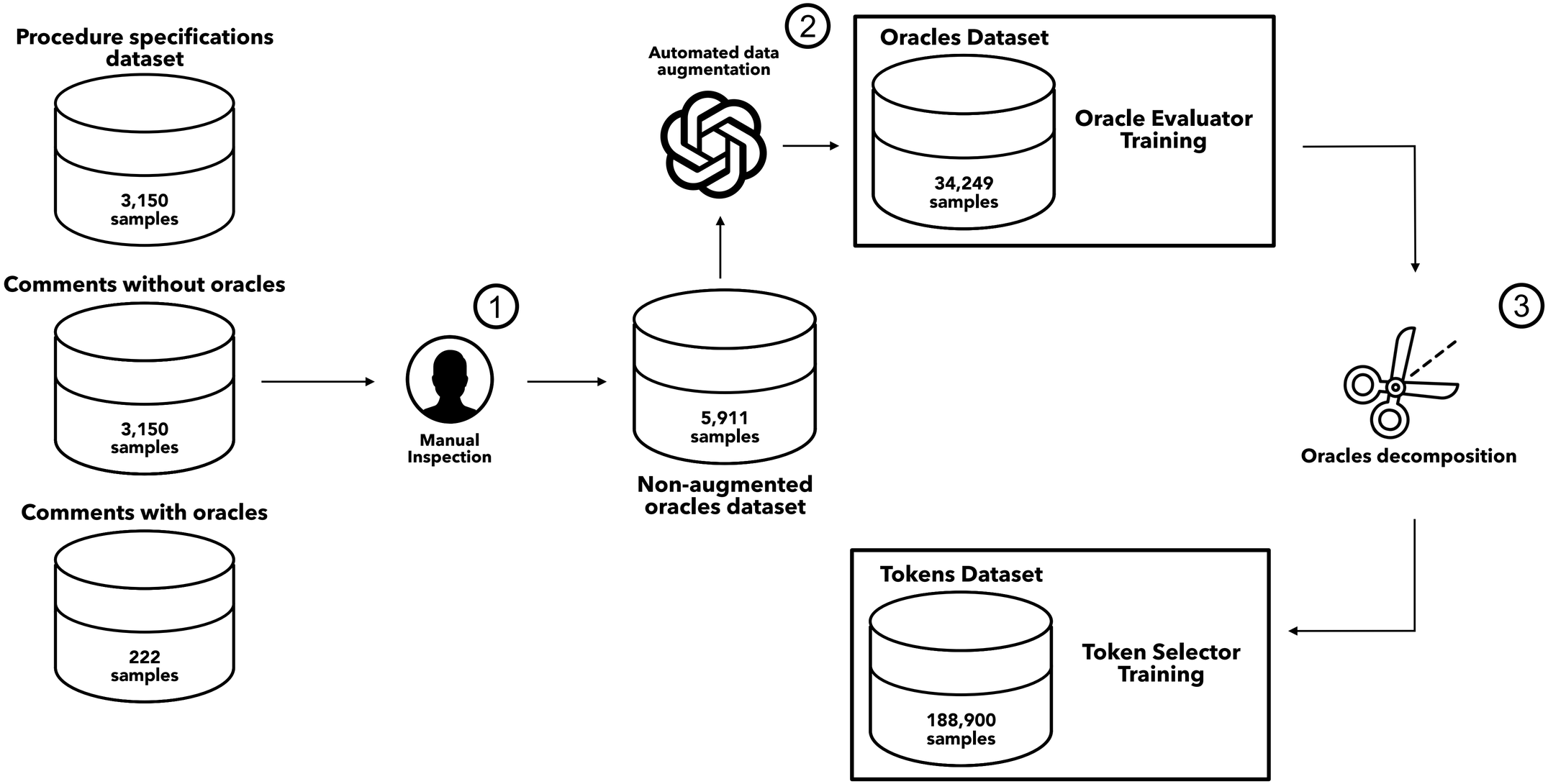
Final oracle: `series >= 0`

Final Oracle

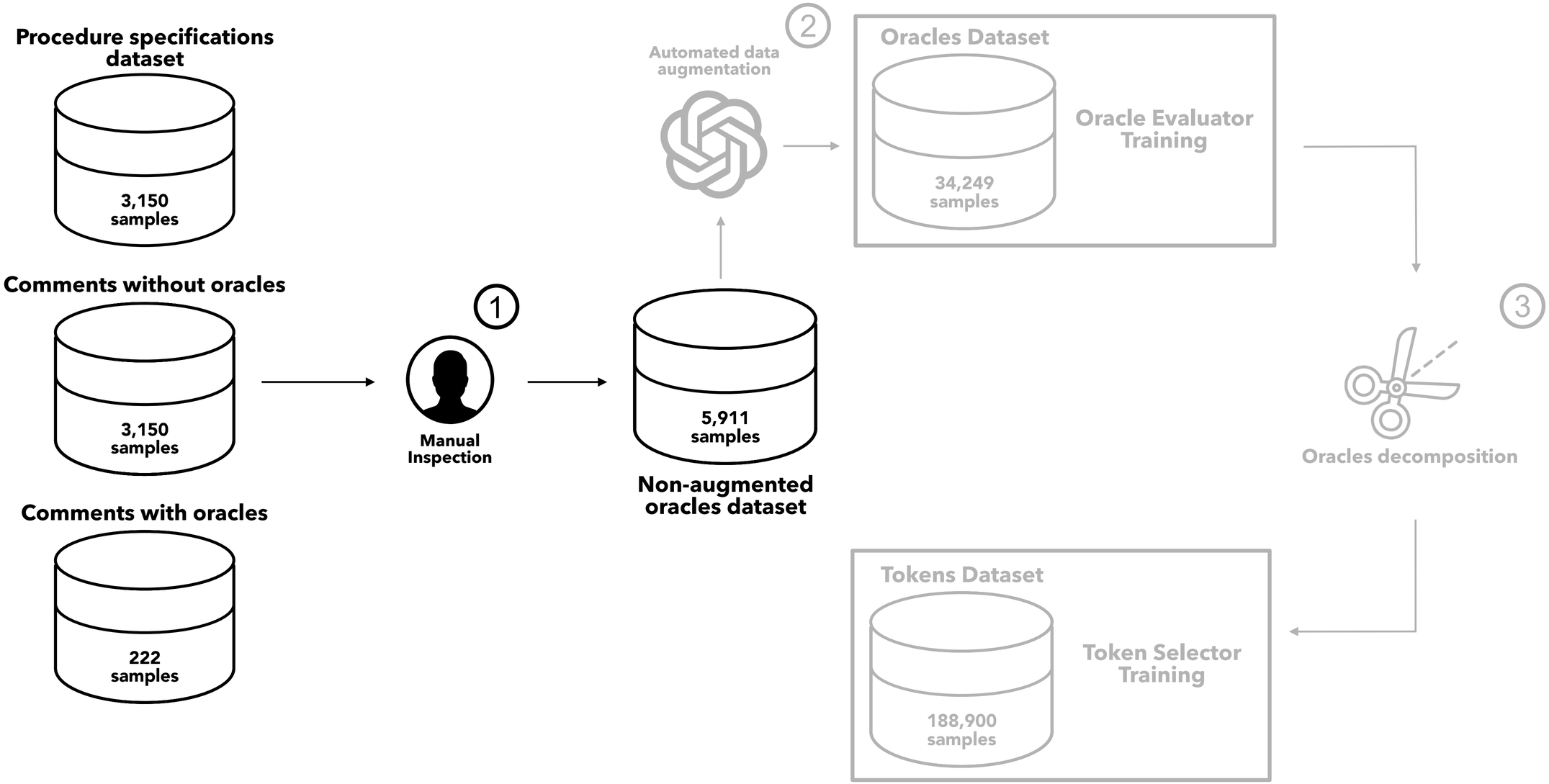


Final oracle: `series >= 0`

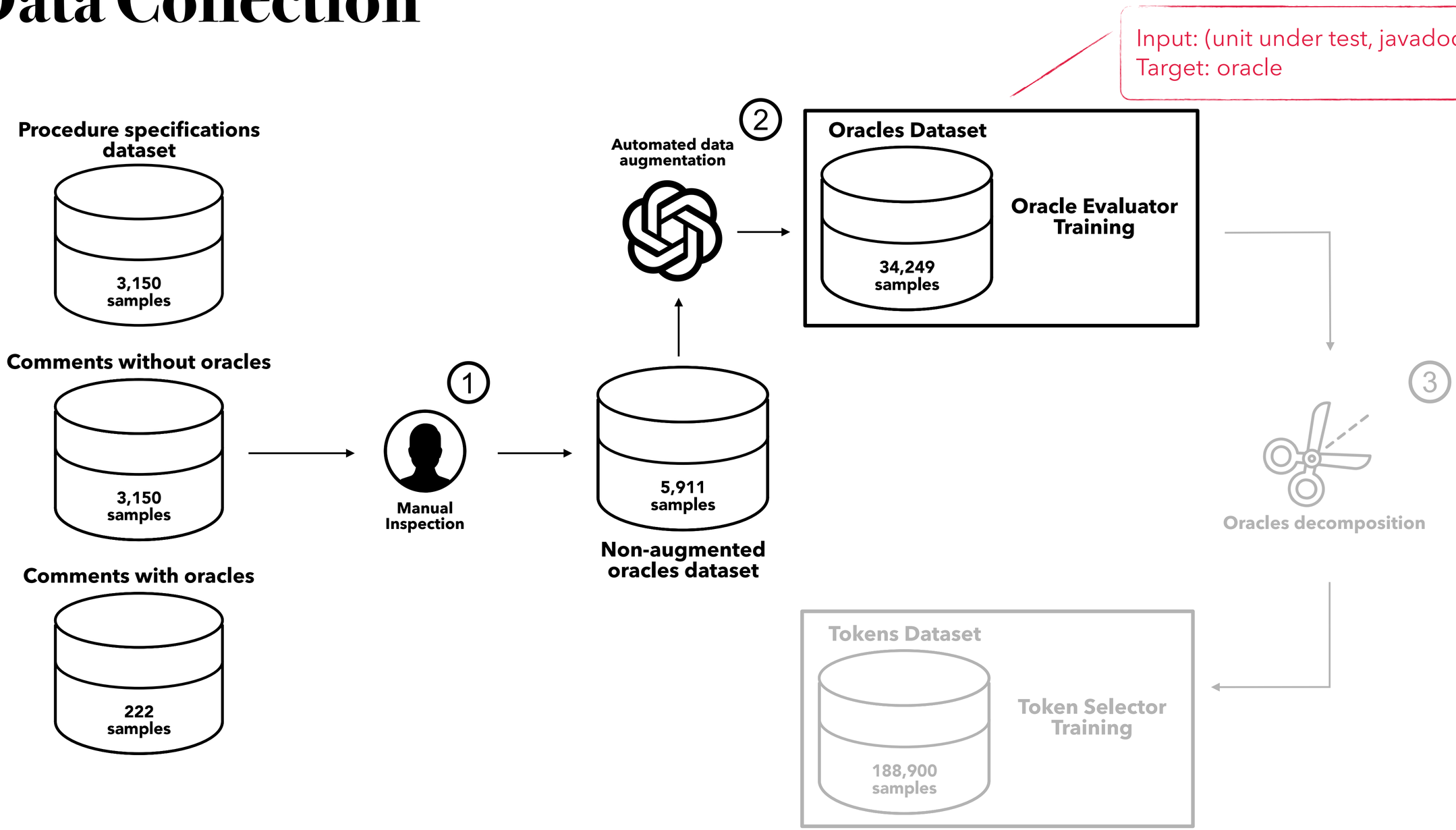
Data Collection



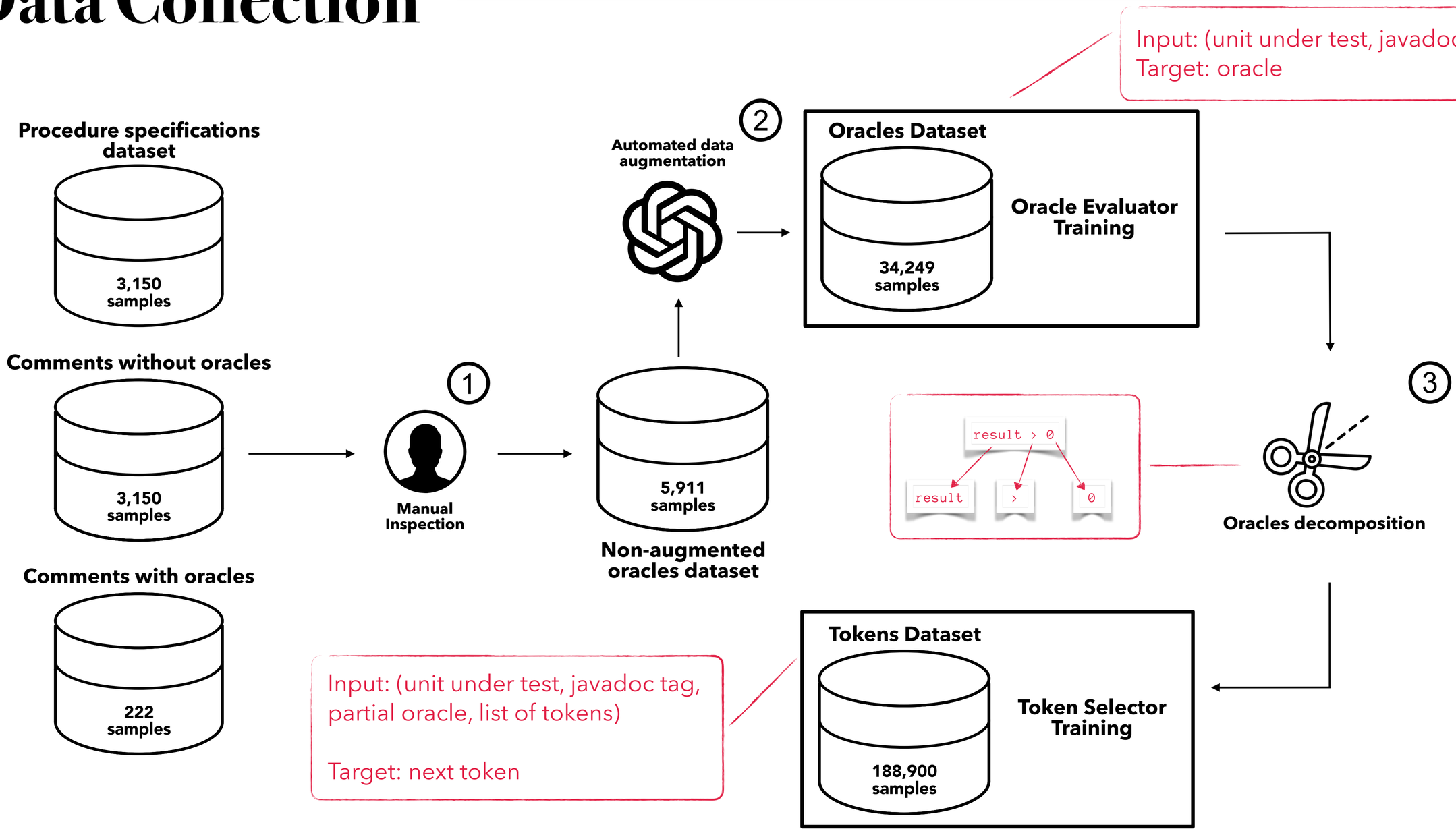
Data Collection



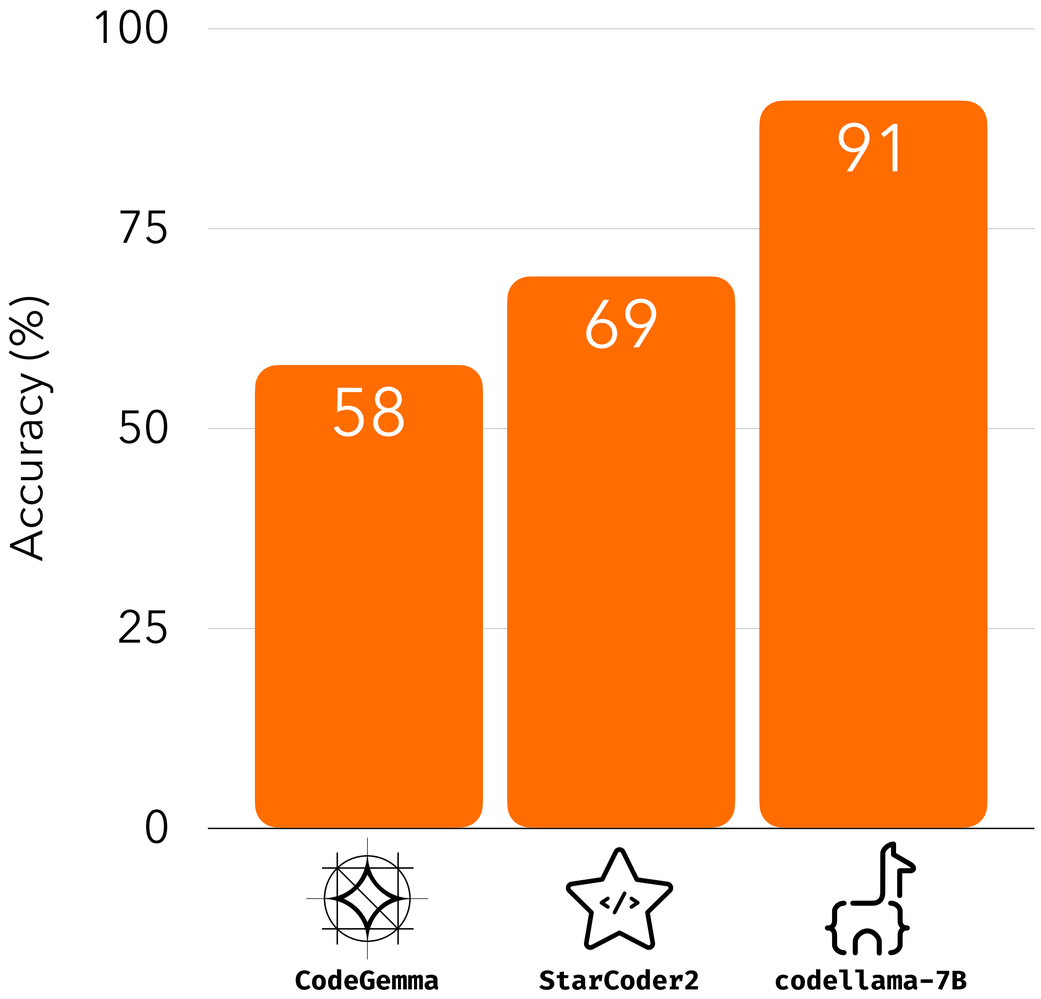
Data Collection



Data Collection



Code Models Comparison



RQ1

Compare code models capability to evaluate oracles and select tokens

Dataset

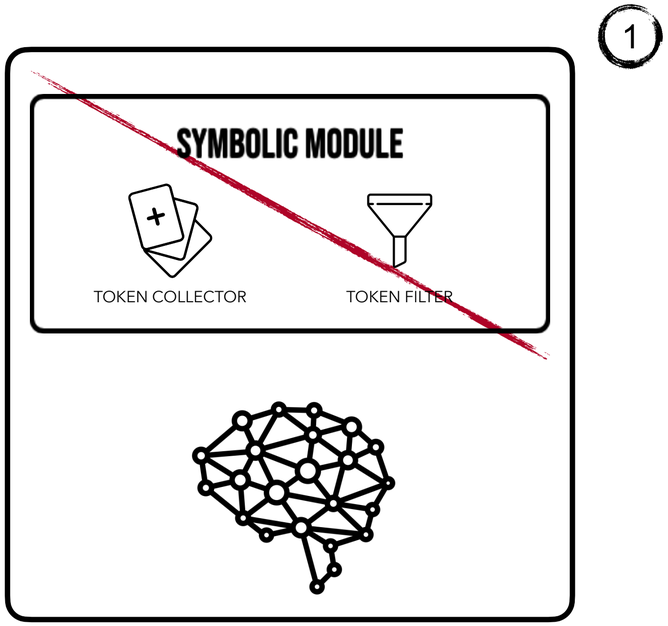
223,149 samples

- **34,249** oracles samples
- **188,900** tokens samples

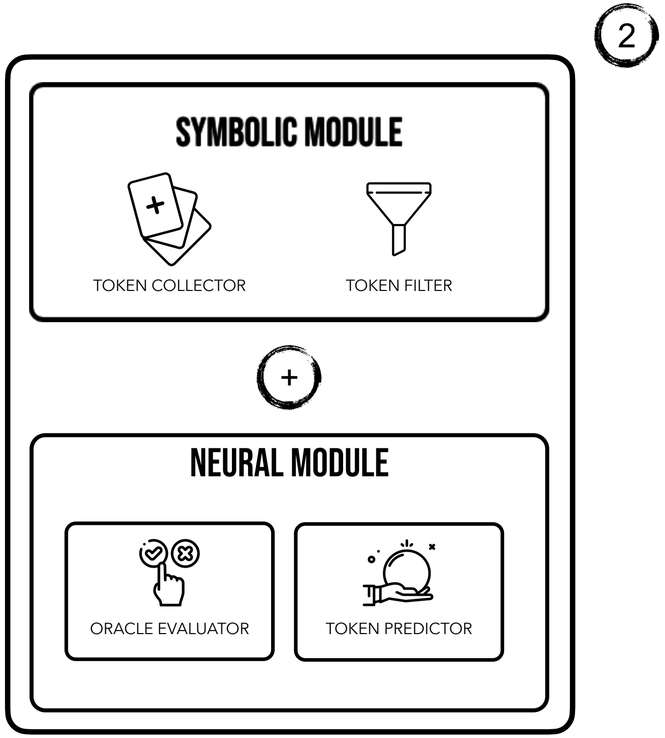
192,258 samples for training (**86%**)

30,891 samples for validation (**14%**)

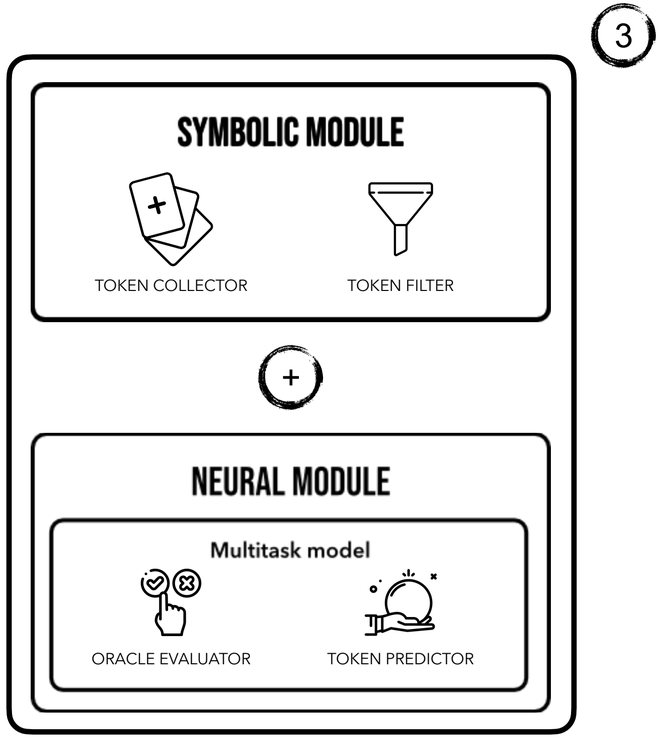
Ablation Studies



NON-SYMBOLIC

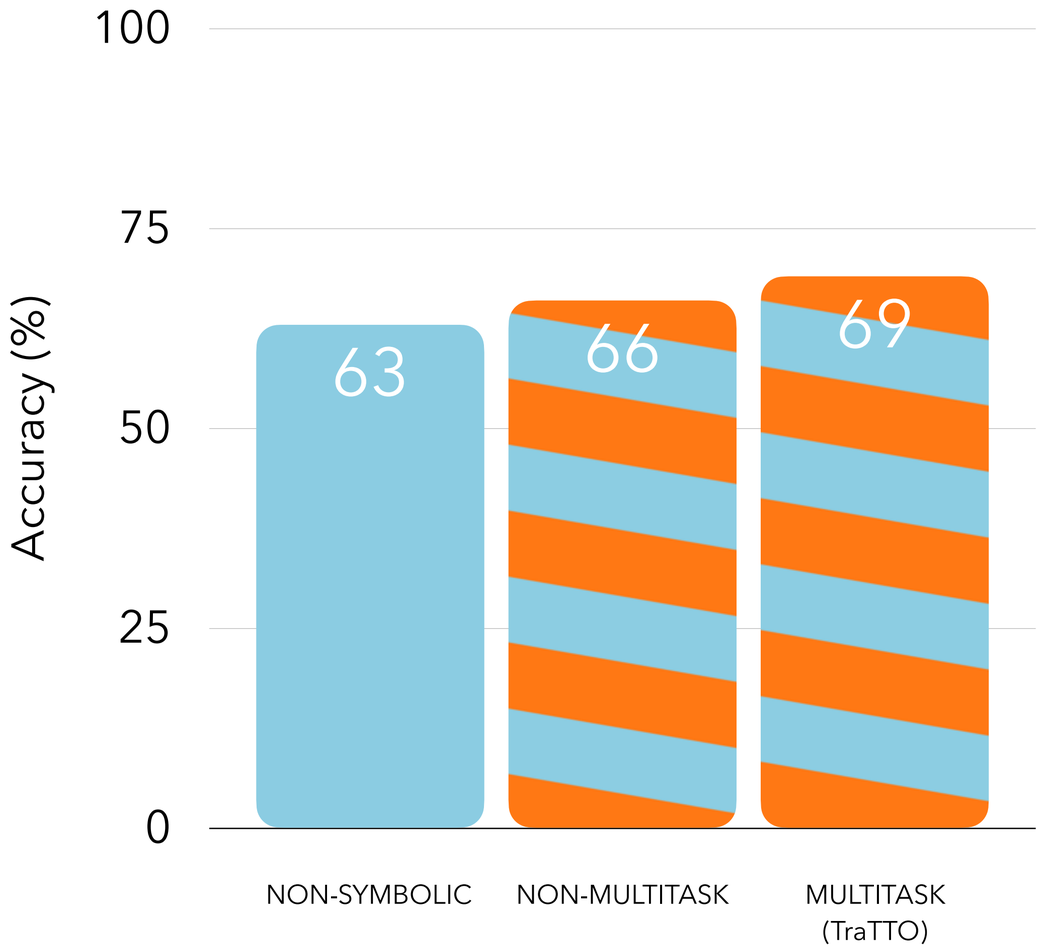


NON-MULTITASK



MULTITASK
(TraTTO)

Ablation Studies



RQ2

Evaluate the contributions of the symbolic module and multitask model of Tratto in generating oracles

McNemar Statistical Test

NON-SYMBOLIC MULTITASK	$Odds\text{-}Ratio = 2.57,\ p\text{-}value < 0.001$
NON-MULTITASK MULTITASK	$Odds\text{-}Ratio = 1.55,\ p\text{-}value < 0.001$

both statistically significant

Oracle Generation

GROUND
TRUTH DATSET

NEURO-SYMBOLIC

TRATTO

SYMBOLIC

JDOCTOR

NEURAL

GPT4

ACCURACY
PRECISION
RECALL
F1-SCORE

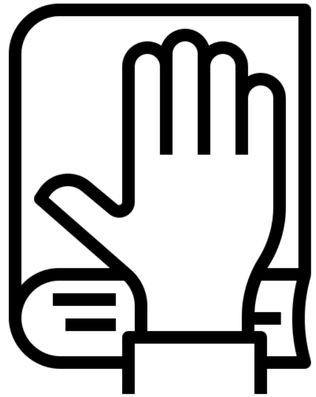
Oracle Generation

15 Defects4J Projects

885 datapoints of axiomatic oracles

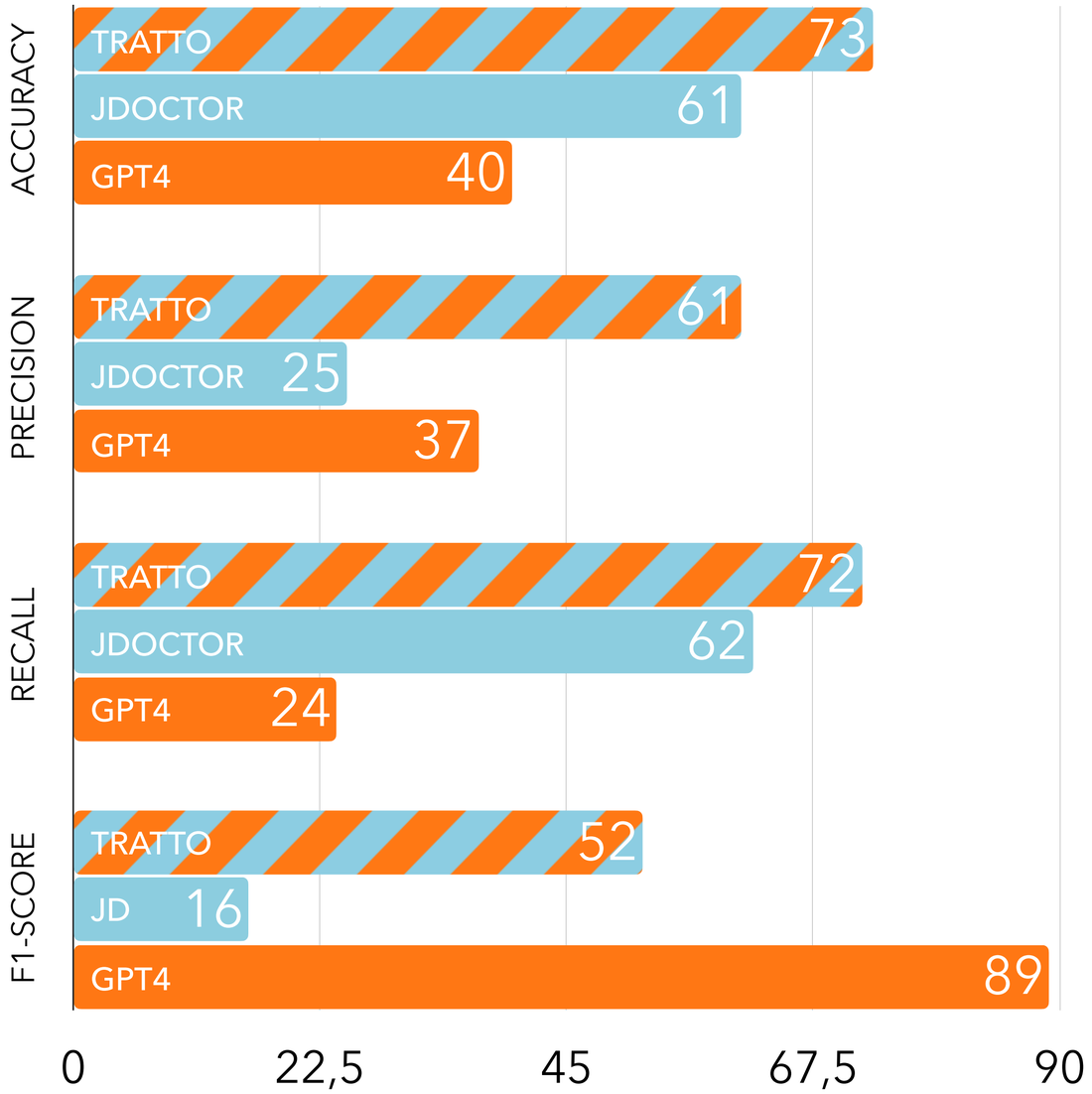
- 389 positive examples
- 496 negative examples

manually extracted from 274 methods



GROUND
TRUTH DATSET

Oracle Generation



RQ3

Compares the effectiveness of TraTTO to the state-of-the-art neural and symbolic approaches

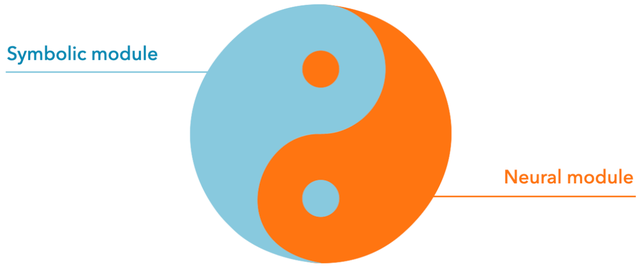
3 times

more correct oracles than Jdoctor

10 times

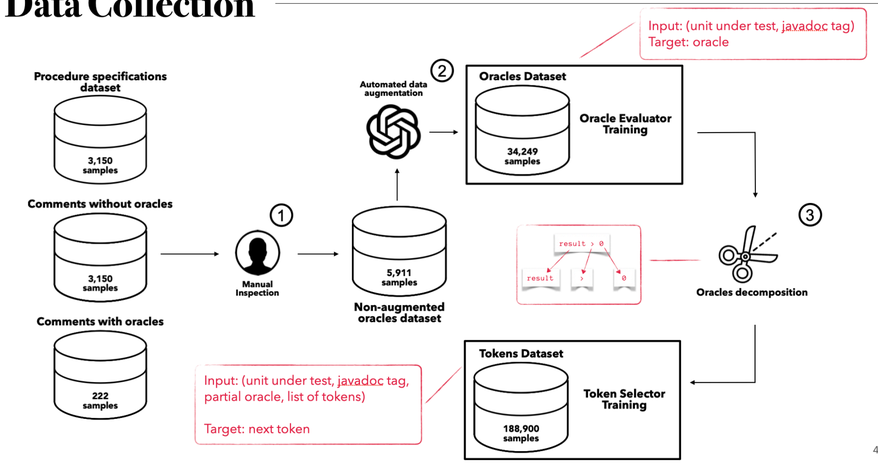
less false positives than GPT4

Our Approach: Neuro-Symbolic

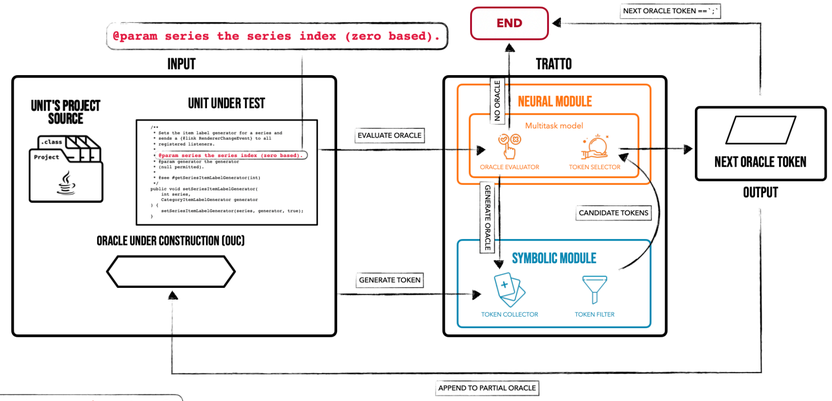


NEURO-SYMBOLIC APPROACHES

Data Collection

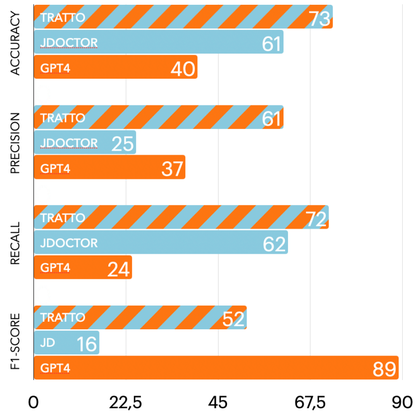


Workflow



Final oracle: `series >= 0`

Oracle Generation



RQ3

Compares the effectiveness of TraTTO to the state-of-the-art neural and symbolic approaches

Email
davide.molinelli@usi.ch
alberto.martin@usi.ch
mernst@cs.washington.edu
mauro.pezze@usi.ch
Website
https://star.inf.usi.ch/

Table 1. Accuracy (A), precision (P), recall (R), and F1-score ($F1$) for all approaches on ground-truth dataset.

		closure-compiler	commons-cli	commons-codec	commons-compress	commons-csv	commons-jxpath	commons-lang	gson	jackson-core	jackson-databind	jackson-dataformat	jfree-chart	joda-time	jsoup	mockito	Total
Tratto	A	56%	83%	61%	74%	91%	62%	75%	80%	68%	58%	67%	77%	81%	60%	67%	**73%**
	P	N/A	60%	29%	82%	88%	N/A	61%	70%	43%	33%	N/A	90%	83%	50%	N/A	**72%**
	R	0%	75%	24%	53%	97%	0%	67%	70%	33%	3%	0%	60%	78%	5%	0%	52%
	F1	N/A	67%	26%	64%	92%	N/A	64%	70%	37%	6%	N/A	72%	80%	8%	N/A	**61%**
Jdoctor	A	56%	72%	56%	73%	79%	62%	60%	64%	67%	61%	67%	47%	59%	57%	67%	61%
	P	N/A	100%	33%	100%	96%	N/A	0%	6%	33%	100%	N/A	N/A	74%	0%	N/A	62%
	R	0%	16%	4%	41%	68%	0%	0%	7%	11%	3%	0%	0%	33%	0%	0%	**16%**
	F1	N/A	29%	7%	58%	79%	N/A	N/A	6%	17%	6%	N/A	N/A	46%	N/A	N/A	**25%**
GPT4	A	18%	26%	37%	42%	55%	5%	53%	19%	47%	35%	50%	35%	53%	41%	50%	**40%**
	P	12%	7%	20%	27%	40%	2%	31%	3%	22%	13%	0%	27%	42%	23%	33%	**24%**
	R	100%	100%	92%	100%	92%	100%	88%	60%	100%	100%	N/A	92%	96%	67%	100%	**89%**
	F1	22%	12%	32%	42%	56%	4%	46%	6%	36%	24%	N/A	42%	58%	34%	50%	37%

Table 2. True/false positives/negatives ($TP/TN/FP/FN$) for all approaches on ground-truth dataset.

Project	M	O	NO	Tratto								Jdoctor								GPT4							
				TP		TN		FP		FN		TP		TN		FP		FN		TP		TN		FP		FN	
closure-compiler	3	4	5	0	(0%)	5	(56%)	0	(0%)	4	(44%)	0	(0%)	5	(56%)	0	(0%)	4	(44%)	2	(12%)	1	(6%)	14	(0%)	0	(82%)
commons-cli	6	6	12	3	(16%)	12	(67%)	2	(6%)	1	(11%)	1	(5%)	12	(67%)	0	(0%)	5	(28%)	1	(5%)	4	(21%)	14	(74%)	0	(0%)
commons-codec	19	24	33	4	(7%)	32	(54%)	10	(22%)	13	(17%)	1	(2%)	31	(54%)	2	(4%)	23	(40%)	11	(15%)	16	(22%)	45	(62%)	1	(1%)
commons-compress	12	17	20	9	(24%)	19	(50%)	2	(5%)	8	(21%)	7	(19%)	20	(54%)	0	(0%)	10	(27%)	12	(21%)	12	(21%)	33	(58%)	0	(0%)
commons-csv	16	35	23	30	(52%)	23	(40%)	4	(7%)	1	(2%)	23	(39%)	23	(39%)	1	(2%)	11	(18%)	24	(28%)	23	(27%)	36	(42%)	2	(2%)
commons-jxpath	4	5	8	0	(0%)	8	(62%)	0	(0%)	5	(38%)	0	(0%)	8	(62%)	0	(0%)	5	(38%)	1	(2%)	2	(4%)	49	(94%)	0	(0%)
commons-lang	53	64	103	37	(22%)	89	(53%)	24	(14%)	18	(11%)	0	(0%)	101	(61%)	2	(1%)	64	(38%)	38	(20%)	62	(33%)	83	(44%)	5	(3%)
gson	27	30	51	19	(23%)	46	(57%)	8	(10%)	8	(10%)	1	(1%)	51	(63%)	16	(20%)	13	(16%)	3	(3%)	18	(16%)	88	(79%)	2	(2%)
jackson-core	10	10	20	3	(10%)	18	(58%)	4	(13%)	6	(19%)	1	(3%)	19	(63%)	2	(7%)	8	(27%)	5	(15%)	11	(32%)	18	(53%)	0	(0%)
jackson-databind	24	30	48	1	(1%)	42	(57%)	2	(3%)	29	(39%)	1	(1%)	46	(60%)	0	(0%)	29	(39%)	13	(10%)	33	(25%)	84	(65%)	0	(0%)
jackson-dataformat	1	1	2	0	(0%)	2	(67%)	0	(0%)	1	(33%)	0	(0%)	2	(67%)	0	(0%)	1	(33%)	0	(0%)	2	(50%)	2	(50%)	0	(0%)
jfree-chart	25	46	40	26	(0%)	40	(47%)	3	(3%)	17	(20%)	0	(0%)	40	(47%)	0	(0%)	46	(53%)	34	(24%)	17	(12%)	90	(62%)	3	(2%)
joda-time	40	71	63	52	(39%)	56	(42%)	11	(8%)	15	(11%)	23	(17%)	56	(42%)	8	(6%)	47	(35%)	50	(33%)	31	(20%)	70	(46%)	2	(1%)
jsoup	33	45	66	2	(2%)	65	(59%)	2	(2%)	42	(38%)	0	(0%)	64	(57%)	4	(3%)	45	(40%)	22	(16%)	36	(25%)	73	(51%)	11	(8%)
mockito	1	1	2	0	(0%)	2	(67%)	0	(0%)	1	(33%)	0	(0%)	2	(67%)	0	(0%)	1	(33%)	1	(25%)	1	(25%)	2	(50%)	0	(0%)
Total	274	389	496	186	**(21%)**	459	(52%)	72	(8%)	169	(19%)	58	**(7%)**	480	**(54%)**	35	**(4%)**	312	**(35%)**	217	(18%)	269	**(22%)**	701	**(58%)**	26	**(2%)**