PL 구현체를 위한 새로운 커버리지를 제안하기까지의 여정

박지혁

고려대학교 정보대학 컴퓨터학과

(with 안승민, 윤동준, 박지희, 김경원, 이강욱, 류석영 교수님)

SIGPL Summer School 2023

2023.08.24

Background + Problem

PLDI 2023

Feature-Sens. Coverage

ASE 2020

ICSE 2021

ECMA-262 (JS Spec.)

Mech. Spec. Extraction

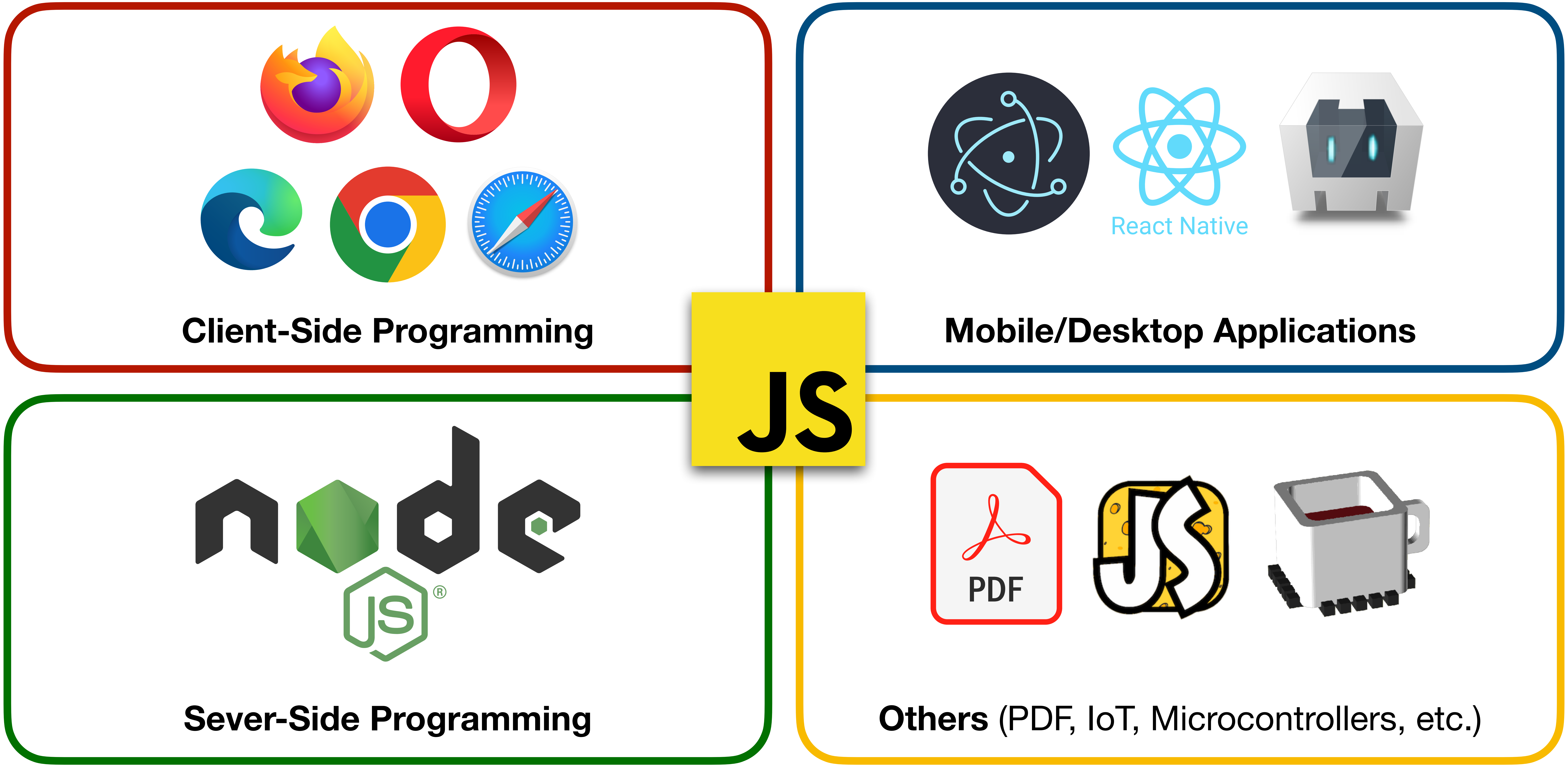
Mechanized Spec.

Conform. Test Synthesis

Conformance Tests

JavaScript Engines

JavaScript is Everywhere



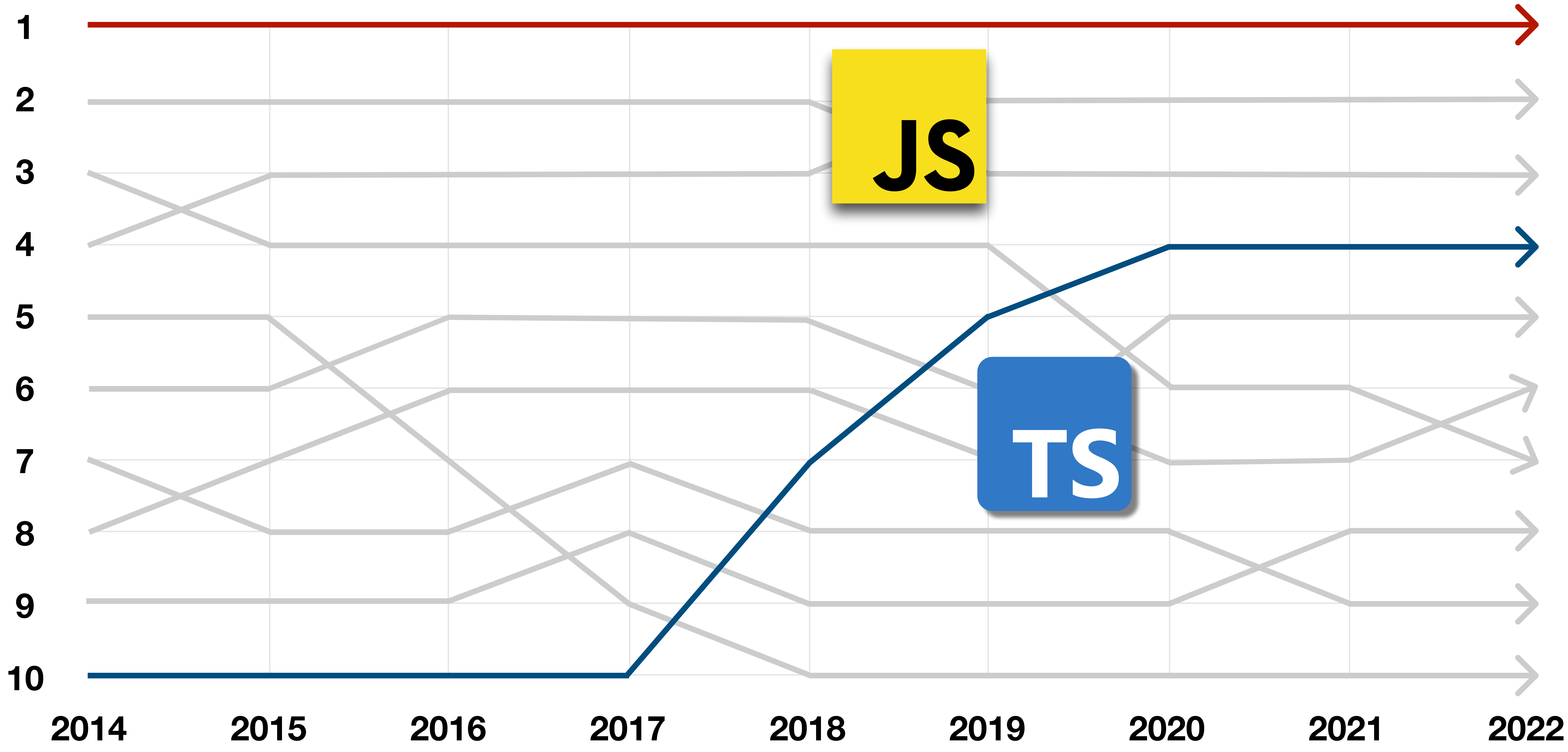
Client-Side Programming

Mobile/Desktop Applications

Sever-Side Programming

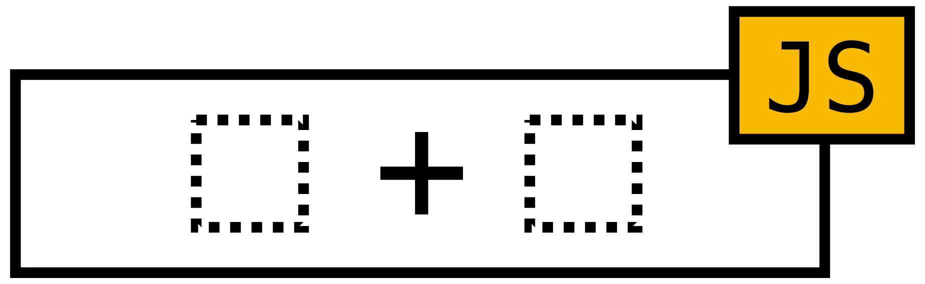
Others (PDF, IoT, Microcontrollers, etc.)

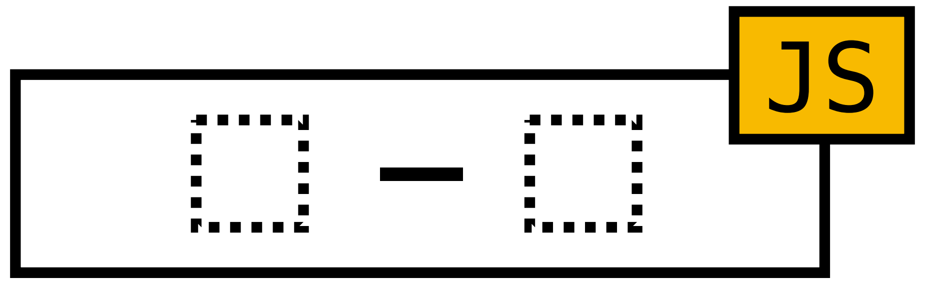
JavaScript is Everywhere



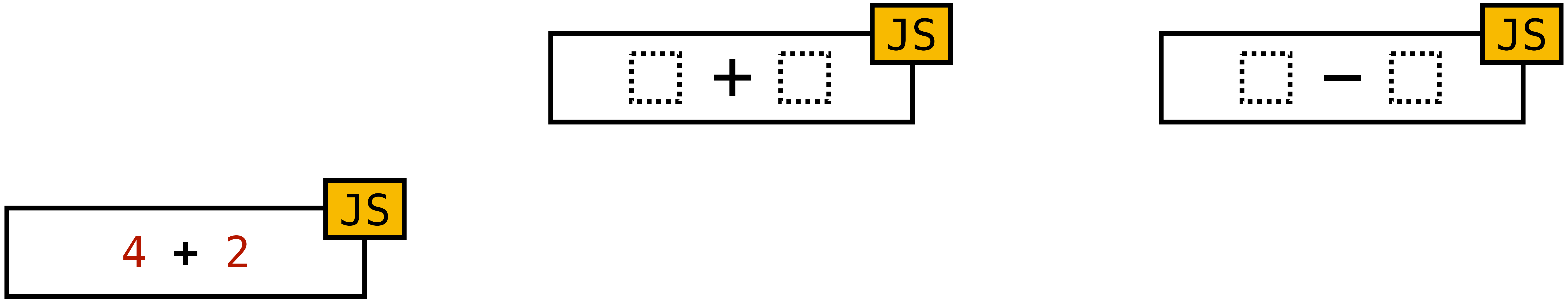
https://octoverse.github.com/

But, JavaScript is Complicated

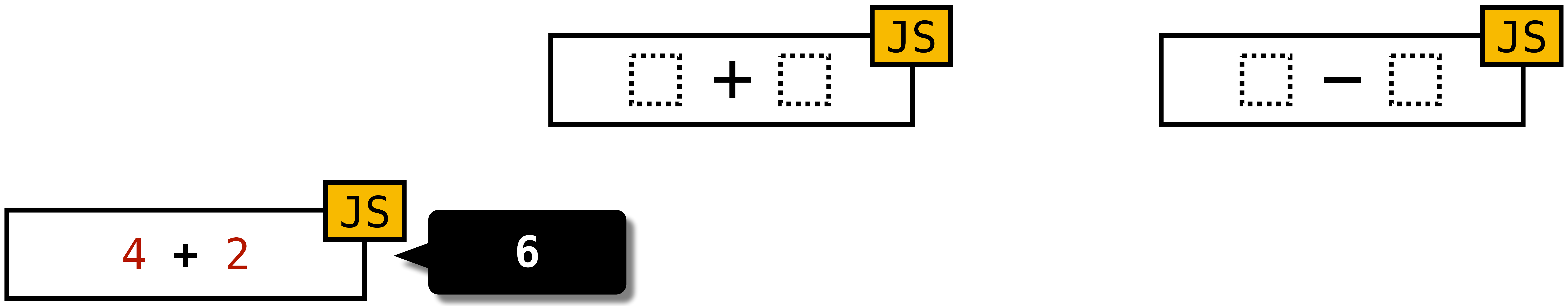




But, JavaScript is Complicated



But, JavaScript is Complicated



JS

□ + □

JS

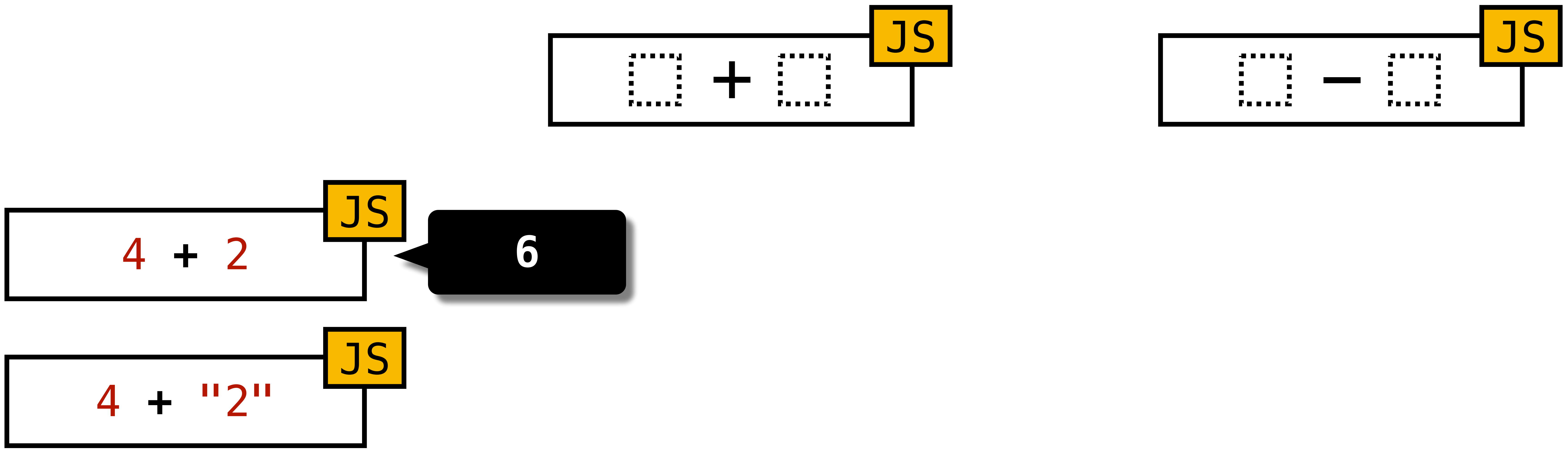
□ − □

JS

4 + 2

6

But, JavaScript is Complicated



JS

```
□ + □
```

JS

```
□ − □
```

JS

```
4 + 2
```

6

JS

```
4 + "2"
```

But, **JavaScript** is Complicated

JS

```
⬚ + ⬚
```

JS

```
⬚ − ⬚
```

JS

```
4 + 2
```

```
6
```

JS

```
4 + "2"
```

```
"42"
```

But, JavaScript is Complicated

JS
```
□ + □
```

JS
```
□ − □
```

JS
```
4 + 2
```

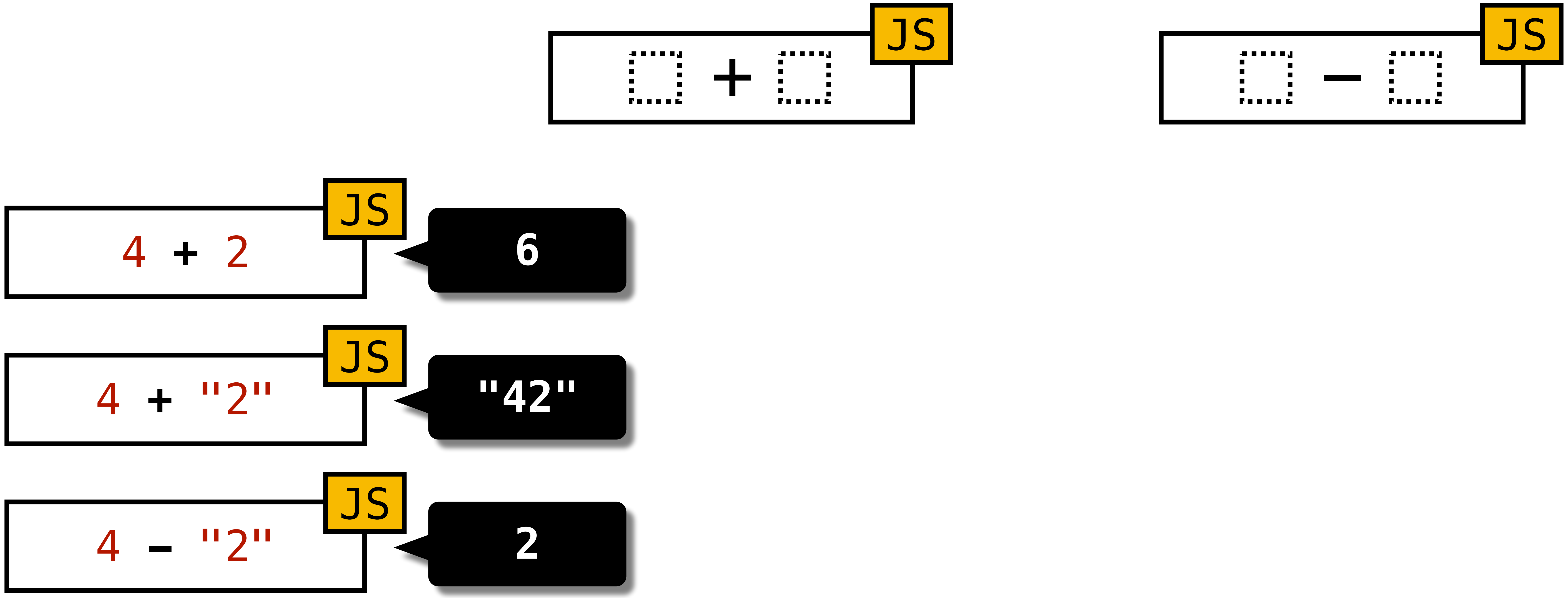
```
6
```

JS
```
4 + "2"
```

```
"42"
```

JS
```
4 - "2"
```

But, JavaScript is Complicated



JS

```
□ + □
```

JS

```
□ − □
```

JS

```
4 + 2
```

```
6
```

JS

```
4 + "2"
```

```
"42"
```

JS

```
4 - "2"
```

```
2
```

But, JavaScript is Complicated

JS
```
⬚ + ⬚
```

JS
```
⬚ - ⬚
```

JS
```
4 + 2
```
```
6
```

JS
```
4 + "2"
```
```
"42"
```

JS
```
4 - "2"
```
```
2
```

JS
```
[1,2] + 3
```

But, JavaScript is Complicated

JS
```
□ + □
```

JS
```
□ - □
```

JS
```
4 + 2
```
6

JS
```
[1,2] + 3
```
"1,23"

JS
```
4 + "2"
```
"42"

JS
```
4 - "2"
```
2

But, JavaScript is Complicated

JS
□ + □

JS
□ − □

JS
```
4 + 2
```
```
6
```

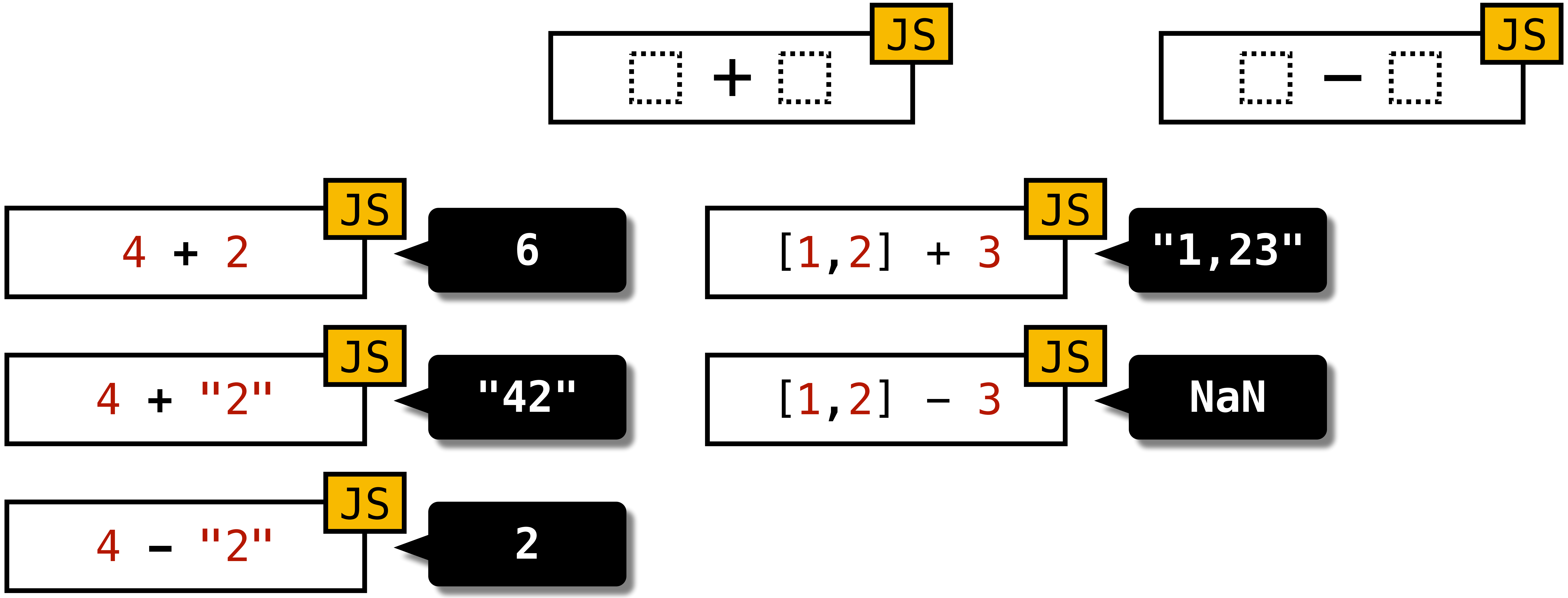
JS
```
4 + "2"
```
```
"42"
```

JS
```
4 - "2"
```
```
2
```

JS
```
[1,2] + 3
```
```
"1,23"
```

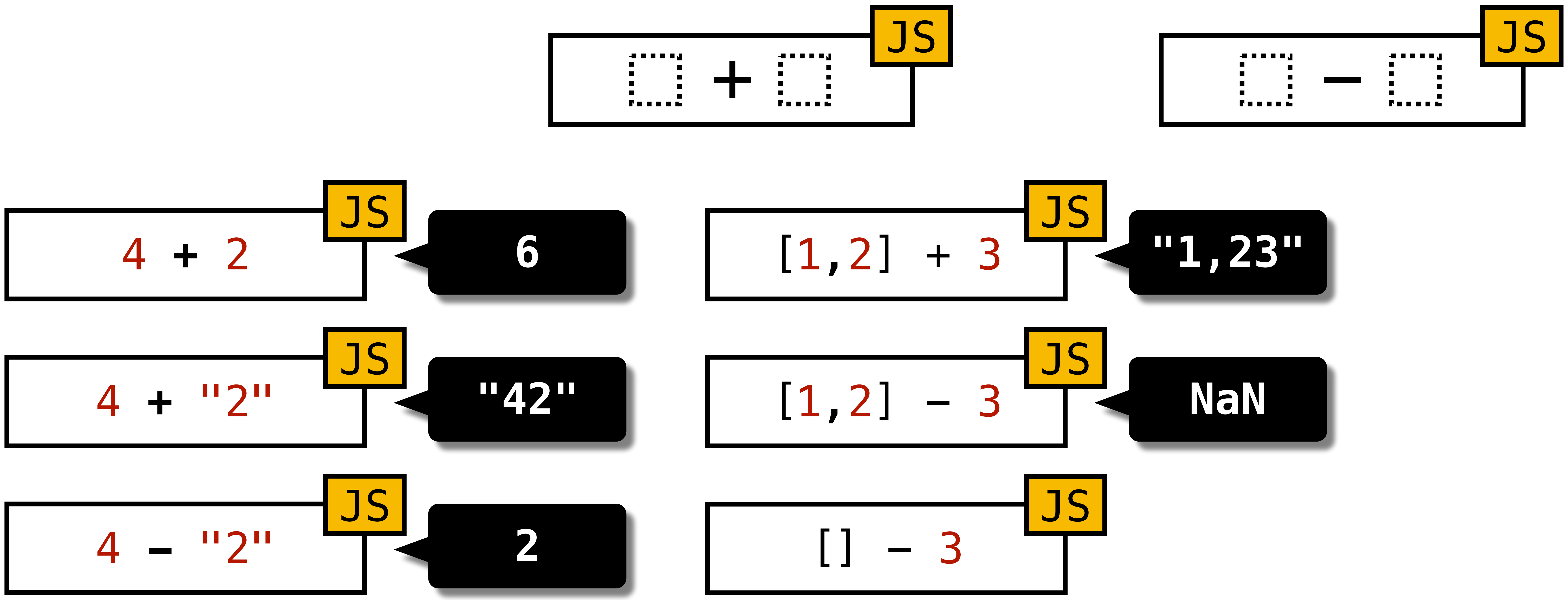
JS
```
[1,2] - 3
```

But, JavaScript is Complicated



```
4 + 2        6
4 + "2"      "42"
4 - "2"      2
[1,2] + 3    "1,23"
[1,2] - 3    NaN
```

But, JavaScript is Complicated



JS `□ + □`

JS `□ − □`

JS `4 + 2` → `6`

JS `4 + "2"` → `"42"`

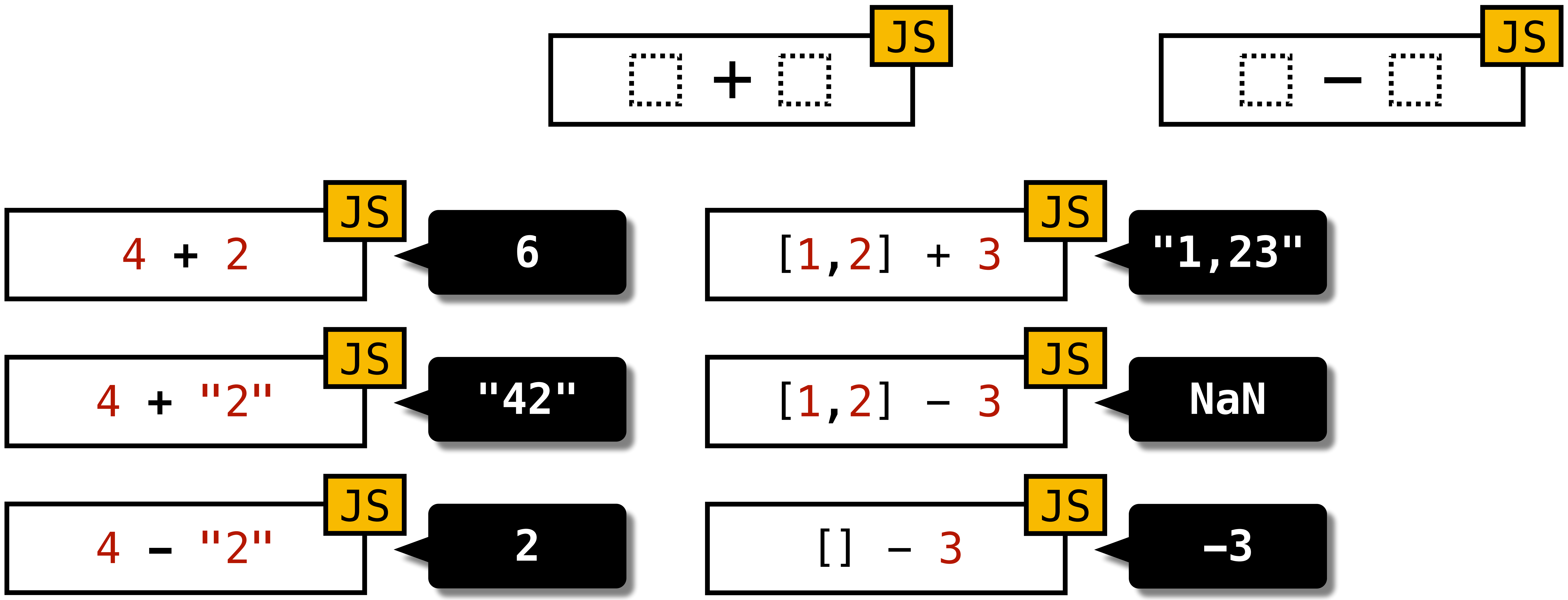
JS `4 - "2"` → `2`

JS `[1,2] + 3` → `"1,23"`

JS `[1,2] - 3` → `NaN`

JS `[] - 3`

But, JavaScript is Complicated

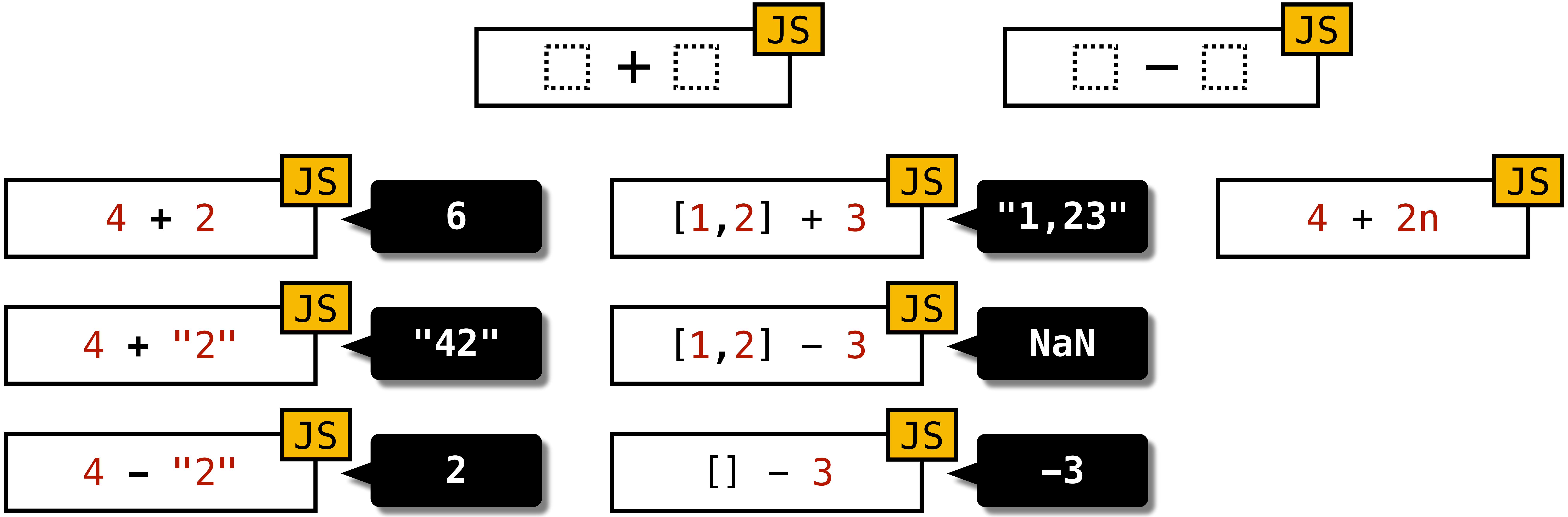


JS
□ + □

JS
□ − □

```
4 + 2          6
4 + "2"        "42"
4 - "2"        2
[1,2] + 3      "1,23"
[1,2] - 3      NaN
[] - 3         -3
```

But, JavaScript is Complicated



JS
```
□ + □
```

JS
```
□ − □
```

JS
```
4 + 2
```
6

JS
```
4 + "2"
```
"42"

JS
```
4 - "2"
```
2

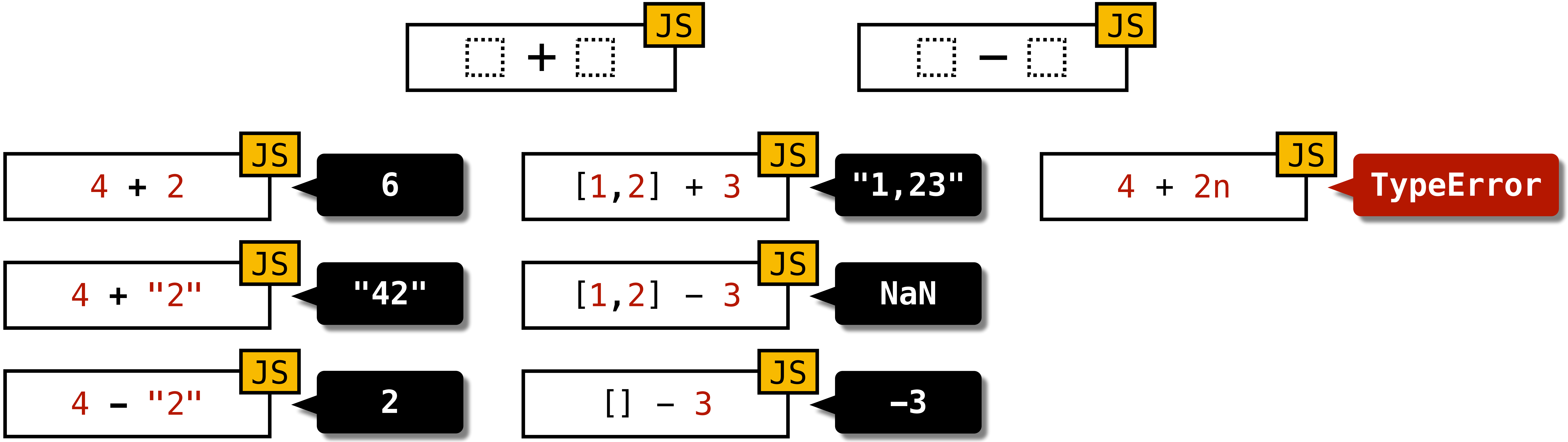
JS
```
[1,2] + 3
```
"1,23"

JS
```
[1,2] - 3
```
NaN

JS
```
[] - 3
```
-3

JS
```
4 + 2n
```

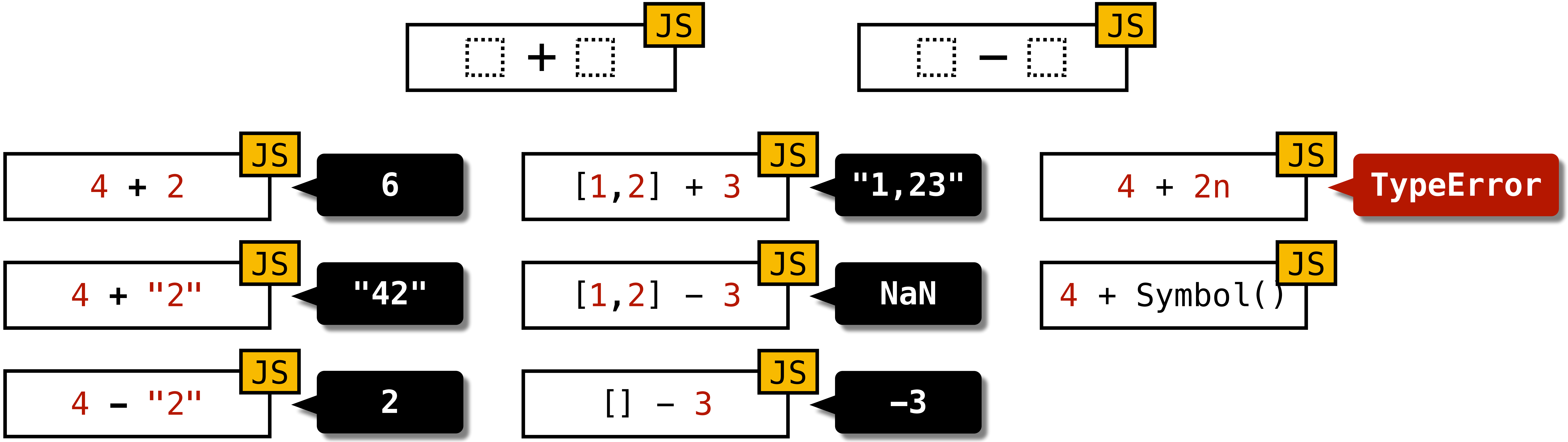
But, JavaScript is Complicated



```
⬚ + ⬚          JS
⬚ − ⬚          JS
```

```
4 + 2        JS   6
4 + "2"      JS   "42"
4 - "2"      JS   2
[1,2] + 3    JS   "1,23"
[1,2] - 3    JS   NaN
[] - 3       JS   -3
4 + 2n       JS   TypeError
```

But, JavaScript is Complicated



JS `□ + □`

JS `□ − □`

```
4 + 2          → 6
4 + "2"        → "42"
4 - "2"        → 2
[1,2] + 3      → "1,23"
[1,2] - 3      → NaN
[] - 3         → -3
4 + 2n         → TypeError
4 + Symbol()
```

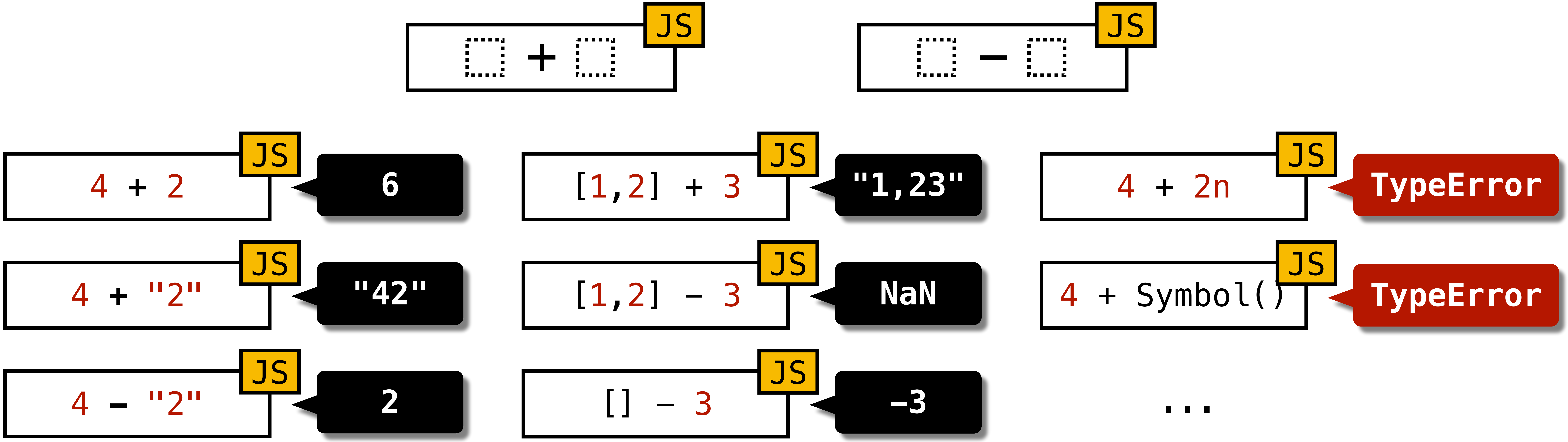
But, JavaScript is Complicated

JS
```
⬚ + ⬚
```

JS
```
⬚ − ⬚
```

JS	
`4 + 2`	6
`4 + "2"`	"42"
`4 - "2"`	2
`[1,2] + 3`	"1,23"
`[1,2] - 3`	NaN
`[] - 3`	-3
`4 + 2n`	TypeError
`4 + Symbol()`	TypeError

But, JavaScript is Complicated



JS `⬚ + ⬚`

JS `⬚ - ⬚`

```
4 + 2            6
4 + "2"          "42"
4 - "2"          2
[1,2] + 3        "1,23"
[1,2] - 3        NaN
[] - 3           -3
4 + 2n           TypeError
4 + Symbol()     TypeError
...
```

But, JavaScript is Complicated

JS
```
□ + □
```

JS
```
□ − □
```

JS
```
4 + 2
```
6

JS
```
4 + "2"
```
"42"

JS
```
4 - "2"
```
2

JS
```
[1,2] + 3
```
"1,23"

JS
```
[1,2] - 3
```
NaN

JS
```
[] - 3
```
-3

JS
```
4 + 2n
```
TypeError

JS
```
4 + Symbol()
```
TypeError

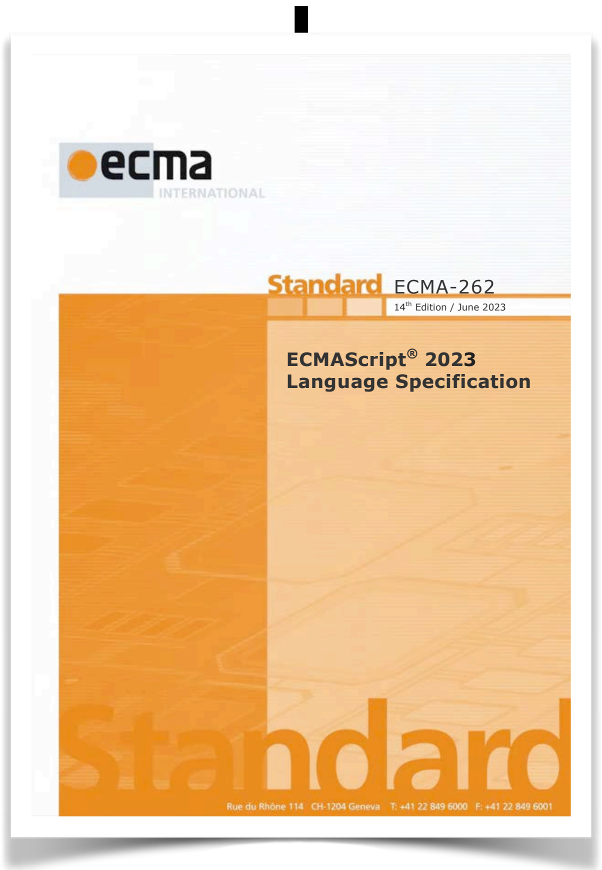
...

JS
```
(![]+[])[+[]]                    +   // "f"
(![]+[])[+!+[]]                  +   // "a"
([![]]+[][[]])[+!+[]+[+[]]] +   // "i"
(![]+[])[!+[]+!+[]]                  // "l"
```
"fail"

Language Specification (ECMA-262) of JavaScript

⬚ + ⬚ JS

TC 39


ecma INTERNATIONAL

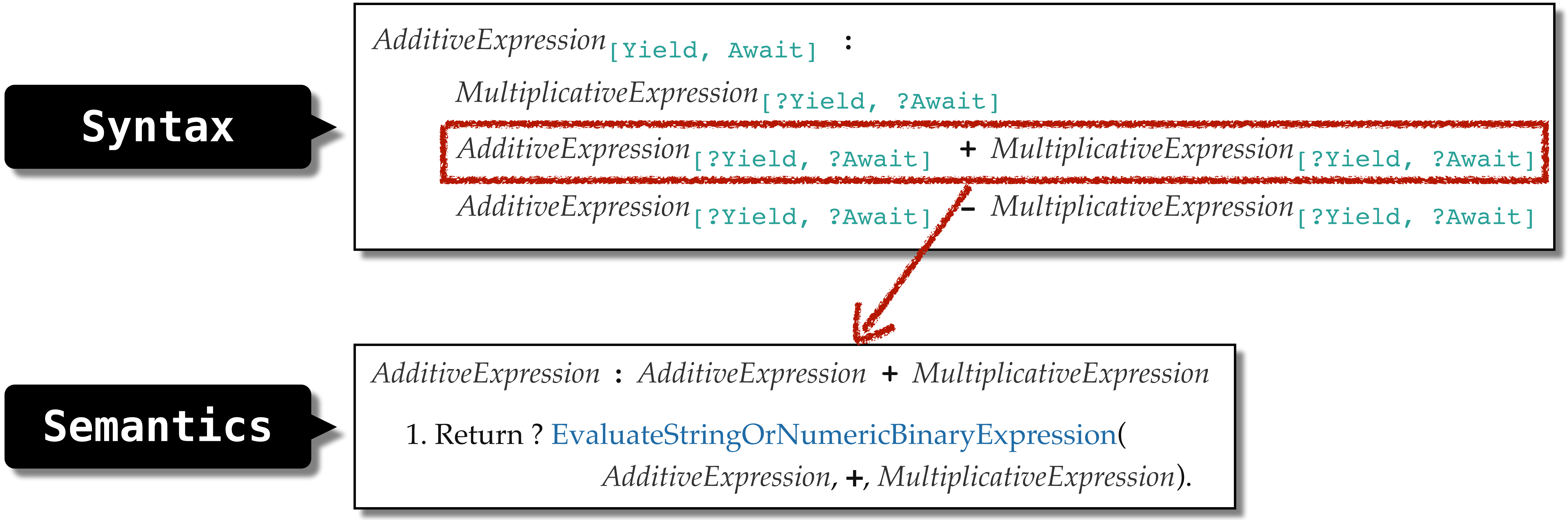
Standard ECMA-262

14th Edition / June 2023

ECMAScript® 2023 Language Specification

Standard

ECMA-262
(JavaScript Spec.)



AdditiveExpression$_{\texttt{[Yield, Await]}}$:
  MultiplicativeExpression$_{\texttt{[?Yield, ?Await]}}$
  AdditiveExpression$_{\texttt{[?Yield, ?Await]}}$ **+** *MultiplicativeExpression*$_{\texttt{[?Yield, ?Await]}}$
  AdditiveExpression$_{\texttt{[?Yield, ?Await]}}$ **-** *MultiplicativeExpression*$_{\texttt{[?Yield, ?Await]}}$

AdditiveExpression : *AdditiveExpression* **+** *MultiplicativeExpression*

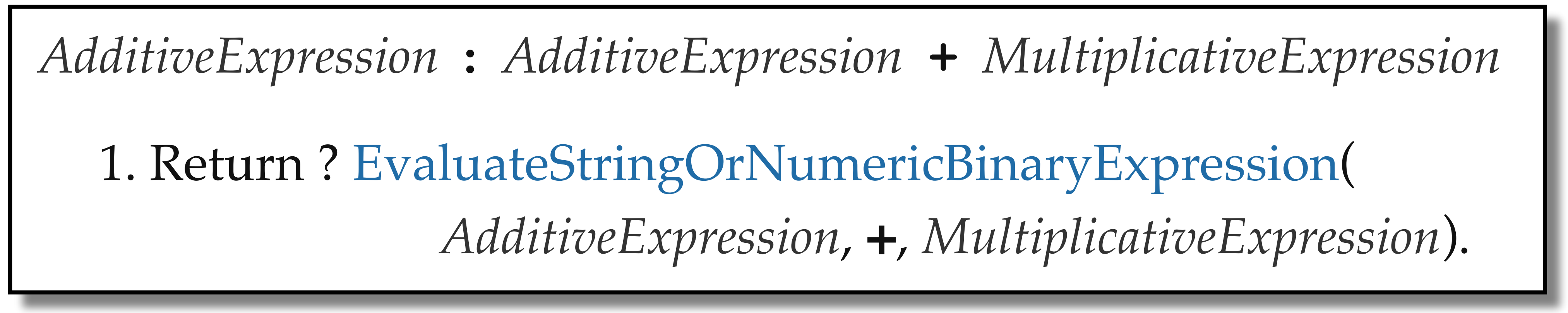
1. Return ? EvaluateStringOrNumericBinaryExpression(*AdditiveExpression*, **+**, *MultiplicativeExpression*).

Language Specification (ECMA-262) of JavaScript

JS

```
4 + 2n
```

TypeError



AdditiveExpression **:** *AdditiveExpression* **+** *MultiplicativeExpression*

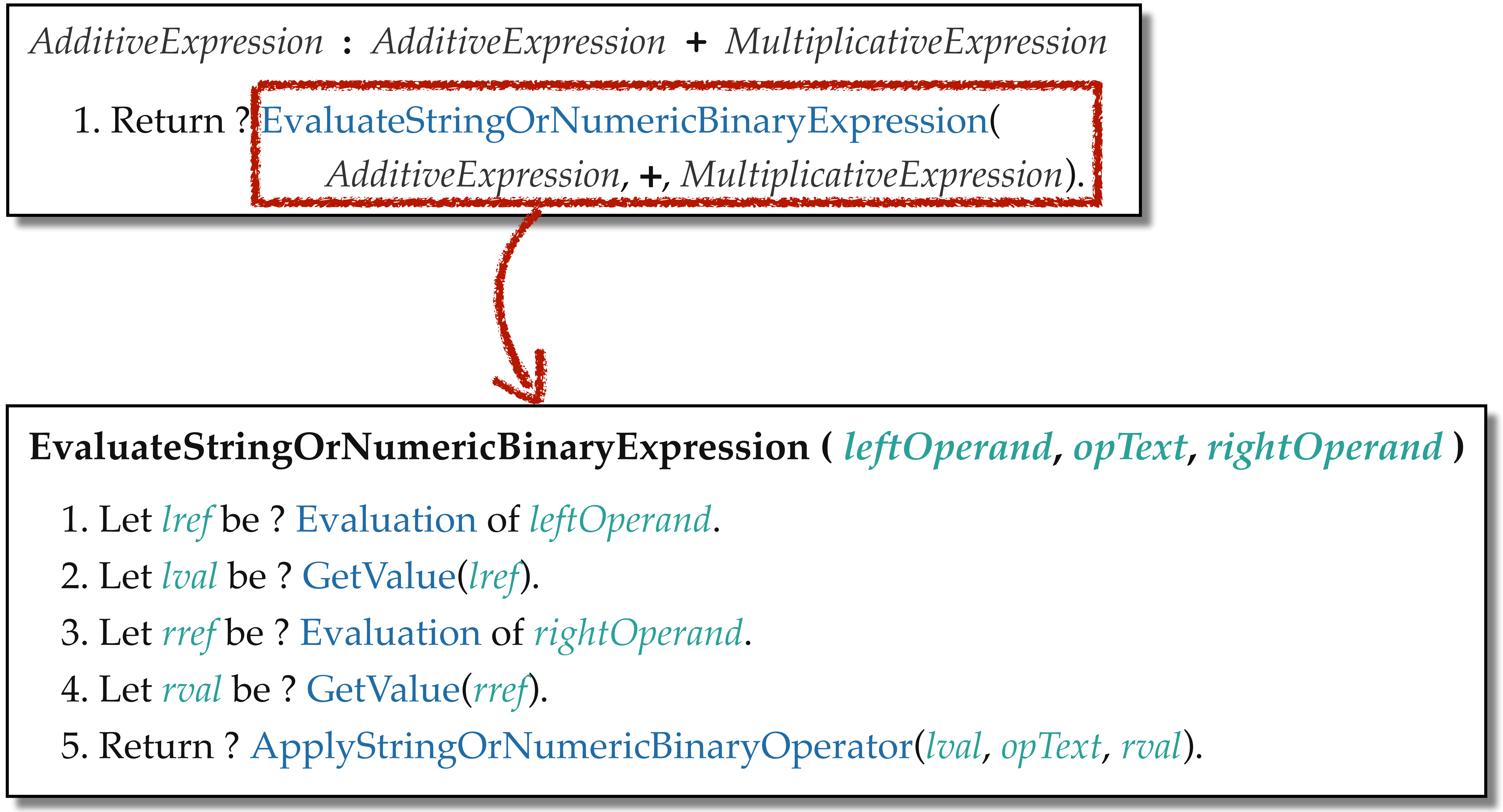
1. Return ? EvaluateStringOrNumericBinaryExpression(*AdditiveExpression*, **+**, *MultiplicativeExpression*).

Language Specification (ECMA-262) of JavaScript

JS

```
4 + 2n
```

TypeError



AdditiveExpression **:** *AdditiveExpression* **+** *MultiplicativeExpression*

```
1. Return ? EvaluateStringOrNumericBinaryExpression(
       AdditiveExpression, +, MultiplicativeExpression).
```

EvaluateStringOrNumericBinaryExpression (*leftOperand*, *opText*, *rightOperand* **)**

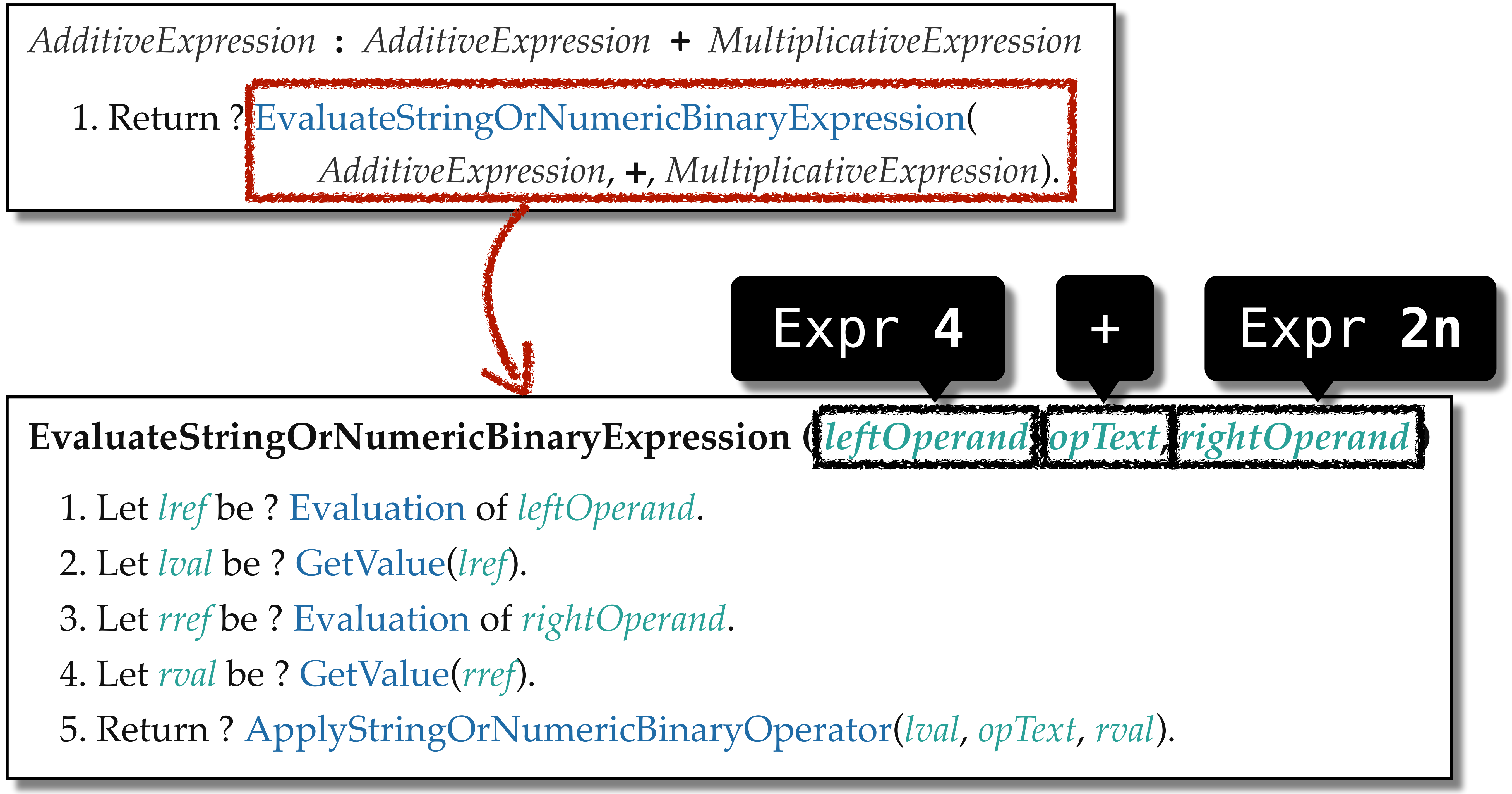
```
1. Let lref be ? Evaluation of leftOperand.
2. Let lval be ? GetValue(lref).
3. Let rref be ? Evaluation of rightOperand.
4. Let rval be ? GetValue(rref).
5. Return ? ApplyStringOrNumericBinaryOperator(lval, opText, rval).
```

Language Specification (ECMA-262) of JavaScript

JS

```
4 + 2n
```

TypeError



AdditiveExpression : *AdditiveExpression* **+** *MultiplicativeExpression*

1. Return ? EvaluateStringOrNumericBinaryExpression(*AdditiveExpression*, **+**, *MultiplicativeExpression*).

EvaluateStringOrNumericBinaryExpression (*leftOperand*, *opText*, *rightOperand*)

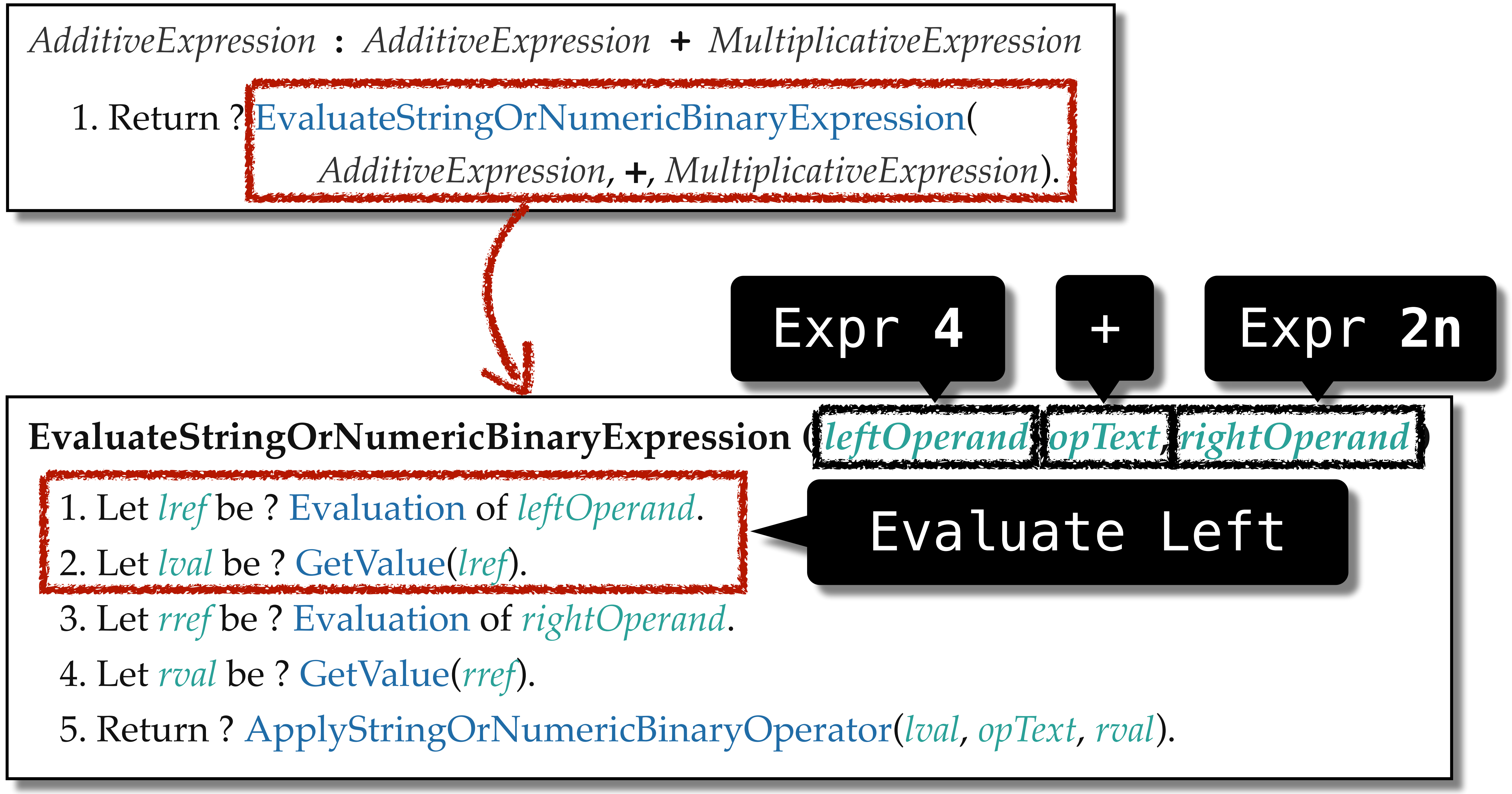
1. Let *lref* be ? Evaluation of *leftOperand*.
2. Let *lval* be ? GetValue(*lref*).
3. Let *rref* be ? Evaluation of *rightOperand*.
4. Let *rval* be ? GetValue(*rref*).
5. Return ? ApplyStringOrNumericBinaryOperator(*lval*, *opText*, *rval*).

Language Specification (ECMA-262) of JavaScript

JS

```
4 + 2n
```

TypeError



AdditiveExpression **:** *AdditiveExpression* **+** *MultiplicativeExpression*

1. Return ? EvaluateStringOrNumericBinaryExpression(*AdditiveExpression*, **+**, *MultiplicativeExpression*).

Expr 4

+

Expr 2n

EvaluateStringOrNumericBinaryExpression (*leftOperand*, *opText*, *rightOperand*)

1. Let *lref* be ? Evaluation of *leftOperand*.
2. Let *lval* be ? GetValue(*lref*).
3. Let *rref* be ? Evaluation of *rightOperand*.
4. Let *rval* be ? GetValue(*rref*).
5. Return ? ApplyStringOrNumericBinaryOperator(*lval*, *opText*, *rval*).

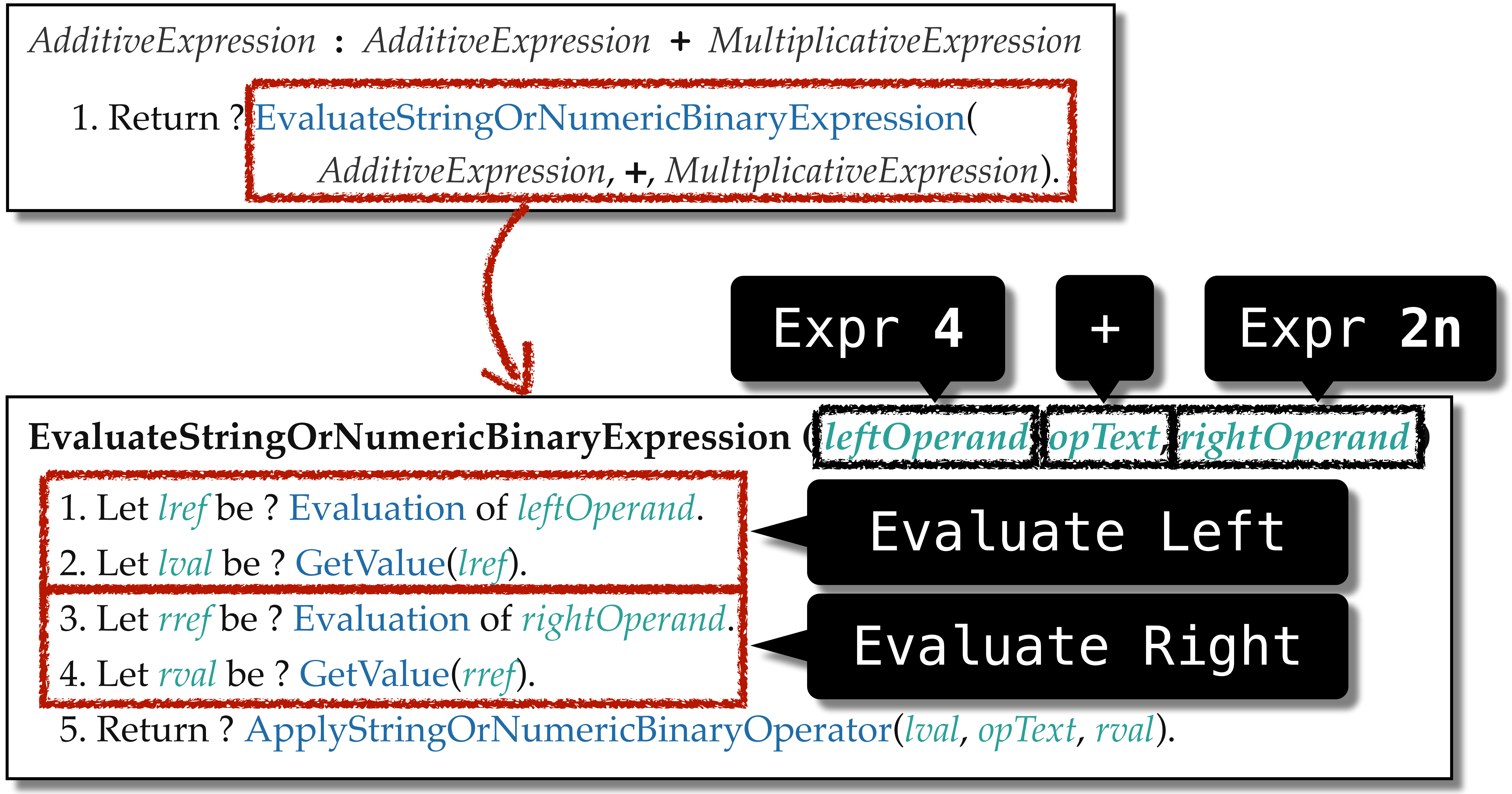
Evaluate Left

Language Specification (ECMA-262) of JavaScript

JS

```
4 + 2n
```

TypeError



AdditiveExpression : *AdditiveExpression* **+** *MultiplicativeExpression*

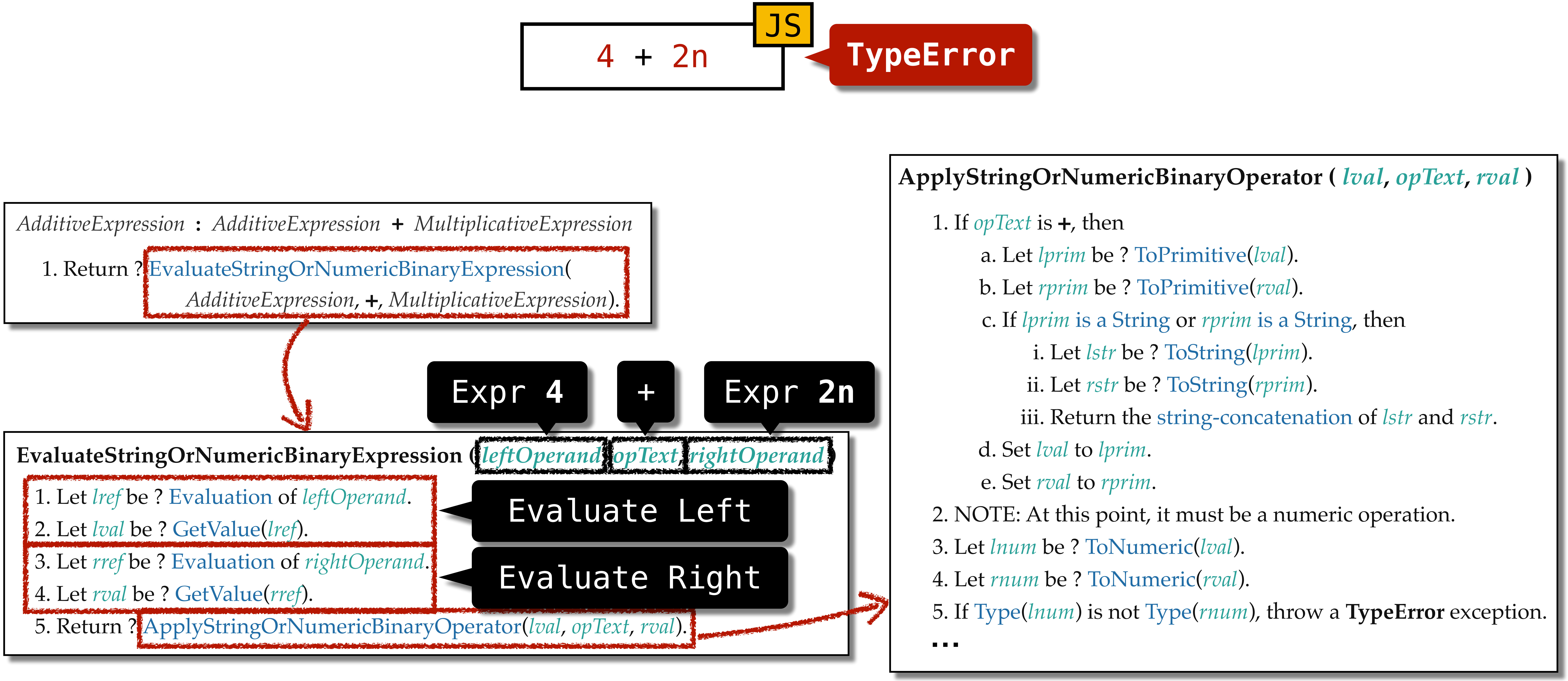
```
1. Return ? EvaluateStringOrNumericBinaryExpression(
       AdditiveExpression, +, MultiplicativeExpression).
```

EvaluateStringOrNumericBinaryExpression (*leftOperand*, *opText*, *rightOperand*)

```
1. Let lref be ? Evaluation of leftOperand.
2. Let lval be ? GetValue(lref).
3. Let rref be ? Evaluation of rightOperand.
4. Let rval be ? GetValue(rref).
5. Return ? ApplyStringOrNumericBinaryOperator(lval, opText, rval).
```

PLRG

Language Specification (ECMA-262) of JavaScript



```
4 + 2n
```

AdditiveExpression **:** *AdditiveExpression* **+** *MultiplicativeExpression*

```
1. Return ? EvaluateStringOrNumericBinaryExpression(
       AdditiveExpression, +, MultiplicativeExpression).
```

EvaluateStringOrNumericBinaryExpression (*leftOperand*, *opText*, *rightOperand* **)**

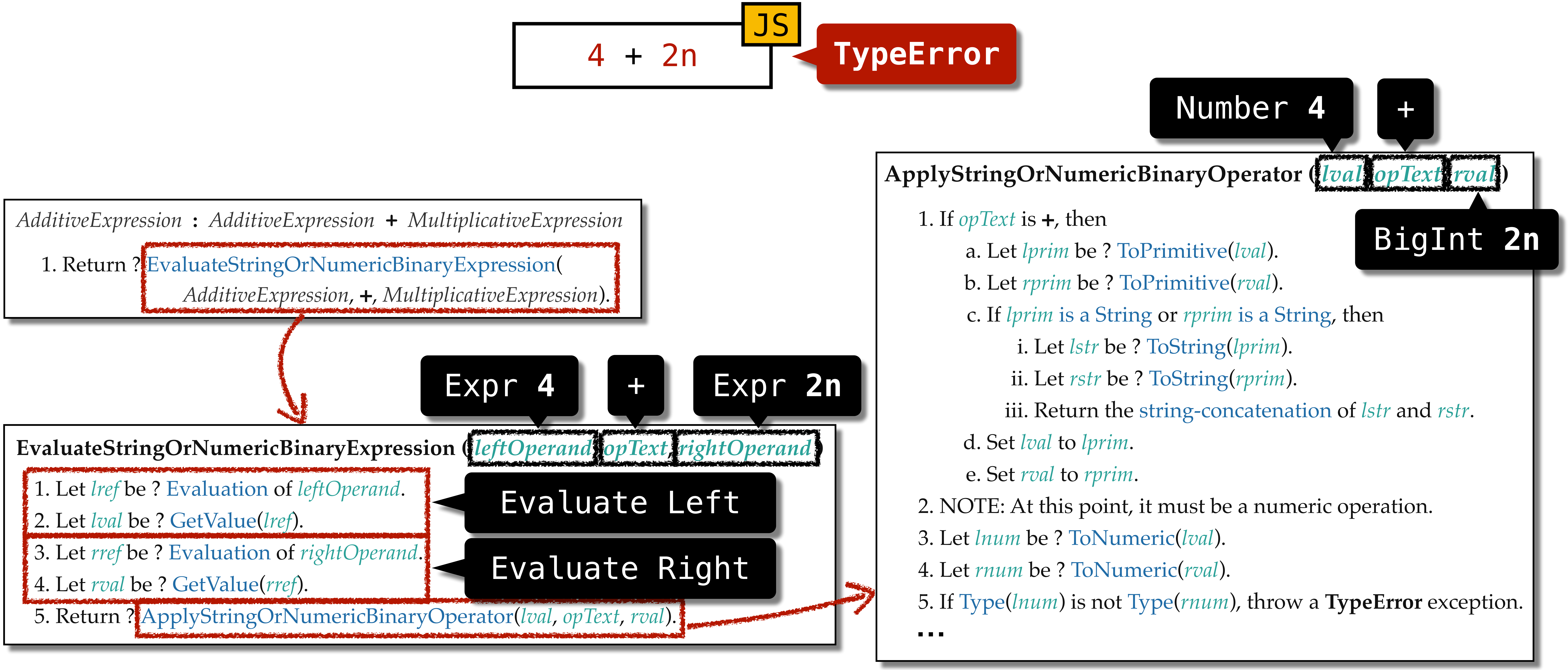
```
1. Let lref be ? Evaluation of leftOperand.
2. Let lval be ? GetValue(lref).
3. Let rref be ? Evaluation of rightOperand.
4. Let rval be ? GetValue(rref).
5. Return ? ApplyStringOrNumericBinaryOperator(lval, opText, rval).
```

ApplyStringOrNumericBinaryOperator (*lval*, *opText*, *rval* **)**

```
1. If opText is +, then
    a. Let lprim be ? ToPrimitive(lval).
    b. Let rprim be ? ToPrimitive(rval).
    c. If lprim is a String or rprim is a String, then
        i. Let lstr be ? ToString(lprim).
        ii. Let rstr be ? ToString(rprim).
        iii. Return the string-concatenation of lstr and rstr.
    d. Set lval to lprim.
    e. Set rval to rprim.
2. NOTE: At this point, it must be a numeric operation.
3. Let lnum be ? ToNumeric(lval).
4. Let rnum be ? ToNumeric(rval).
5. If Type(lnum) is not Type(rnum), throw a TypeError exception.
...
```

PLRG

Language Specification (ECMA-262) of JavaScript



```
AdditiveExpression : AdditiveExpression + MultiplicativeExpression

  1. Return ? EvaluateStringOrNumericBinaryExpression(
         AdditiveExpression, +, MultiplicativeExpression).
```

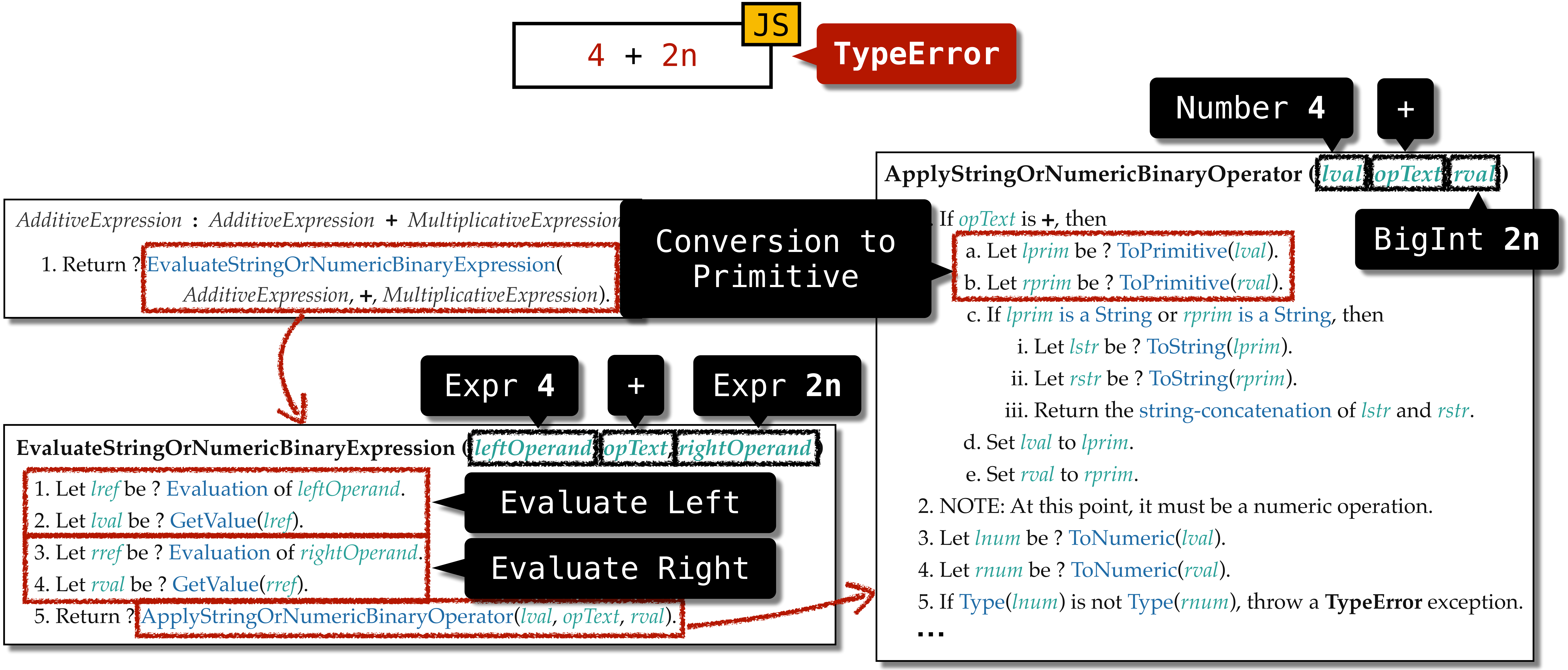
```
EvaluateStringOrNumericBinaryExpression (leftOperand, opText, rightOperand)

  1. Let lref be ? Evaluation of leftOperand.
  2. Let lval be ? GetValue(lref).
  3. Let rref be ? Evaluation of rightOperand.
  4. Let rval be ? GetValue(rref).
  5. Return ? ApplyStringOrNumericBinaryOperator(lval, opText, rval).
```

```
ApplyStringOrNumericBinaryOperator (lval, opText, rval)

  1. If opText is +, then
      a. Let lprim be ? ToPrimitive(lval).
      b. Let rprim be ? ToPrimitive(rval).
      c. If lprim is a String or rprim is a String, then
          i. Let lstr be ? ToString(lprim).
          ii. Let rstr be ? ToString(rprim).
          iii. Return the string-concatenation of lstr and rstr.
      d. Set lval to lprim.
      e. Set rval to rprim.
  2. NOTE: At this point, it must be a numeric operation.
  3. Let lnum be ? ToNumeric(lval).
  4. Let rnum be ? ToNumeric(rval).
  5. If Type(lnum) is not Type(rnum), throw a TypeError exception.
  ...
```

Language Specification (ECMA-262) of JavaScript



```
4 + 2n
```

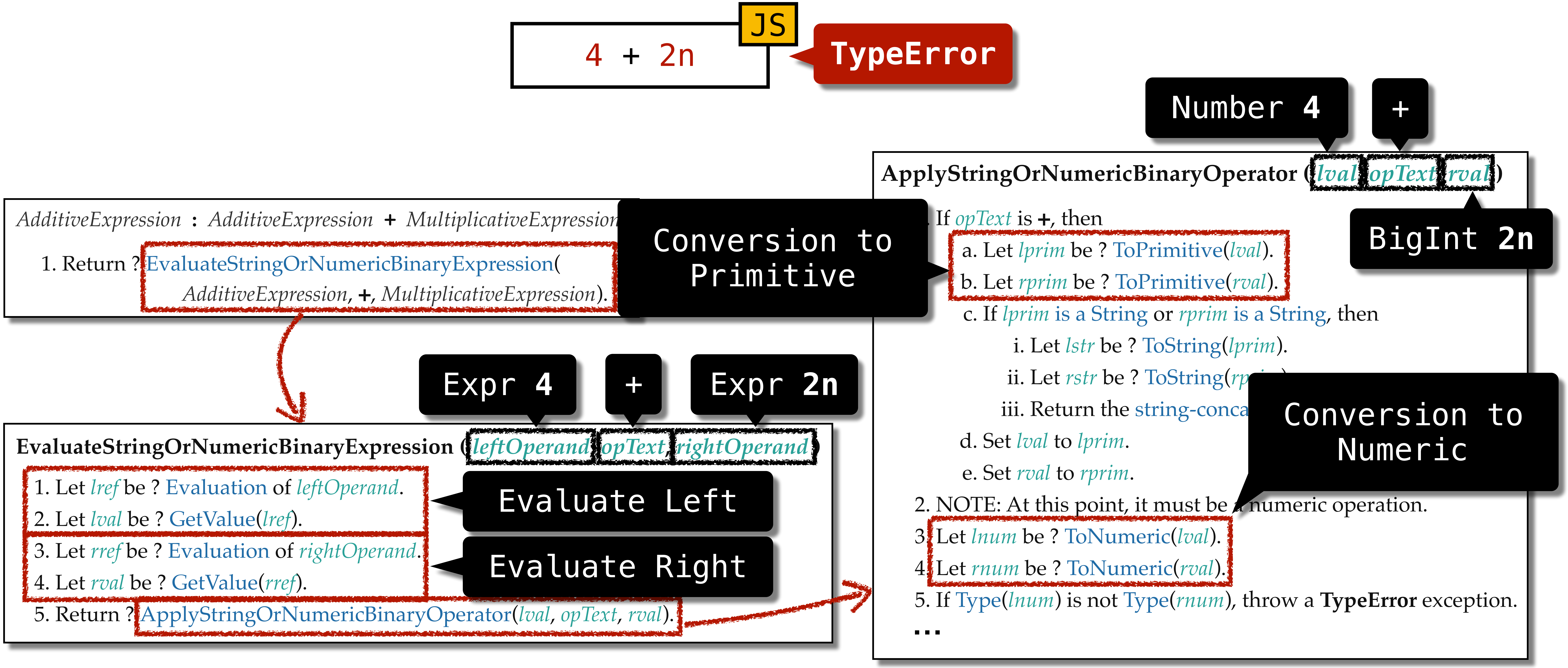
```
AdditiveExpression : AdditiveExpression + MultiplicativeExpression
  1. Return ? EvaluateStringOrNumericBinaryExpression(
         AdditiveExpression, +, MultiplicativeExpression).
```

```
EvaluateStringOrNumericBinaryExpression (leftOperand, opText, rightOperand)
  1. Let lref be ? Evaluation of leftOperand.
  2. Let lval be ? GetValue(lref).
  3. Let rref be ? Evaluation of rightOperand.
  4. Let rval be ? GetValue(rref).
  5. Return ? ApplyStringOrNumericBinaryOperator(lval, opText, rval).
```

```
ApplyStringOrNumericBinaryOperator (lval, opText, rval)
  1. If opText is +, then
     a. Let lprim be ? ToPrimitive(lval).
     b. Let rprim be ? ToPrimitive(rval).
     c. If lprim is a String or rprim is a String, then
        i. Let lstr be ? ToString(lprim).
        ii. Let rstr be ? ToString(rprim).
        iii. Return the string-concatenation of lstr and rstr.
     d. Set lval to lprim.
     e. Set rval to rprim.
  2. NOTE: At this point, it must be a numeric operation.
  3. Let lnum be ? ToNumeric(lval).
  4. Let rnum be ? ToNumeric(rval).
  5. If Type(lnum) is not Type(rnum), throw a TypeError exception.
  ...
```

PLRG

Language Specification (ECMA-262) of JavaScript



AdditiveExpression : *AdditiveExpression* **+** *MultiplicativeExpression*

```
1. Return ? EvaluateStringOrNumericBinaryExpression(
       AdditiveExpression, +, MultiplicativeExpression).
```

EvaluateStringOrNumericBinaryExpression (*leftOperand*, *opText*, *rightOperand*)

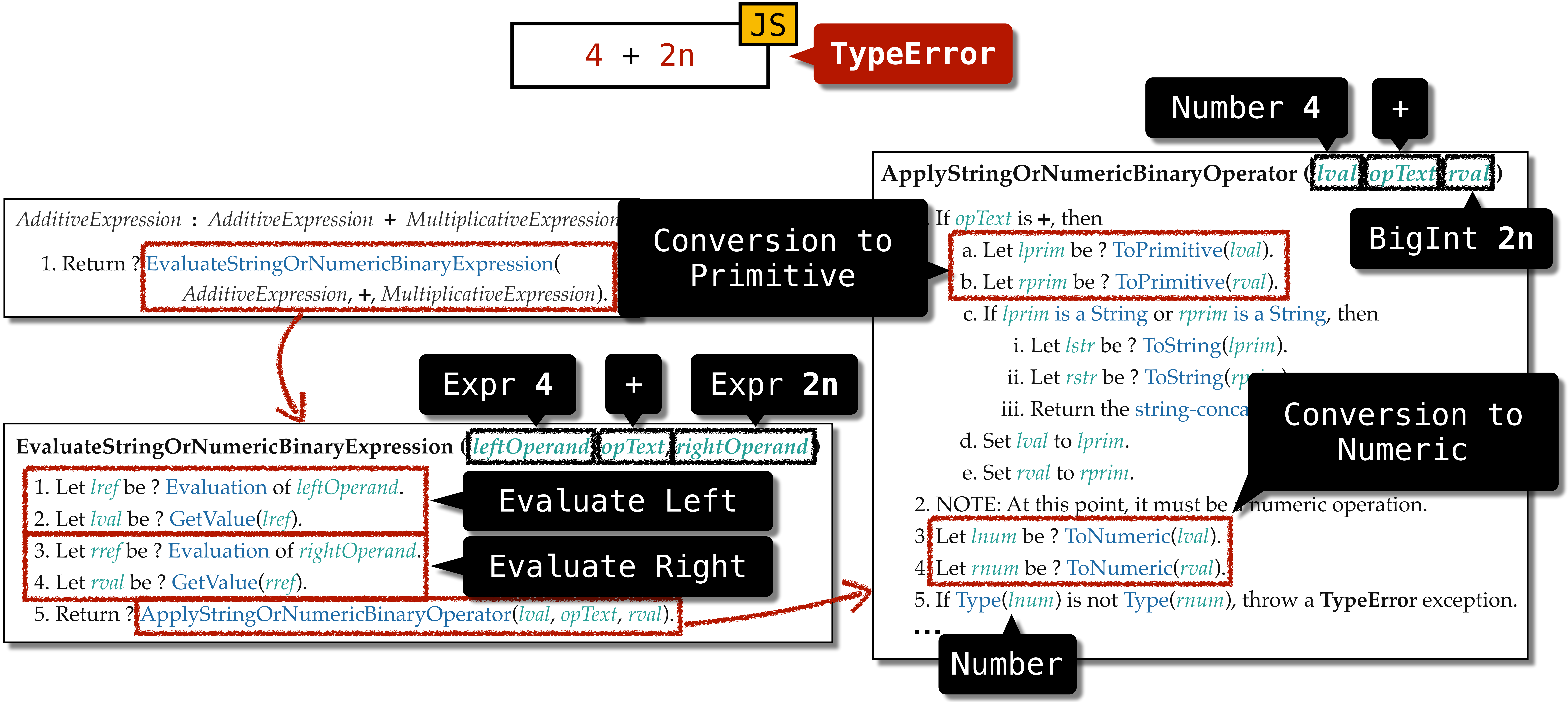
```
1. Let lref be ? Evaluation of leftOperand.
2. Let lval be ? GetValue(lref).
3. Let rref be ? Evaluation of rightOperand.
4. Let rval be ? GetValue(rref).
5. Return ? ApplyStringOrNumericBinaryOperator(lval, opText, rval).
```

ApplyStringOrNumericBinaryOperator (*lval*, *opText*, *rval*)

```
   If opText is +, then
     a. Let lprim be ? ToPrimitive(lval).
     b. Let rprim be ? ToPrimitive(rval).
     c. If lprim is a String or rprim is a String, then
         i. Let lstr be ? ToString(lprim).
        ii. Let rstr be ? ToString(rp
       iii. Return the string-conca
     d. Set lval to lprim.
     e. Set rval to rprim.
2. NOTE: At this point, it must be a numeric operation.
3. Let lnum be ? ToNumeric(lval).
4. Let rnum be ? ToNumeric(rval).
5. If Type(lnum) is not Type(rnum), throw a TypeError exception.
...
```

PLRG

Language Specification (ECMA-262) of JavaScript

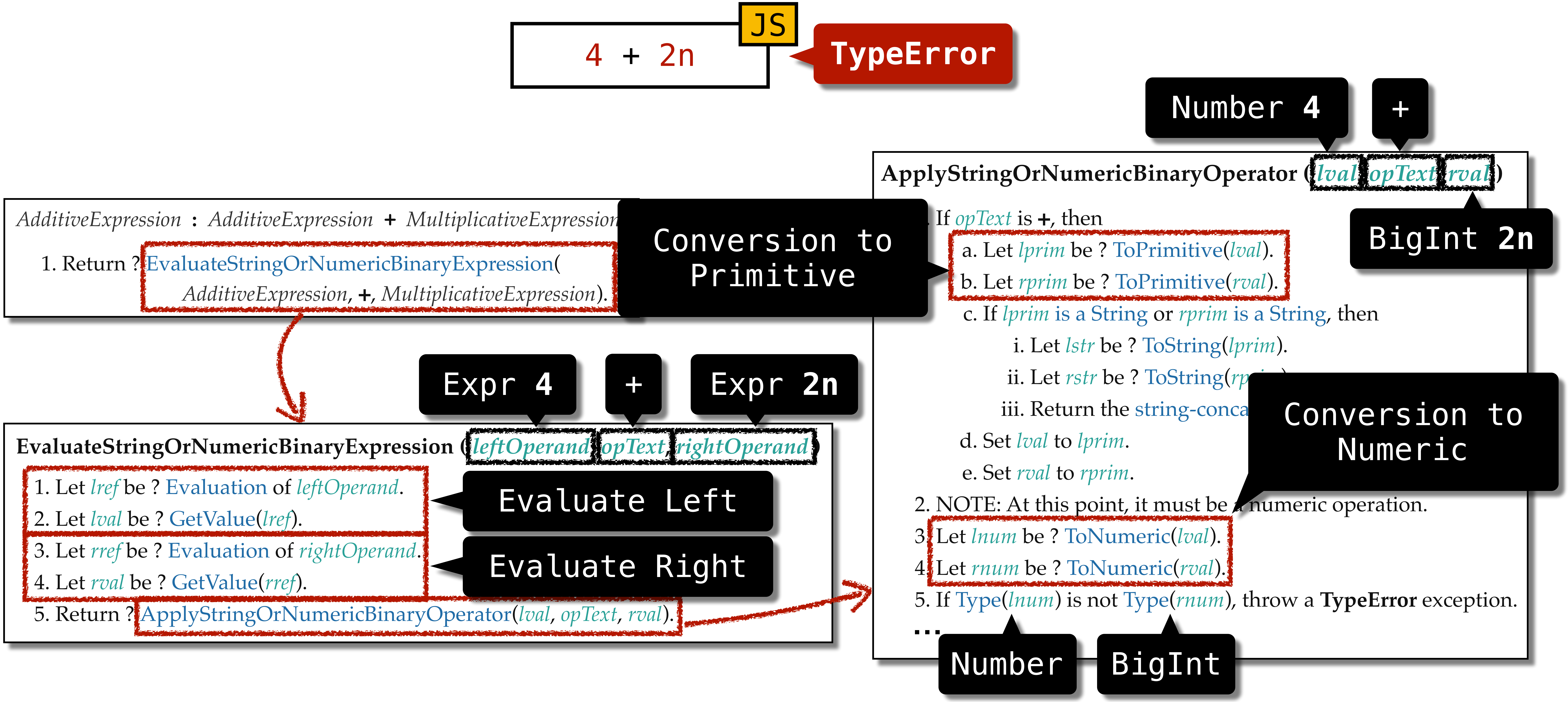


```
AdditiveExpression : AdditiveExpression + MultiplicativeExpression
    1. Return ? EvaluateStringOrNumericBinaryExpression(
           AdditiveExpression, +, MultiplicativeExpression).
```

```
EvaluateStringOrNumericBinaryExpression (leftOperand, opText, rightOperand)
    1. Let lref be ? Evaluation of leftOperand.
    2. Let lval be ? GetValue(lref).
    3. Let rref be ? Evaluation of rightOperand.
    4. Let rval be ? GetValue(rref).
    5. Return ? ApplyStringOrNumericBinaryOperator(lval, opText, rval).
```

```
ApplyStringOrNumericBinaryOperator (lval, opText, rval)
    If opText is +, then
        a. Let lprim be ? ToPrimitive(lval).
        b. Let rprim be ? ToPrimitive(rval).
        c. If lprim is a String or rprim is a String, then
            i. Let lstr be ? ToString(lprim).
            ii. Let rstr be ? ToString(rp
            iii. Return the string-conca
        d. Set lval to lprim.
        e. Set rval to rprim.
    2. NOTE: At this point, it must be a numeric operation.
    3. Let lnum be ? ToNumeric(lval).
    4. Let rnum be ? ToNumeric(rval).
    5. If Type(lnum) is not Type(rnum), throw a TypeError exception.
    ...
```

Language Specification (ECMA-262) of JavaScript



```
4 + 2n
```

AdditiveExpression : *AdditiveExpression* **+** *MultiplicativeExpression*

```
1. Return ? EvaluateStringOrNumericBinaryExpression(
       AdditiveExpression, +, MultiplicativeExpression).
```

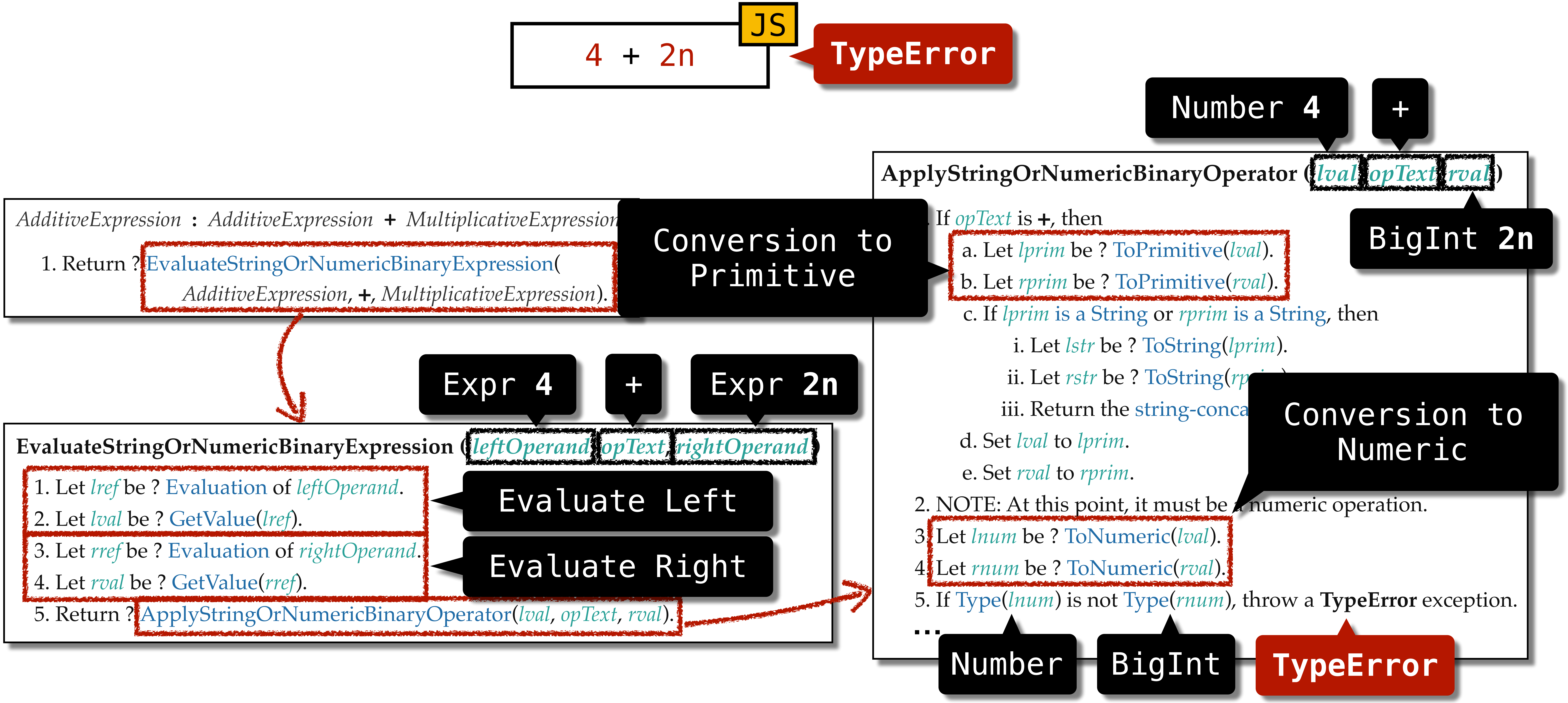
EvaluateStringOrNumericBinaryExpression (*leftOperand*, *opText*, *rightOperand*)

```
1. Let lref be ? Evaluation of leftOperand.
2. Let lval be ? GetValue(lref).
3. Let rref be ? Evaluation of rightOperand.
4. Let rval be ? GetValue(rref).
5. Return ? ApplyStringOrNumericBinaryOperator(lval, opText, rval).
```

ApplyStringOrNumericBinaryOperator (*lval*, *opText*, *rval*)

```
If opText is +, then
   a. Let lprim be ? ToPrimitive(lval).
   b. Let rprim be ? ToPrimitive(rval).
   c. If lprim is a String or rprim is a String, then
         i. Let lstr be ? ToString(lprim).
        ii. Let rstr be ? ToString(rp
       iii. Return the string-conca
   d. Set lval to lprim.
   e. Set rval to rprim.
2. NOTE: At this point, it must be a numeric operation.
3. Let lnum be ? ToNumeric(lval).
4. Let rnum be ? ToNumeric(rval).
5. If Type(lnum) is not Type(rnum), throw a TypeError exception.
...
```

Language Specification (ECMA-262) of JavaScript



JS

```
4 + 2n
```

TypeError

AdditiveExpression : *AdditiveExpression* **+** *MultiplicativeExpression*

```
1. Return ? EvaluateStringOrNumericBinaryExpression(
        AdditiveExpression, +, MultiplicativeExpression).
```

Expr 4 | + | Expr 2n

EvaluateStringOrNumericBinaryExpression (*leftOperand*, *opText*, *rightOperand*)

```
1. Let lref be ? Evaluation of leftOperand.
2. Let lval be ? GetValue(lref).
3. Let rref be ? Evaluation of rightOperand.
4. Let rval be ? GetValue(rref).
5. Return ? ApplyStringOrNumericBinaryOperator(lval, opText, rval).
```

Evaluate Left

Evaluate Right

Number 4 | + | BigInt 2n

ApplyStringOrNumericBinaryOperator (*lval*, *opText*, *rval*)

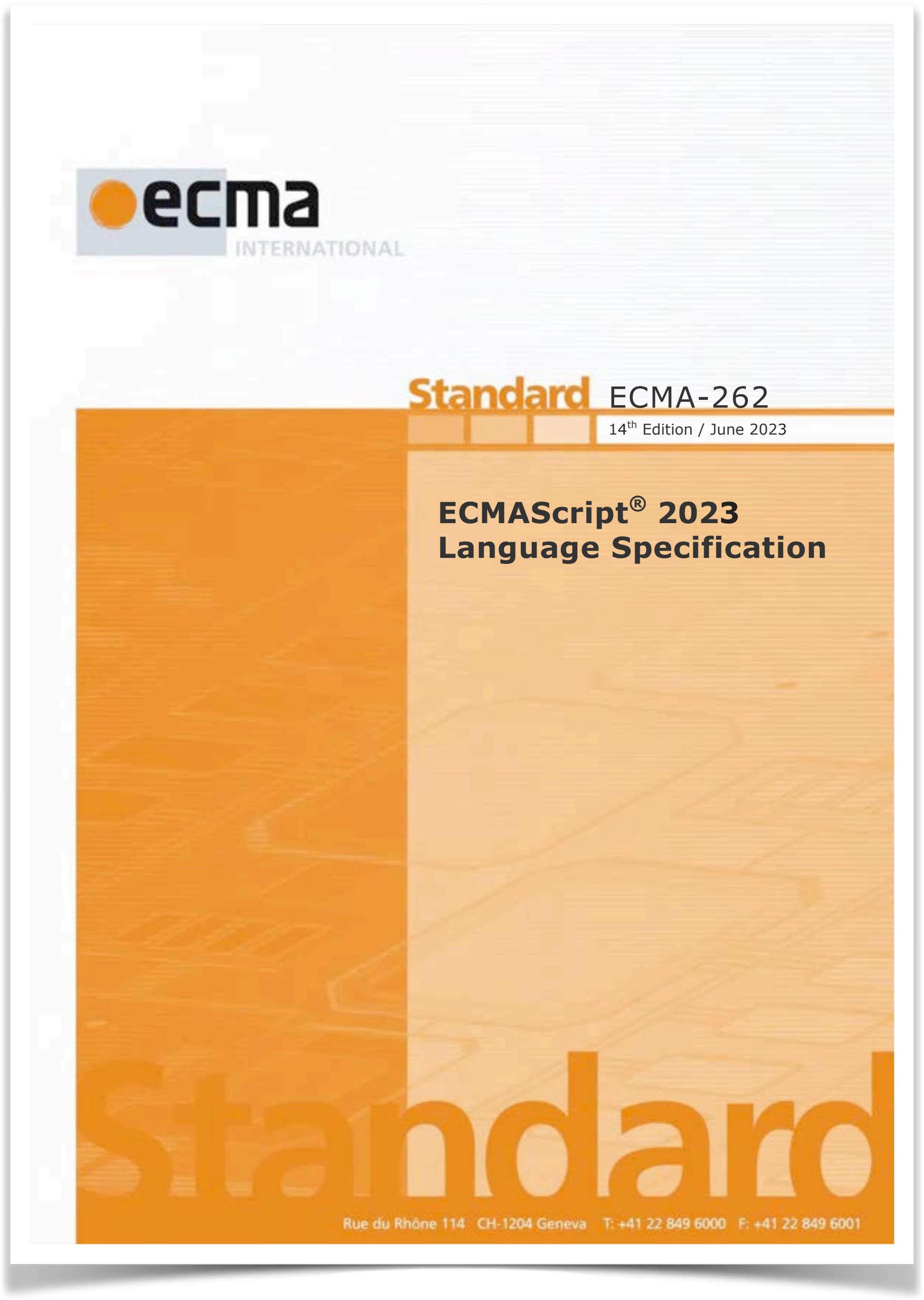
```
If opText is +, then
    a. Let lprim be ? ToPrimitive(lval).
    b. Let rprim be ? ToPrimitive(rval).
    c. If lprim is a String or rprim is a String, then
        i. Let lstr be ? ToString(lprim).
        ii. Let rstr be ? ToString(rp
        iii. Return the string-conca
    d. Set lval to lprim.
    e. Set rval to rprim.
2. NOTE: At this point, it must be a numeric operation.
3. Let lnum be ? ToNumeric(lval).
4. Let rnum be ? ToNumeric(rval).
5. If Type(lnum) is not Type(rnum), throw a TypeError exception.
...
```

Conversion to Primitive

Conversion to Numeric

Number | BigInt | TypeError

Conformance of JavaScript Engines



ecma INTERNATIONAL

Standard ECMA-262

14^{th} Edition / June 2023

ECMAScript® 2023
Language Specification

Standard

Rue du Rhône 114 CH-1204 Geneva T: +41 22 849 6000 F: +41 22 849 6001

ECMA-262
(JavaScript Spec.)

Conformance

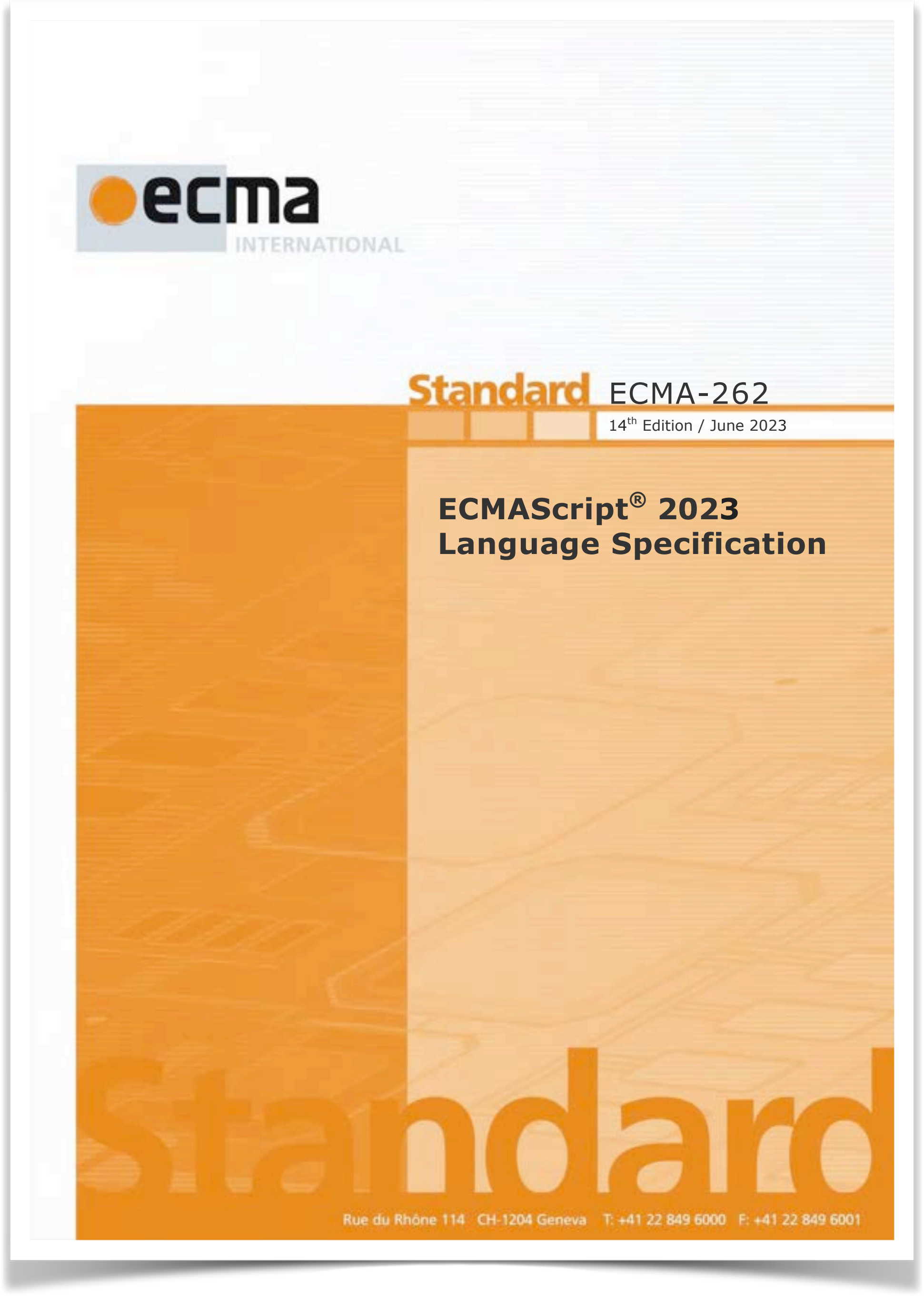
GraalVM™

QuickJS

moddable

**JavaScript
Engines**

Conformance of JavaScript Engines



ecma INTERNATIONAL

Standard ECMA-262

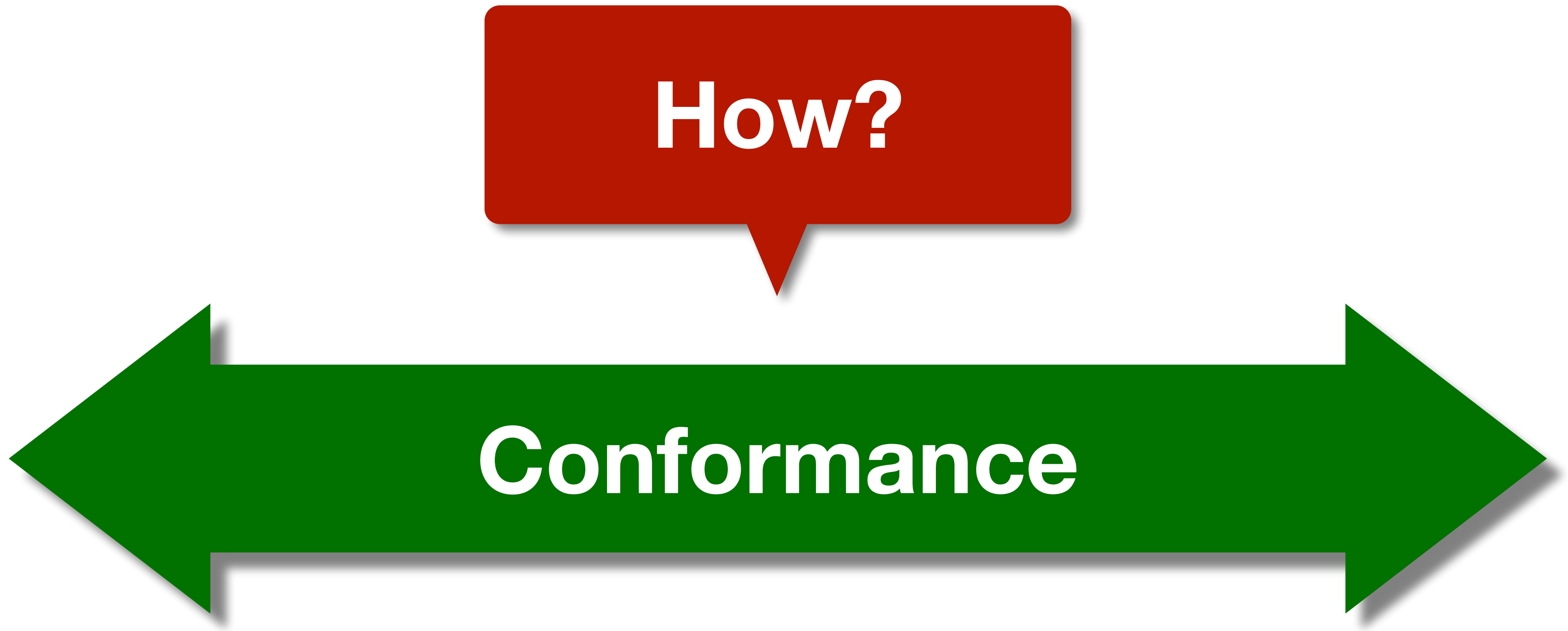
14th Edition / June 2023

ECMAScript® 2023 Language Specification

Standard

Rue du Rhône 114 CH-1204 Geneva T: +41 22 849 6000 F: +41 22 849 6001

ECMA-262
(JavaScript Spec.)



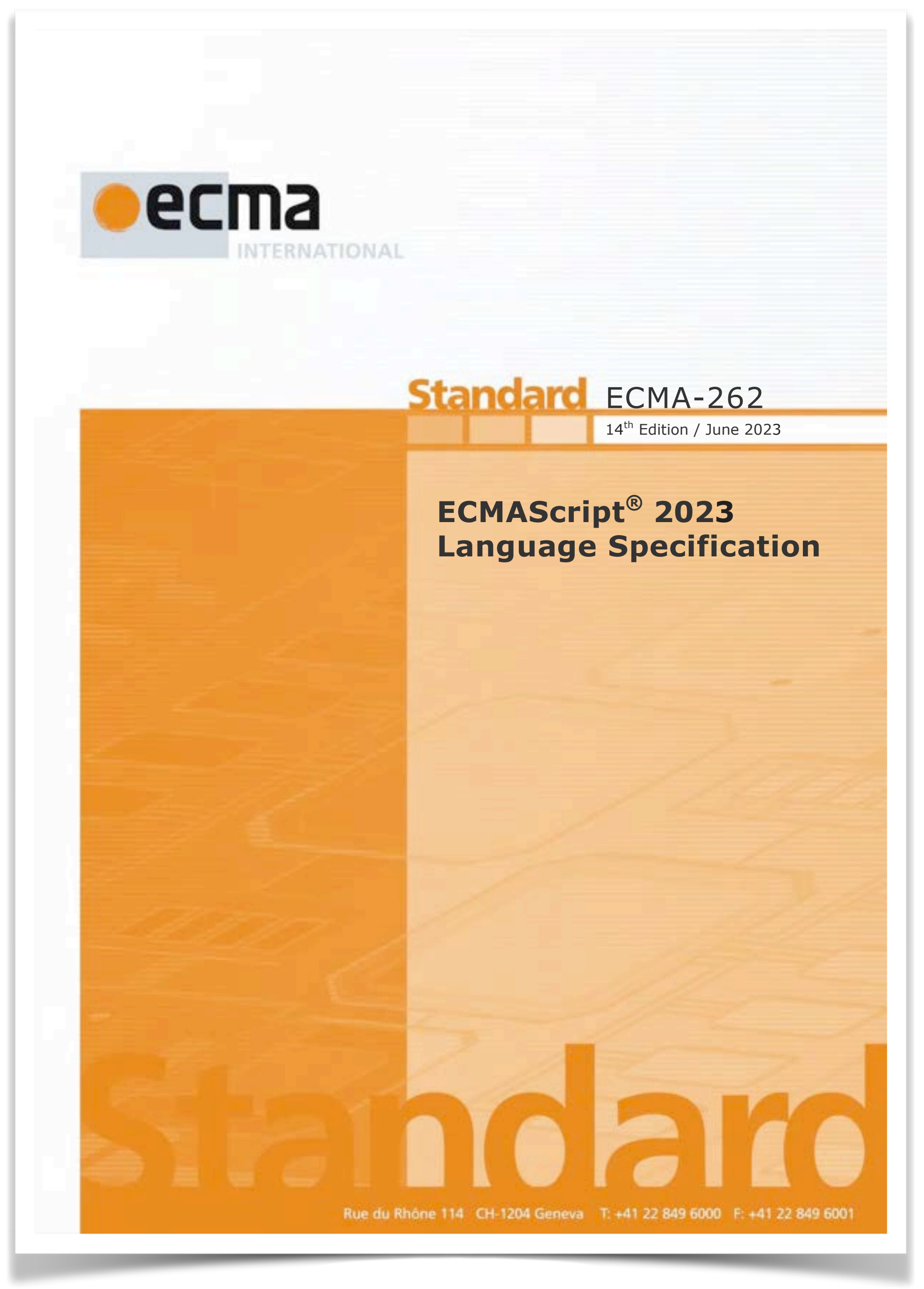
GraalVM™

QuickJS

moddable

JavaScript Engines

Conformance of JavaScript Engines



ecma INTERNATIONAL

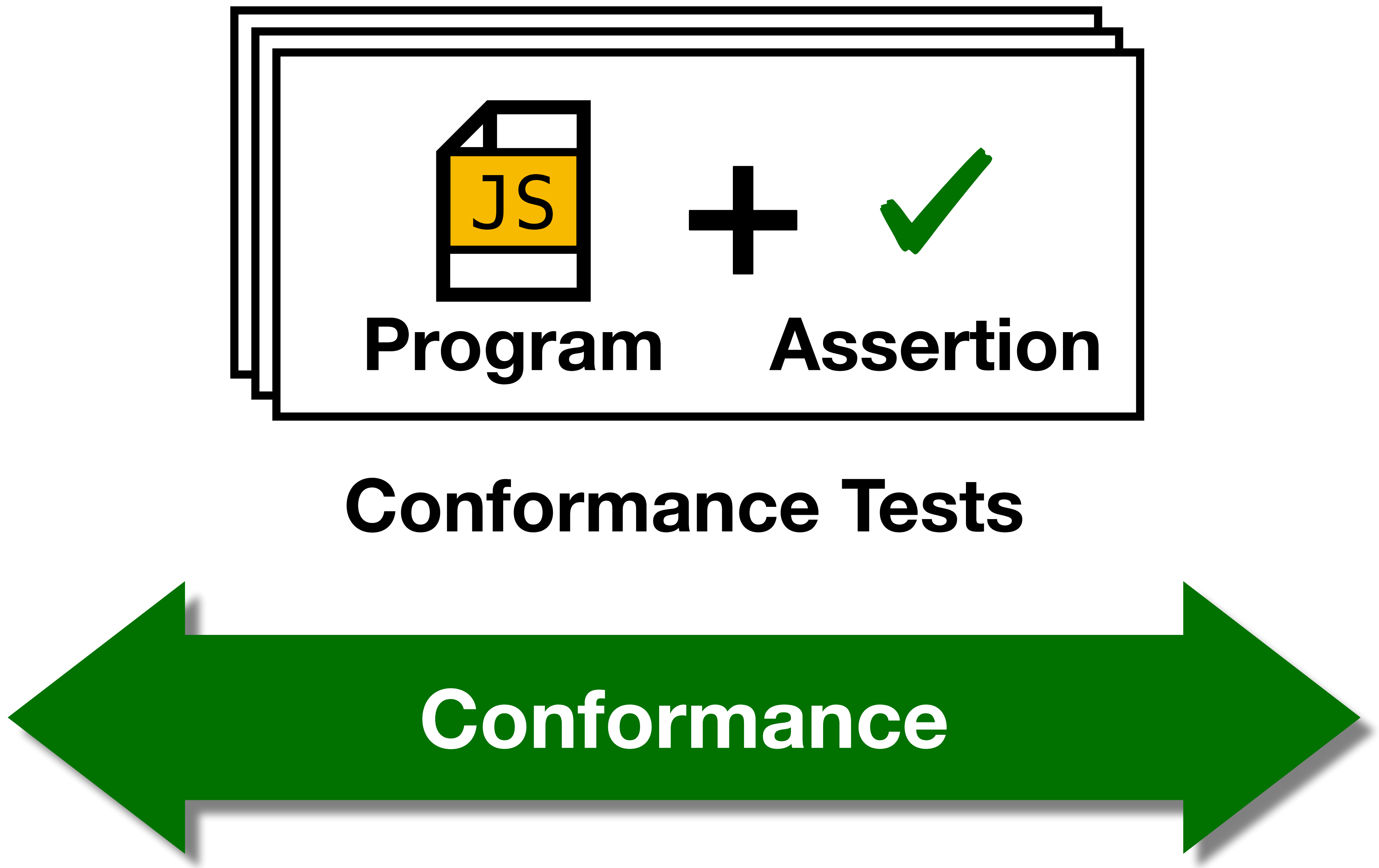
Standard ECMA-262
14th Edition / June 2023

ECMAScript® 2023
Language Specification

Standard

Rue du Rhône 114 CH-1204 Geneva T: +41 22 849 6000 F: +41 22 849 6001

ECMA-262
(JavaScript Spec.)



Test262
(Official Test Suite)

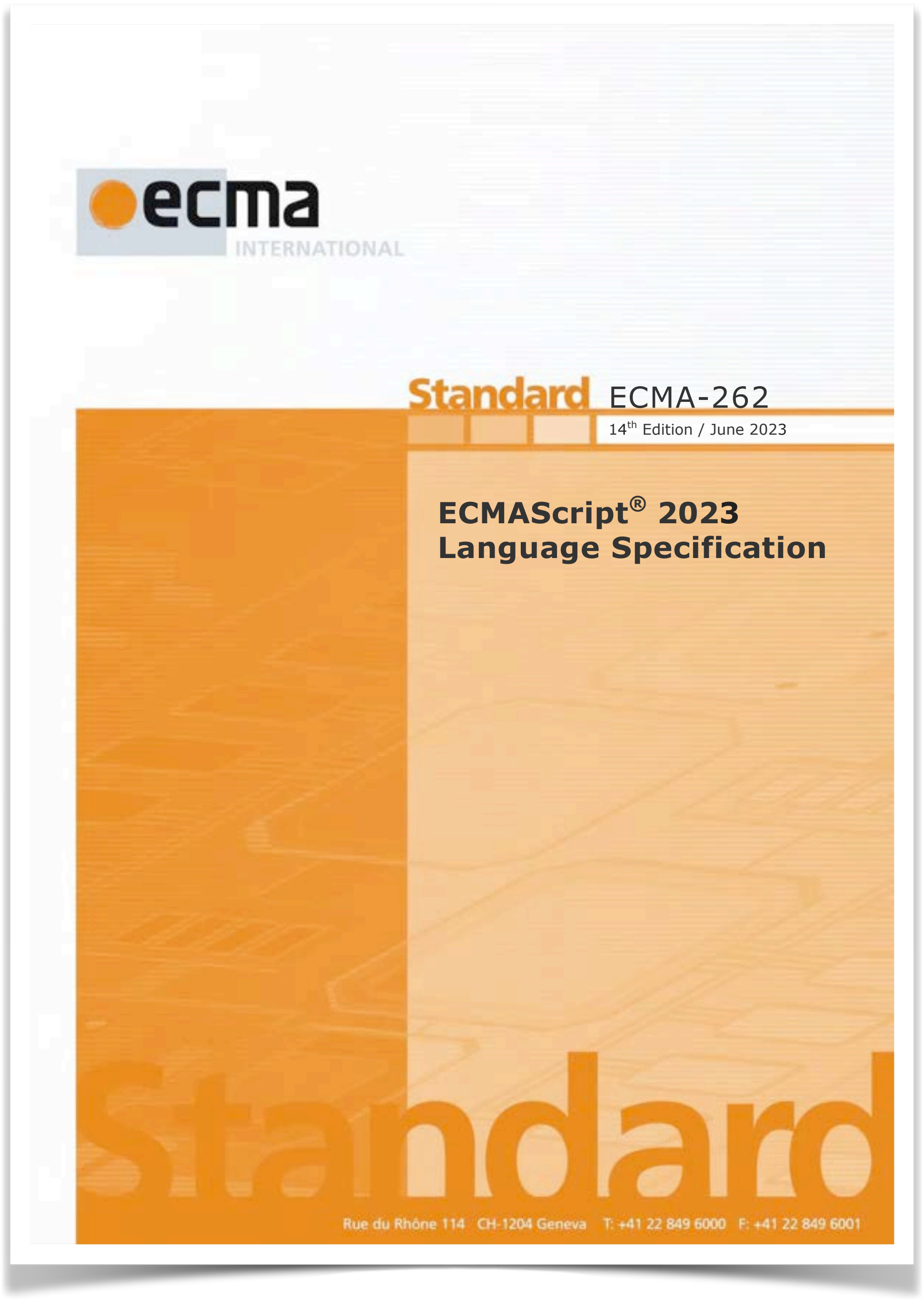
GraalVM™

QuickJS

moddable

JavaScript
Engines

Conformance of JavaScript Engines



ecma INTERNATIONAL

Standard ECMA-262

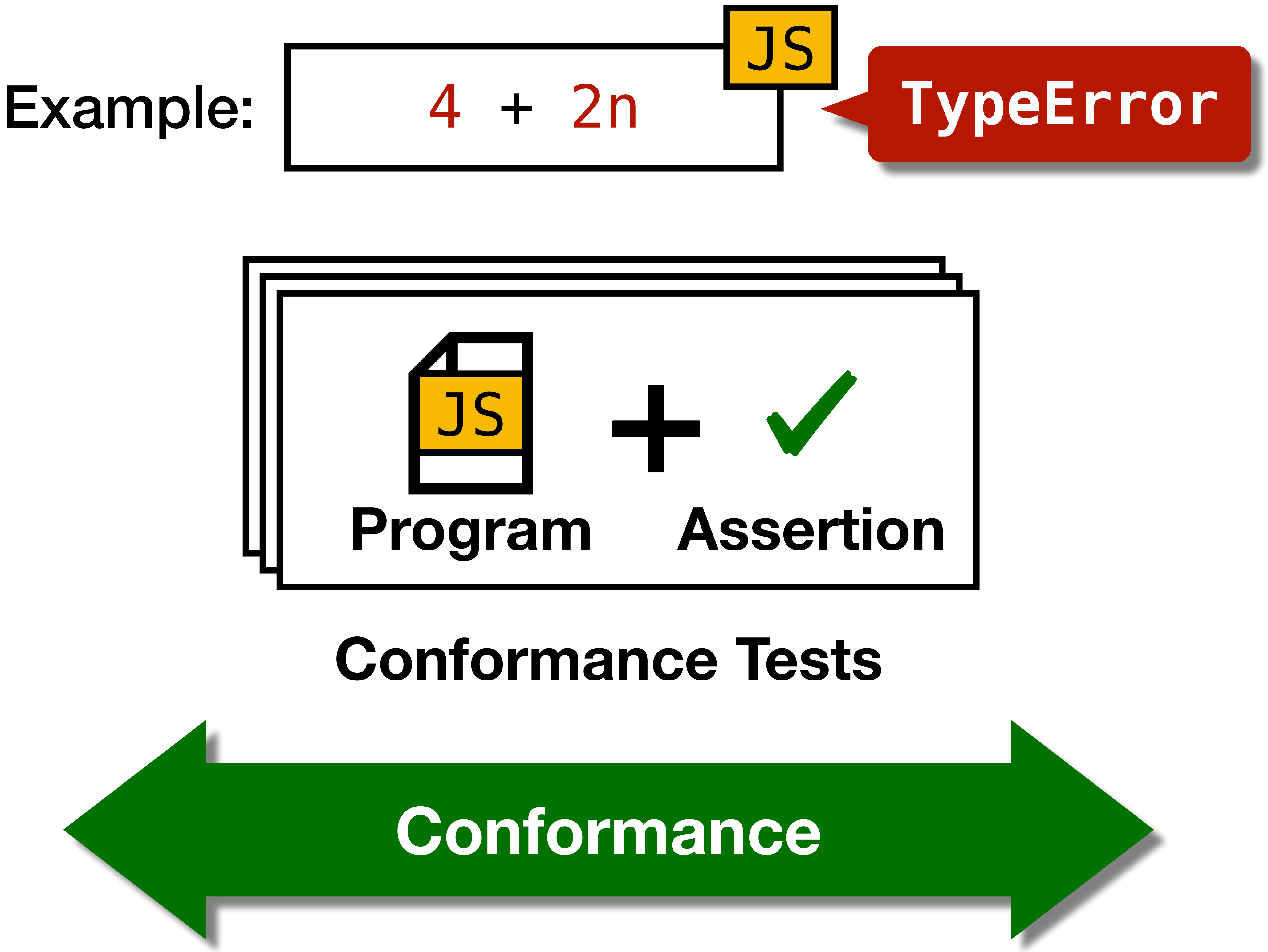
14th Edition / June 2023

ECMAScript® 2023
Language Specification

Standard

Rue du Rhône 114 CH-1204 Geneva T: +41 22 849 6000 F: +41 22 849 6001

ECMA-262
(JavaScript Spec.)



Example:

```
4 + 2n
```

Test262
(Official Test Suite)

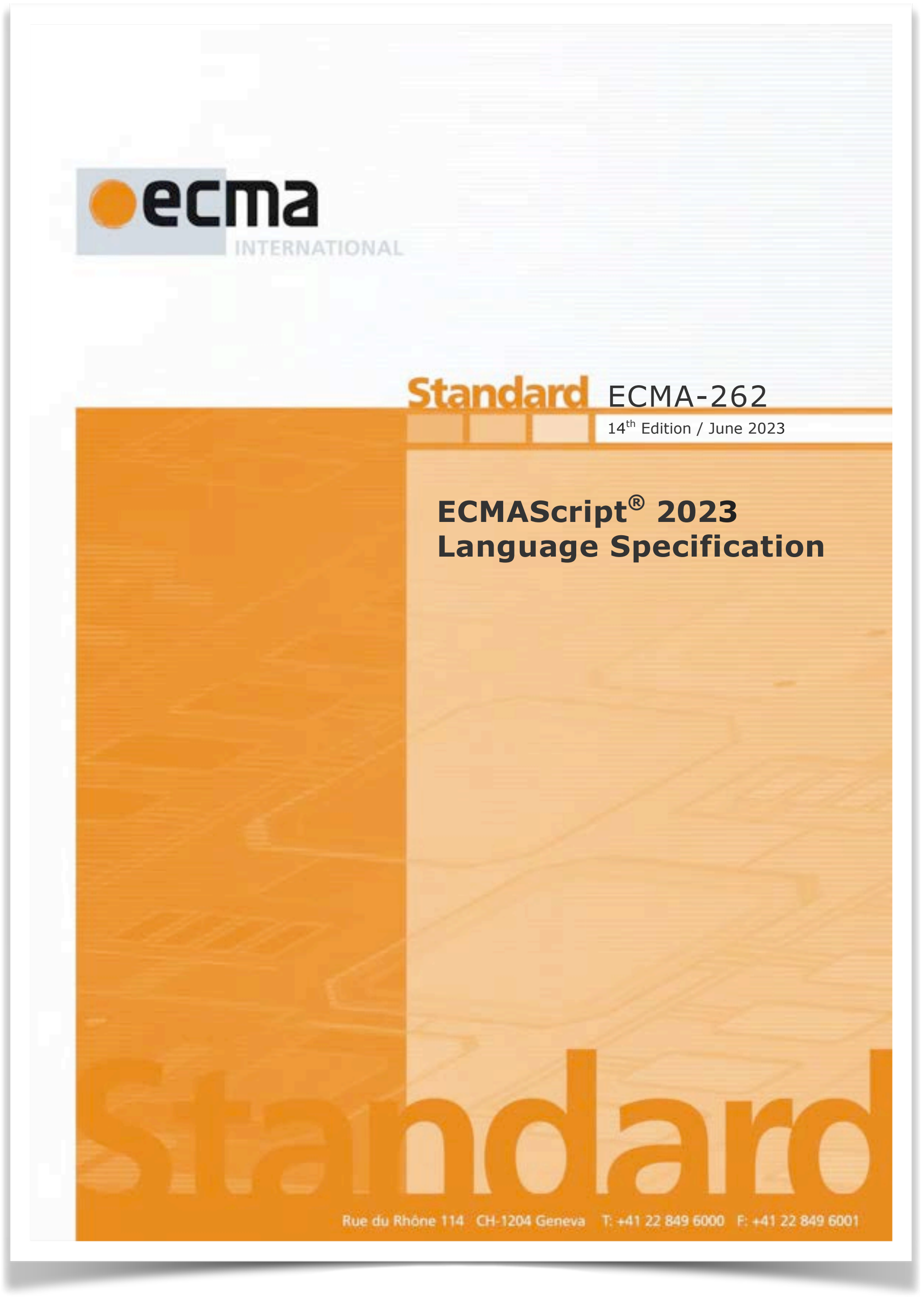
GraalVM

QuickJS

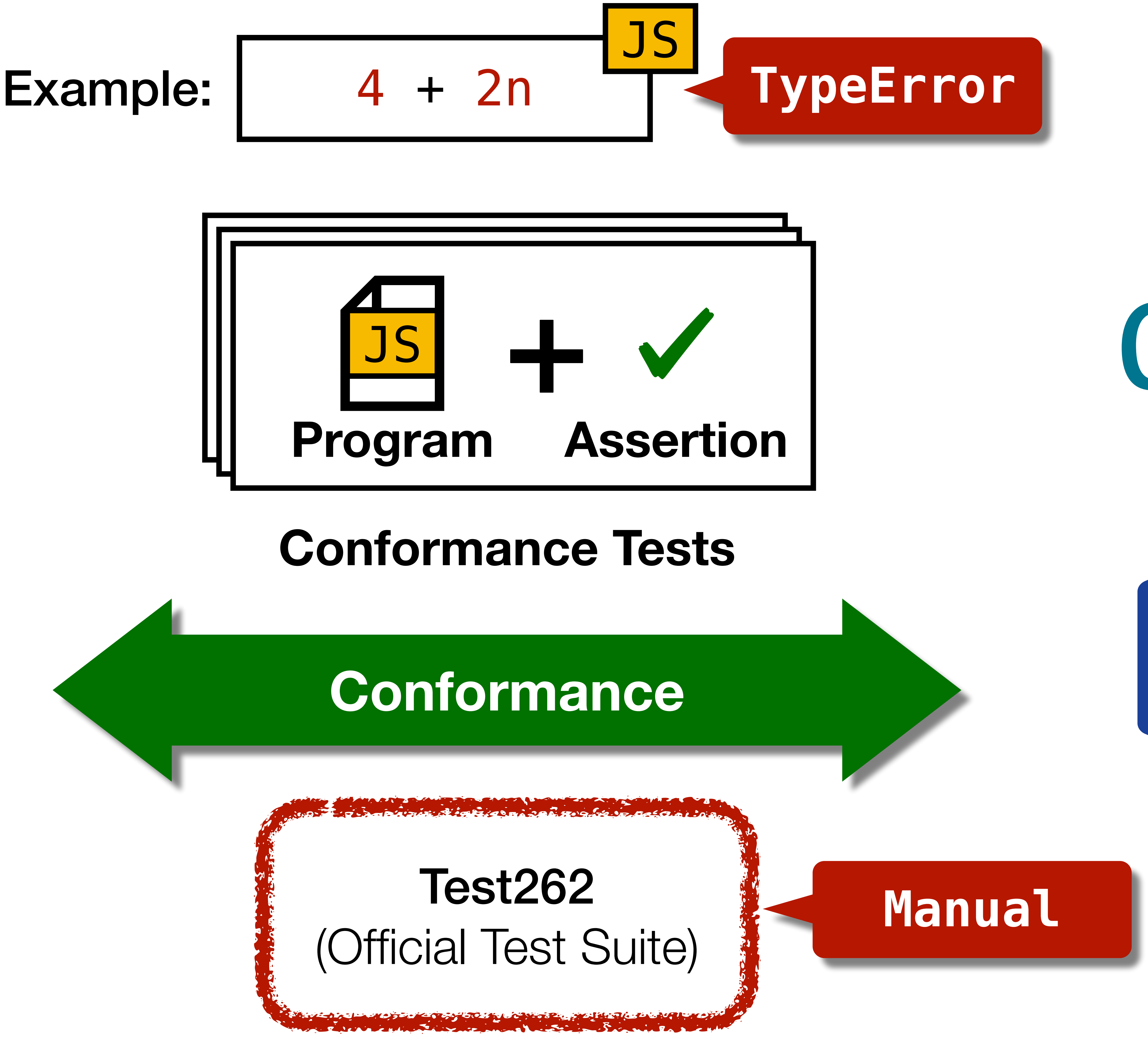
moddable

JavaScript
Engines

Problem - Manual Approach



ECMA-262
(JavaScript Spec.)



```
4 + 2n
```

Conformance Tests

Test262
(Official Test Suite)

Manual

GraalVM™

QuickJS

moddable

JavaScript
Engines

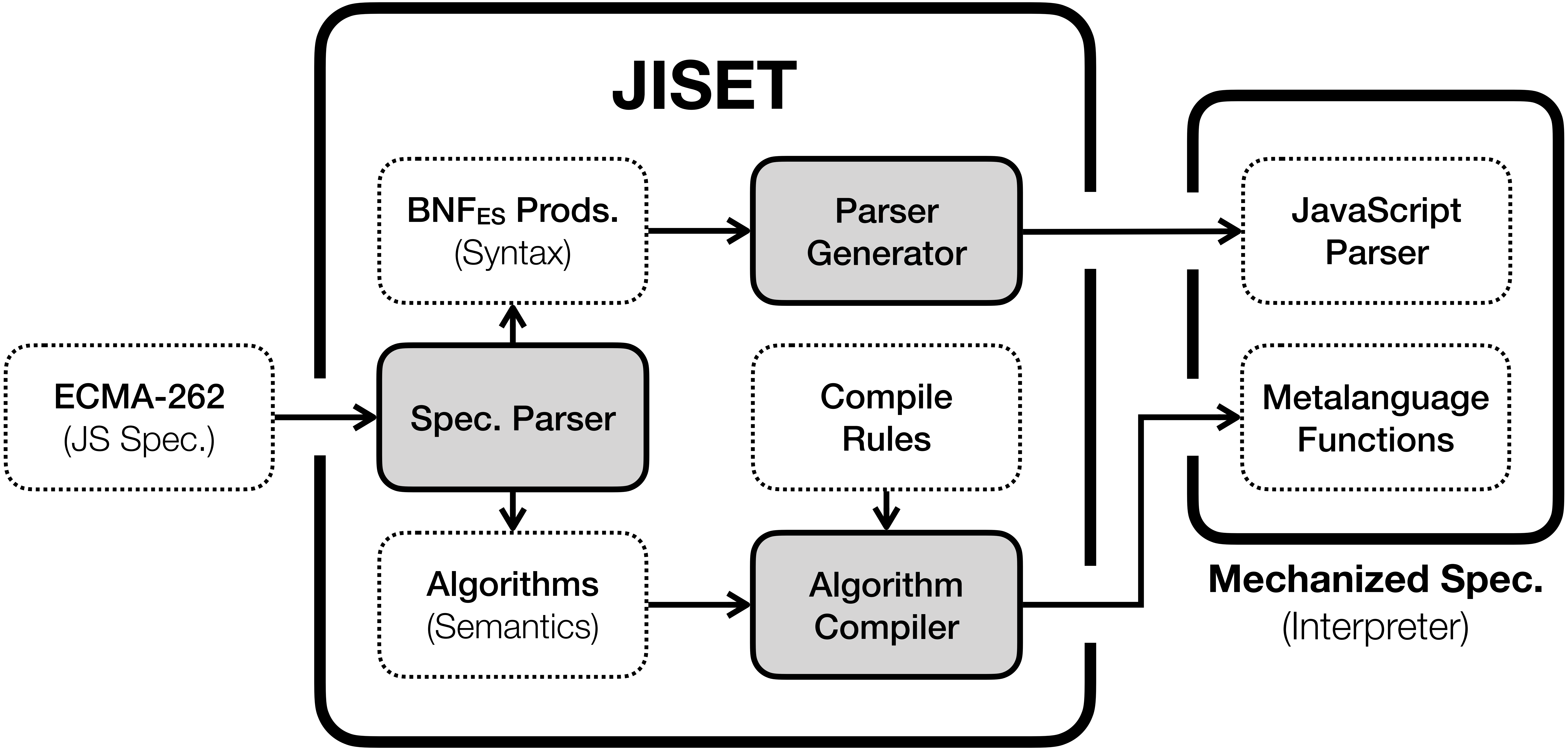
PLDI 2023
Feature-Sens. Coverage
ASE 2020
ICSE 2021
ECMA-262 (JS Spec.)
Mech. Spec. Extraction
Mechanized Spec.
Conform. Test Synthesis
Conformance Tests
JavaScript Engines

[ASE'20] J. Park, J. Park, S. An, and S. Ryu, **JISET: JavaScript IR-based Semantics Extraction Toolchain**

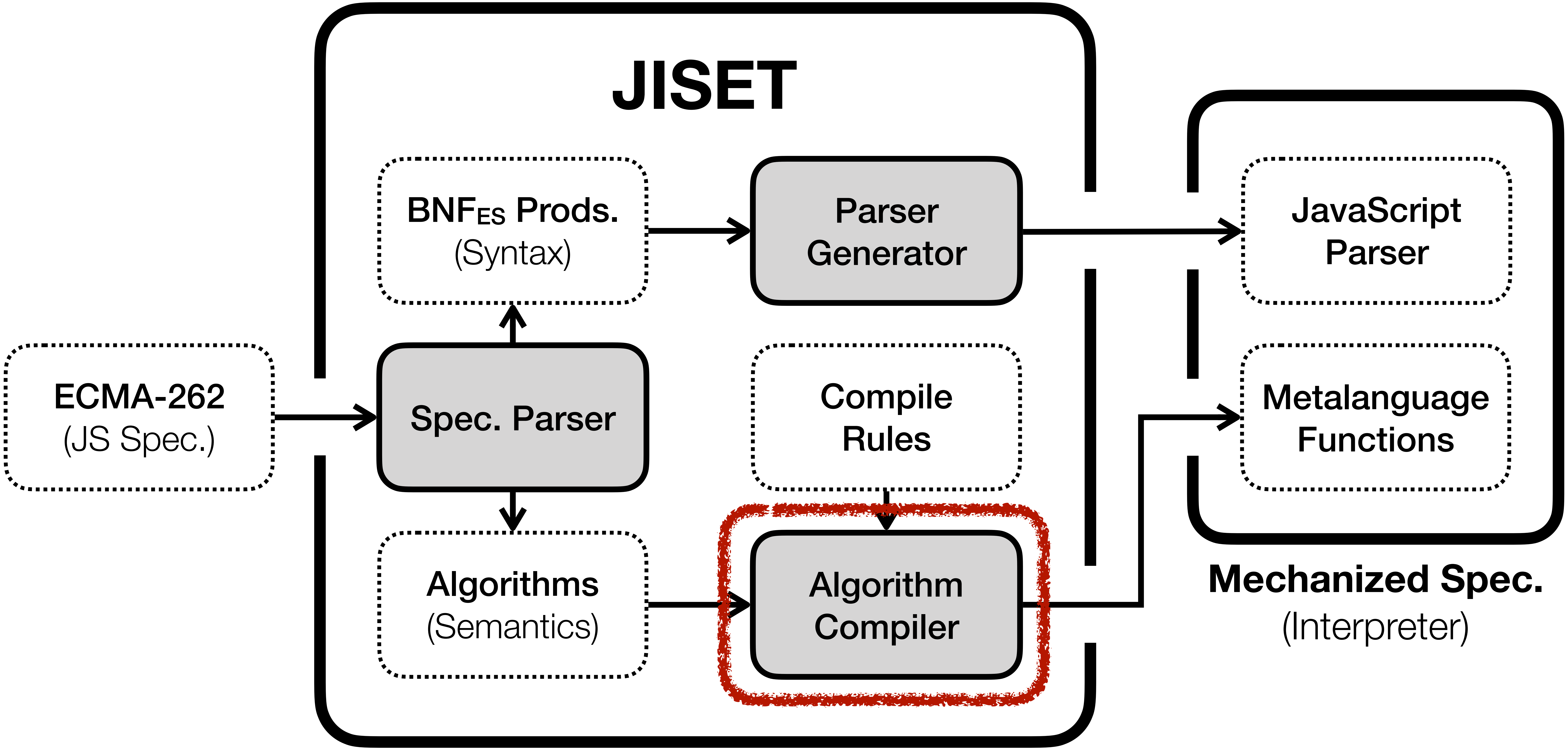
PLDI 2023
Feature-Sens. Coverage
ASE 2020
ICSE 2021
ECMA-262 (JS Spec.)
Mech. Spec. Extraction
Mechanized Spec.
Conform. Test Synthesis
Conformance Tests
JavaScript Engines

[ASE'20] J. Park, J. Park, S. An, and S. Ryu, **JISET: JavaScript IR-based Semantics Extraction Toolchain**

JISET (JavaScript IR-based Semantics Extraction Toolchain)



JISET (**J**avaScript **I**R-based **S**emantics **E**xtraction **T**oolchain)



Mechanized Spec.
(Interpreter)

JISET - Metalanguage for **Spec.** (ECMA-262)

$$
\begin{array}{lll}
\text{Programs} & \mathfrak{P} \ni P & ::= f^* \\
\text{Functions} & \mathcal{F} \ni f & ::= \mathtt{syntax}^? \; \mathtt{def} \; \mathtt{x}(\mathtt{x}^*) \; \{[\ell : i]^*\} \\
\text{Variables} & \mathcal{X} \ni \mathtt{x} & \\
\text{Labels} & \mathcal{L} \ni \ell & \\
\text{Instructions} & \mathcal{I} \ni i & ::= r := e \mid \mathtt{x} := \{\} \mid \mathtt{x} := e(e^*) \\
 & & \quad \mid \; \mathtt{if} \; e \; \ell \; \ell \mid \mathtt{return} \; e \\
\text{Expressions} & \mathcal{E} \ni e & ::= v^{\mathsf{p}} \mid \mathtt{op}(e^*) \mid r \\
\text{References} & \mathcal{R} \ni r & ::= \mathtt{x} \mid e[e] \mid e[e]_{\mathsf{js}}
\end{array}
$$

• • •

$$
\begin{array}{lll}
\text{Values} & v \in \mathbb{V} & = \mathbb{A} \uplus \mathbb{V}^{\mathsf{p}} \uplus \mathbb{T} \uplus \mathcal{F} \\
\text{Primitive Values} & v^{\mathsf{p}} \in \mathbb{V}^{\mathsf{p}} & = \mathbb{V}_{\mathsf{bool}} \uplus \mathbb{V}_{\mathsf{int}} \uplus \mathbb{V}_{\mathsf{str}} \uplus \cdots \\
\text{JS ASTs} & t \in \mathbb{T} &
\end{array}
$$

• • •

JISET - Metalanguage for Spec. (ECMA-262)

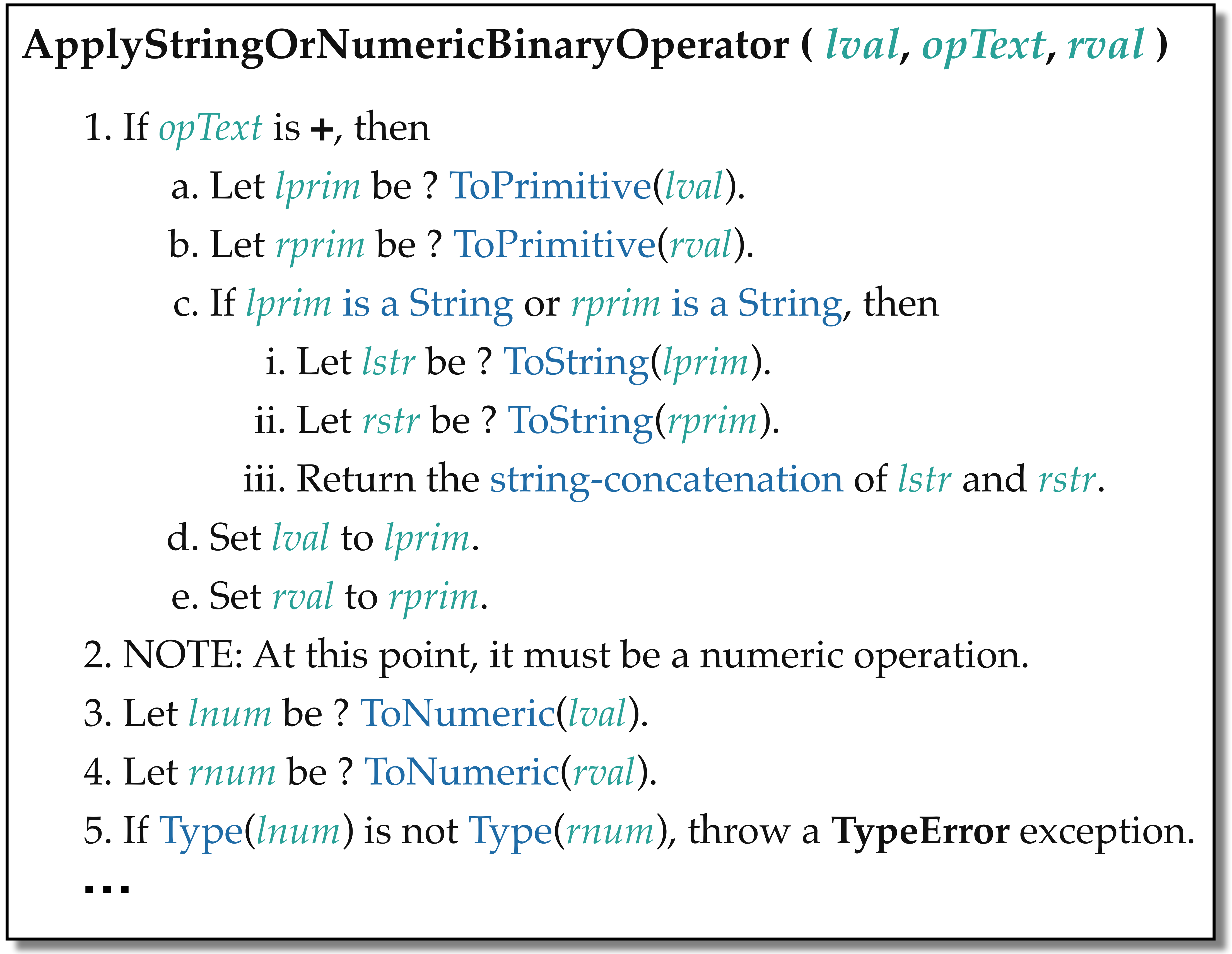
Programs	$\mathfrak{P} \ni P ::= f^*$
Functions	$\mathcal{F} \ni f ::= \texttt{syntax}^? \ \texttt{def} \ \texttt{x}(\texttt{x}^*) \ \{[\ell : i]^*\}$
Variables	$\mathcal{X} \ni \texttt{x}$
Labels	$\mathcal{L} \ni \ell$
Instructions	$\mathcal{I} \ni i ::= r := e \mid \texttt{x} := \{\} \mid \texttt{x} := e(e^*)$ $\mid \ \texttt{if} \ e \ \ell \ \ell \mid \texttt{return} \ e$
Expressions	$\mathcal{E} \ni e ::= v^{\mathrm{p}} \mid \texttt{op}(e^*) \mid r$
References	$\mathcal{R} \ni r ::= \texttt{x} \mid e[e] \mid e[e]_{\mathrm{js}}$

• • •

Values	$v \in \mathbb{V}$	$= \mathbb{A} \uplus \mathbb{V}^{\mathrm{p}} \uplus \mathbb{T} \uplus \mathcal{F}$
Primitive Values	$v^{\mathrm{p}} \in \mathbb{V}^{\mathrm{p}}$	$= \mathbb{V}_{\mathrm{bool}} \uplus \mathbb{V}_{\mathrm{int}} \uplus \mathbb{V}_{\mathrm{str}} \uplus \cdots$
JS ASTs	$t \in \mathbb{T}$	

• • •

JISET - Algorithm Compiler



ApplyStringOrNumericBinaryOperator (*lval*, *opText*, *rval* **)**

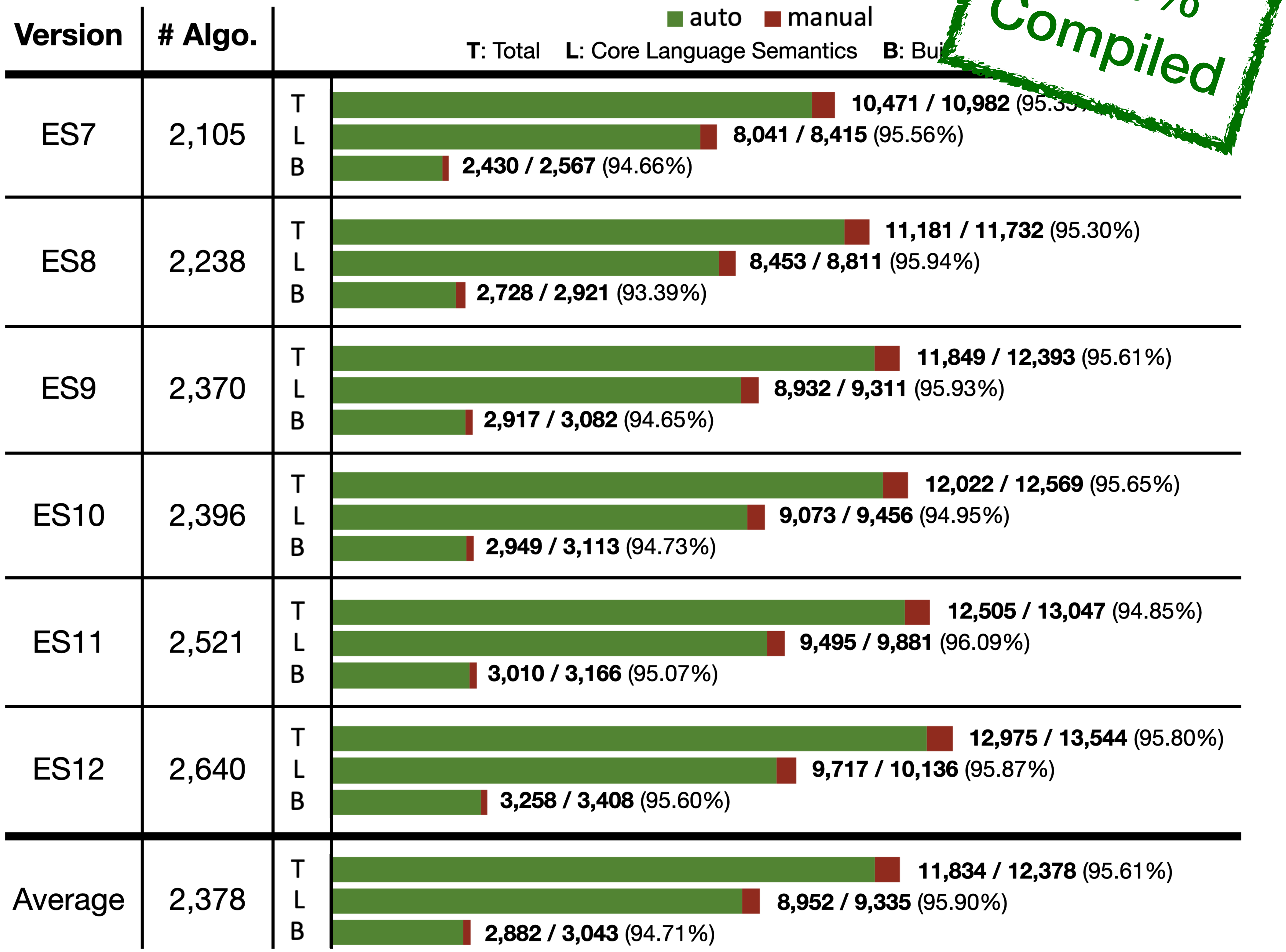
```
1. If opText is +, then
    a. Let lprim be ? ToPrimitive(lval).
    b. Let rprim be ? ToPrimitive(rval).
    c. If lprim is a String or rprim is a String, then
        i. Let lstr be ? ToString(lprim).
        ii. Let rstr be ? ToString(rprim).
        iii. Return the string-concatenation of lstr and rstr.
    d. Set lval to lprim.
    e. Set rval to rprim.
2. NOTE: At this point, it must be a numeric operation.
3. Let lnum be ? ToNumeric(lval).
4. Let rnum be ? ToNumeric(rval).
5. If Type(lnum) is not Type(rnum), throw a TypeError exception.
...
```

118 Compile Rules

```
def ApplyStringOrNumericBinaryOperator(
  lval, opText, rval
) {
  if (= opText "+") {
    let lprim = [? ToPrimitive(lval)]
    let rprim = [? ToPrimitive(rval)]
    if (|| (= (typeof lprim) @String)
           (= (typeof rprim) @String)) {
      let lstr = [? ToString(lprim)]
      let rstr = [? ToString(rprim)]
      return (concat lstr rstr)
    }
    lval = lprim
    rval = rprim
  }
  let lnum = [? ToNumeric(lval)]
  let rnum = [? ToNumeric(rval)]
  if (! (= (typeof lnum) (typeof rnum))) {
    return comp[~throw~](new TypeError)
  }
  ...
}
```

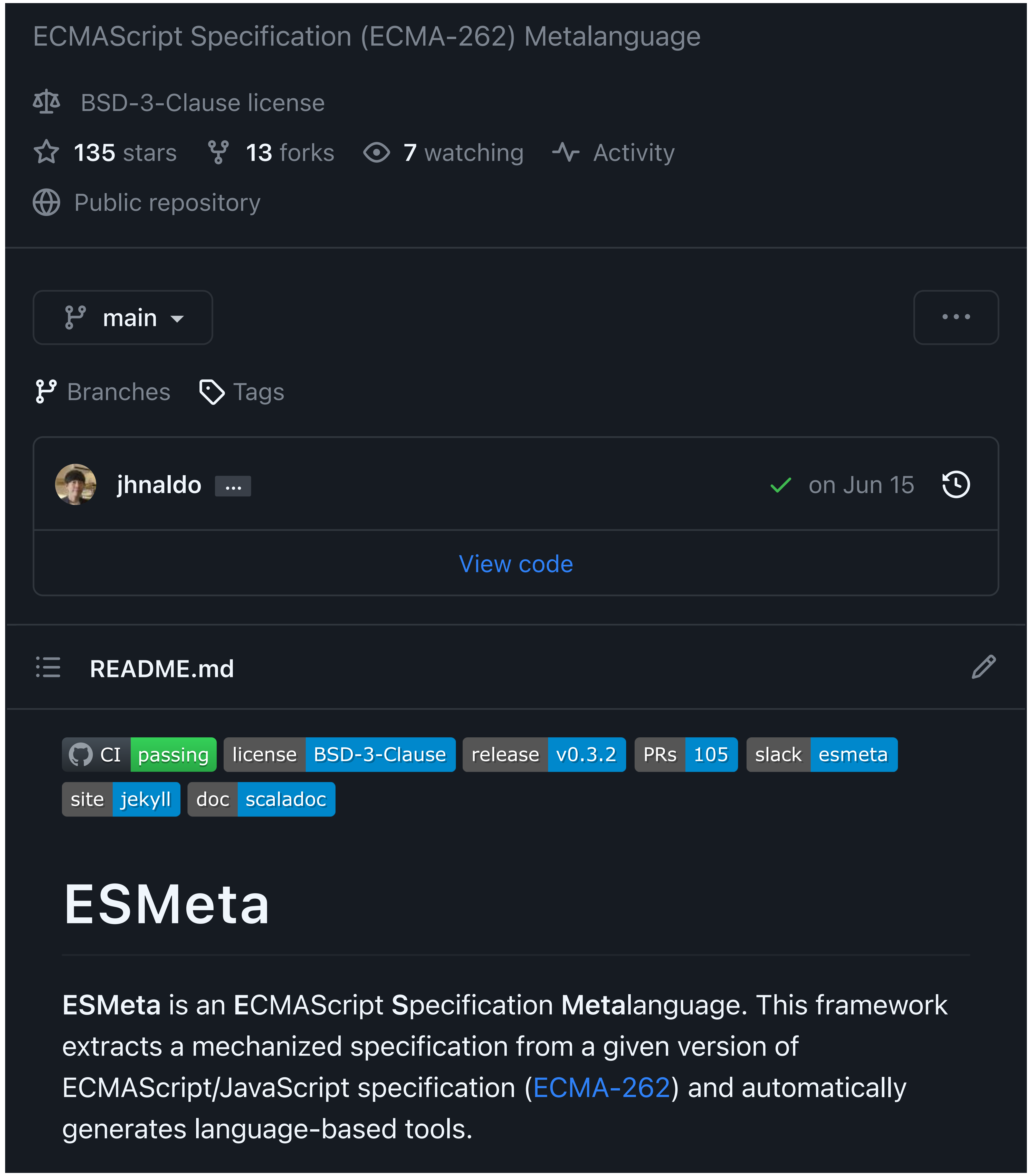
JISET - Evaluation

≈ 96% Compiled



Version	# Algo.		
ES7	2,105	T	**10,471 / 10,982** (95.35%)
		L	**8,041 / 8,415** (95.56%)
		B	**2,430 / 2,567** (94.66%)
ES8	2,238	T	**11,181 / 11,732** (95.30%)
		L	**8,453 / 8,811** (95.94%)
		B	**2,728 / 2,921** (93.39%)
ES9	2,370	T	**11,849 / 12,393** (95.61%)
		L	**8,932 / 9,311** (95.93%)
		B	**2,917 / 3,082** (94.65%)
ES10	2,396	T	**12,022 / 12,569** (95.65%)
		L	**9,073 / 9,456** (94.95%)
		B	**2,949 / 3,113** (94.73%)
ES11	2,521	T	**12,505 / 13,047** (94.85%)
		L	**9,495 / 9,881** (96.09%)
		B	**3,010 / 3,166** (95.07%)
ES12	2,640	T	**12,975 / 13,544** (95.80%)
		L	**9,717 / 10,136** (95.87%)
		B	**3,258 / 3,408** (95.60%)
Average	2,378	T	**11,834 / 12,378** (95.61%)
		L	**8,952 / 9,335** (95.90%)
		B	**2,882 / 3,043** (94.71%)

ESMeta



ECMAScript Specification (ECMA-262) Metalanguage

BSD-3-Clause license

135 stars 13 forks 7 watching Activity

Public repository

ESMeta

ESMeta is an **E**CMA**S**cript Specification **Meta**language. This framework extracts a mechanized specification from a given version of ECMAScript/JavaScript specification (ECMA-262) and automatically generates language-based tools.

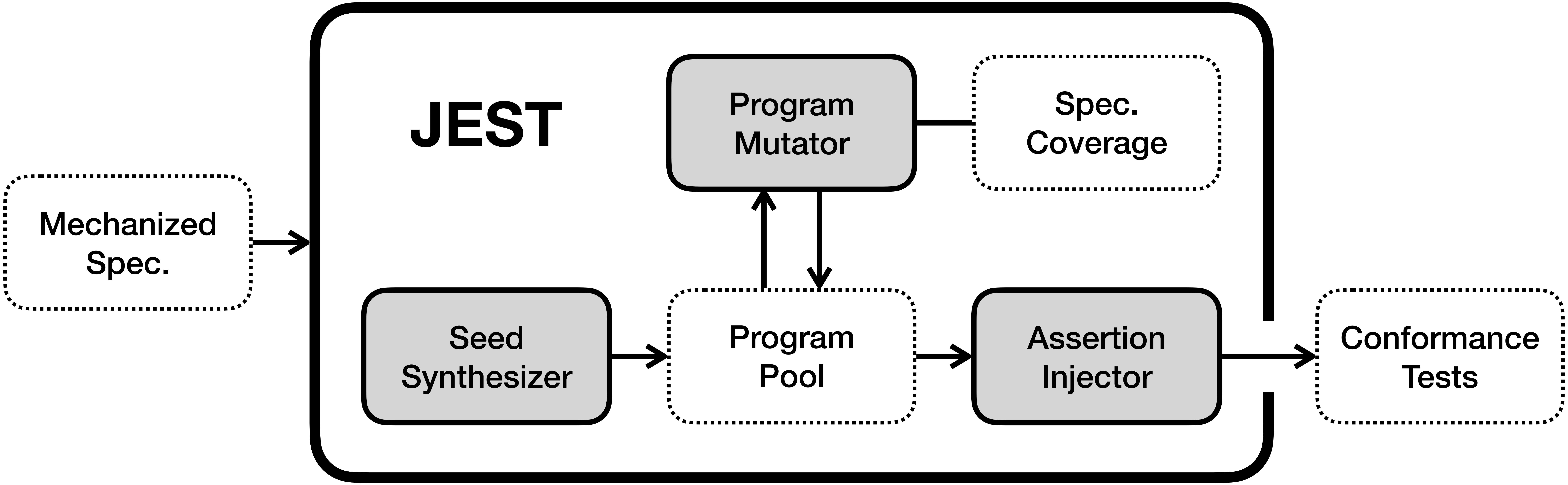
https://github.com/es-meta/esmeta

PLDI 2023
Feature-Sens. Coverage
ASE 2020
ICSE 2021
ECMA-262 (JS Spec.)
Mech. Spec. Extraction
Mechanized Spec.
Conform. Test Synthesis
Conformance Tests
JavaScript Engines

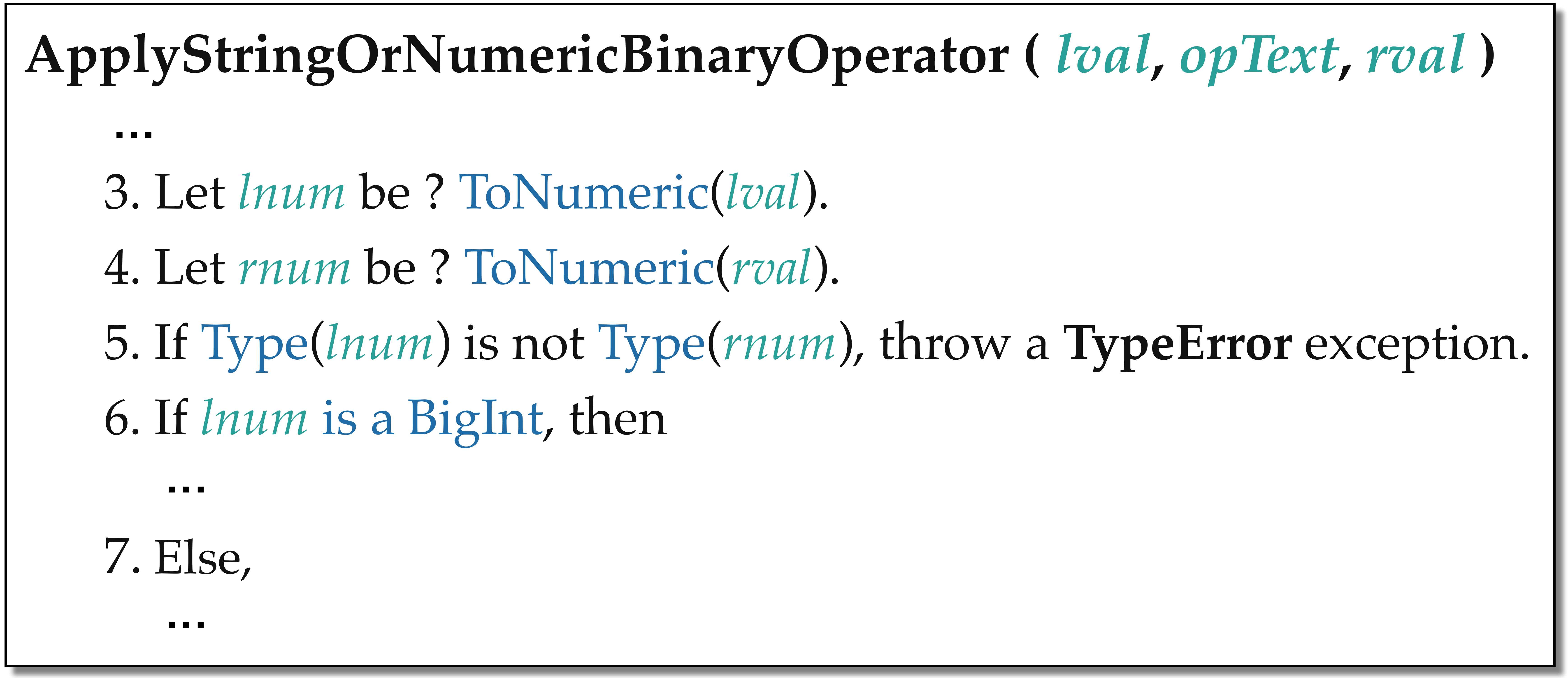
[ICSE'21] J. Park, S. An, D. Youn, G. Kim, and S. Ryu, **JEST: N+1-version Differential Testing of Both JavaScript Engines and Specification**

JEST (**J**avaScript **E**ngines and **S**pecification **T**ester)

- **Conformance Test Synthesis** using **Coverage-guided Fuzzing** in **Mechanized Spec.**

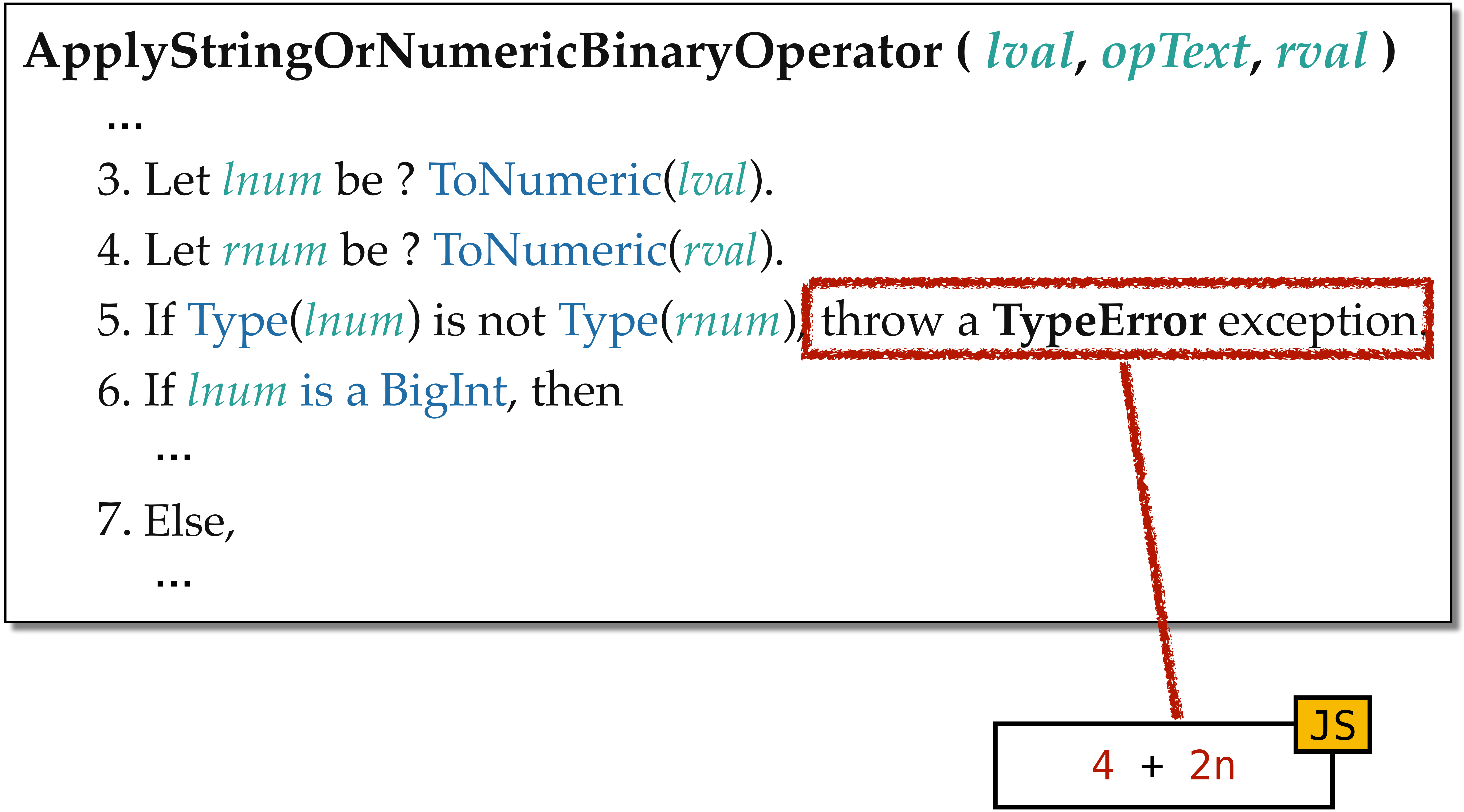


JEST - Coverage-guided Fuzzing (in Spec.)



```
ApplyStringOrNumericBinaryOperator ( lval, opText, rval )
    ...
    3. Let lnum be ? ToNumeric(lval).
    4. Let rnum be ? ToNumeric(rval).
    5. If Type(lnum) is not Type(rnum), throw a TypeError exception.
    6. If lnum is a BigInt, then
        ...
    7. Else,
        ...
```

JEST - Coverage-guided Fuzzing (in Spec.)

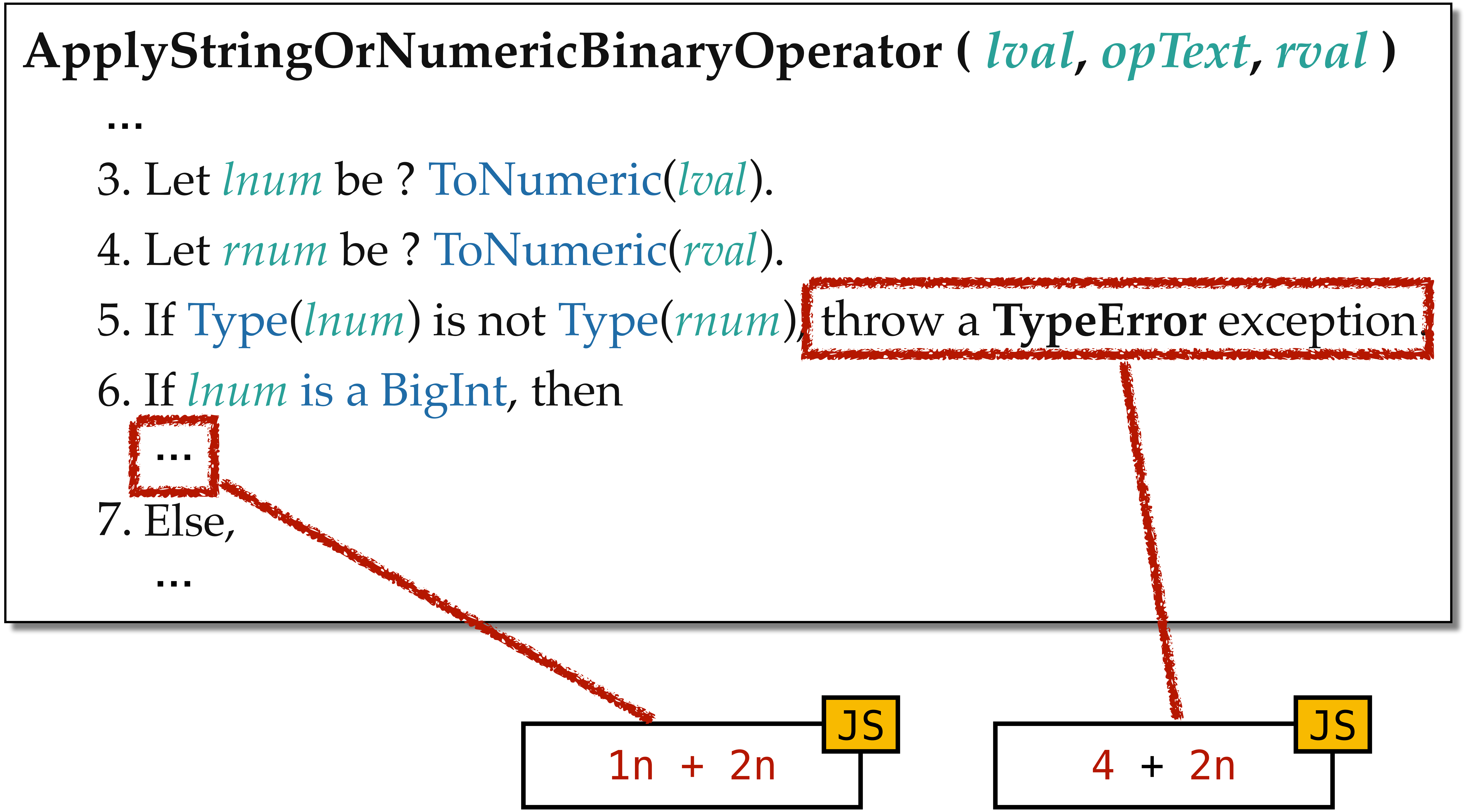


ApplyStringOrNumericBinaryOperator (*lval*, *opText*, *rval*)

```
  ...
  3. Let lnum be ? ToNumeric(lval).
  4. Let rnum be ? ToNumeric(rval).
  5. If Type(lnum) is not Type(rnum), throw a TypeError exception.
  6. If lnum is a BigInt, then
       ...
  7. Else,
       ...
```

```
4 + 2n
```

JEST - Coverage-guided Fuzzing (in Spec.)

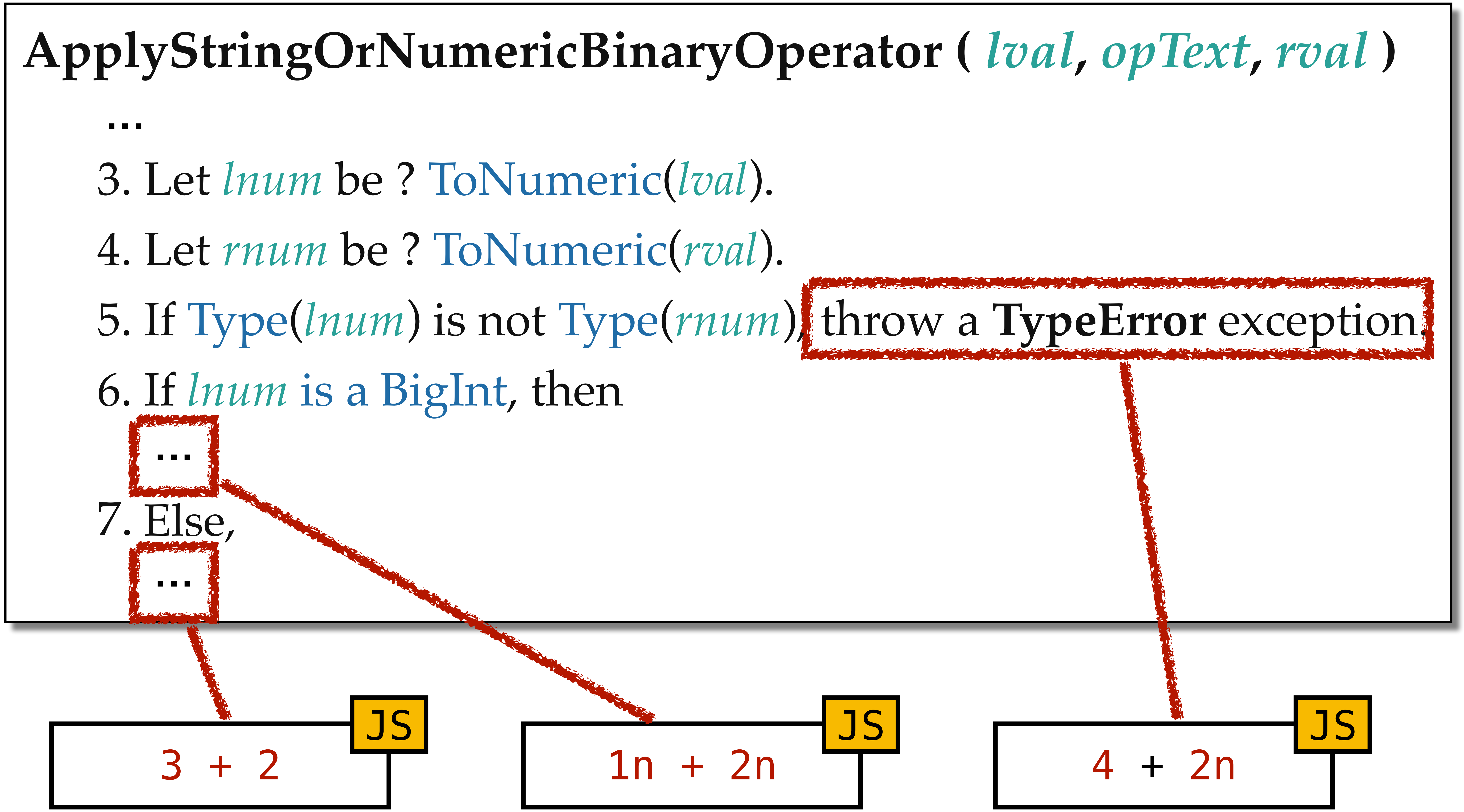


```
ApplyStringOrNumericBinaryOperator ( lval, opText, rval )
    ...
    3. Let lnum be ? ToNumeric(lval).
    4. Let rnum be ? ToNumeric(rval).
    5. If Type(lnum) is not Type(rnum), throw a TypeError exception.
    6. If lnum is a BigInt, then
        ...
    7. Else,
        ...
```

```
1n + 2n
```

```
4 + 2n
```

JEST - Coverage-guided Fuzzing (in Spec.)



ApplyStringOrNumericBinaryOperator (*lval*, *opText*, *rval*)

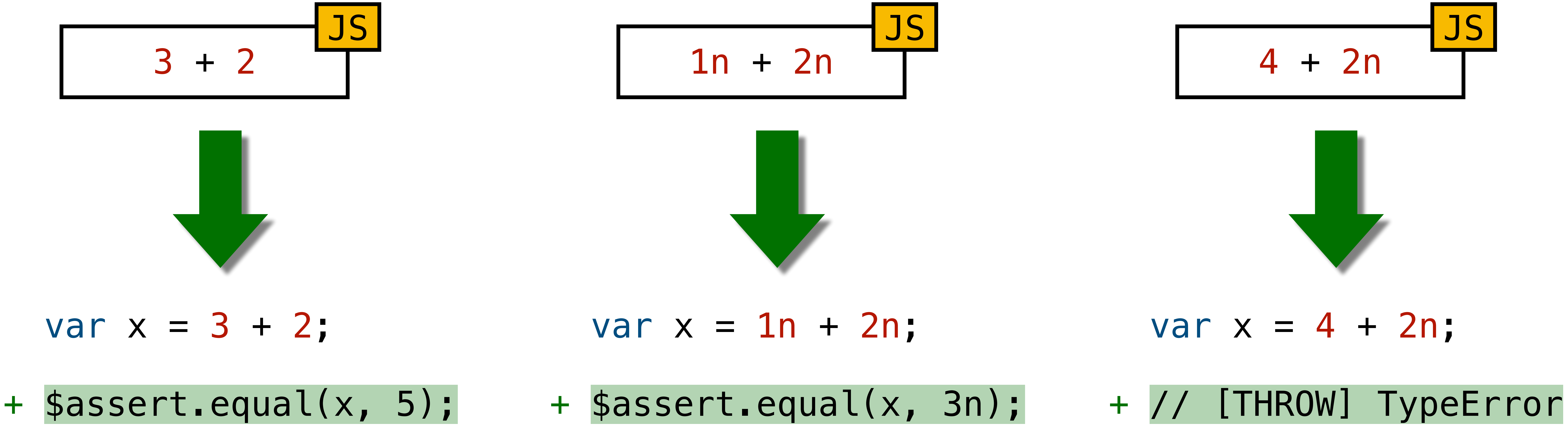
```
    …
    3. Let lnum be ? ToNumeric(lval).
    4. Let rnum be ? ToNumeric(rval).
    5. If Type(lnum) is not Type(rnum), throw a TypeError exception.
    6. If lnum is a BigInt, then
        …
    7. Else,
        …
```

```js
3 + 2
```

```js
1n + 2n
```

```js
4 + 2n
```

JEST - Assertion Injection



```
3 + 2
```

```
var x = 3 + 2;
+ $assert.equal(x, 5);
```

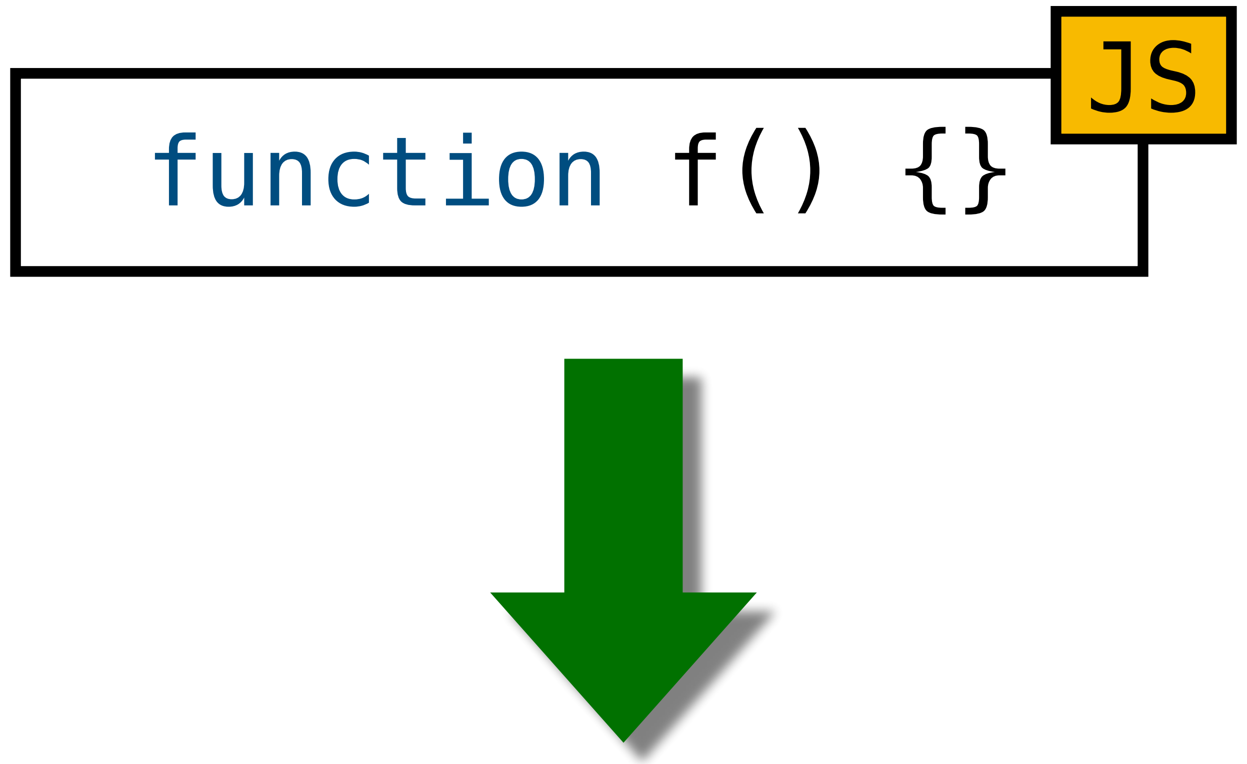
```
1n + 2n
```

```
var x = 1n + 2n;
+ $assert.equal(x, 3n);
```

```
4 + 2n
```

```
var x = 4 + 2n;
+ // [THROW] TypeError
```

JEST - Assertion Injection



```
function f() {}
```

```diff
  function f() {}

+ $assert.equal(Object.getPrototypeOf(f), Function.prototype);

+ $assert.verifyProperty(f, "prototype", {
+   writable: true,
+   enumerable: false,
+   configurable: false,
+ });

+ $assert.compare(Reflect.ownKeys(f), ['length', 'name', 'prototype'], f);

+ ...
```

JEST - Assertion Injection

JS

```js
function f() {}
```

Prototype Chain

```diff
  function f() {}

+ $assert.equal(Object.getPrototypeOf(f), Function.prototype);

+ $assert.verifyProperty(f, "prototype", {
+   writable: true,
+   enumerable: false,
+   configurable: false,
+ });

+ $assert.compare(Reflect.ownKeys(f), ['length', 'name', 'prototype'], f);

+ ...
```

JEST - Assertion Injection

JS

```
function f() {}
```

Prototype Chain

```
  function f() {}

+ $assert.equal(Object.getPrototypeOf(f), Function.prototype);

+ $assert.verifyProperty(f, "prototype", {
+   writable: true,
+   enumerable: false,
+   configurable: false,
+ });

+ $assert.compare(Reflect.ownKeys(f), ['length', 'name', 'prototype'], f);

+ ...
```

Property Descriptor

JEST - Assertion Injection

JS

```
function f() {}
```

```
  function f() {}

+ $assert.equal(Object.getPrototypeOf(f), Function.prototype);

+ $assert.verifyProperty(f, "prototype", {
+   writable: true,
+   enumerable: false,
+   configurable: false,
+ });

+ $assert.compare(Reflect.ownKeys(f), ['length', 'name', 'prototype'], f);

+ ...
```

Prototype Chain

Property Descriptor

Property Order

JEST - Assertion Injection

JS

```
function f() {}
```

```
  function f() {}

+ $assert.equal(Object.getPrototypeOf(f), Function.prototype);

+ $assert.verifyProperty(f, "prototype", {
+   writable: true,
+   enumerable: false,
+   configurable: false,
+ });

+ $assert.compare(Reflect.ownKeys(f), ['length', 'name', 'prototype'], f);

+ ...
```

Prototype Chain

Property Descriptor

Property Order

Etc.

JEST - Evaluation

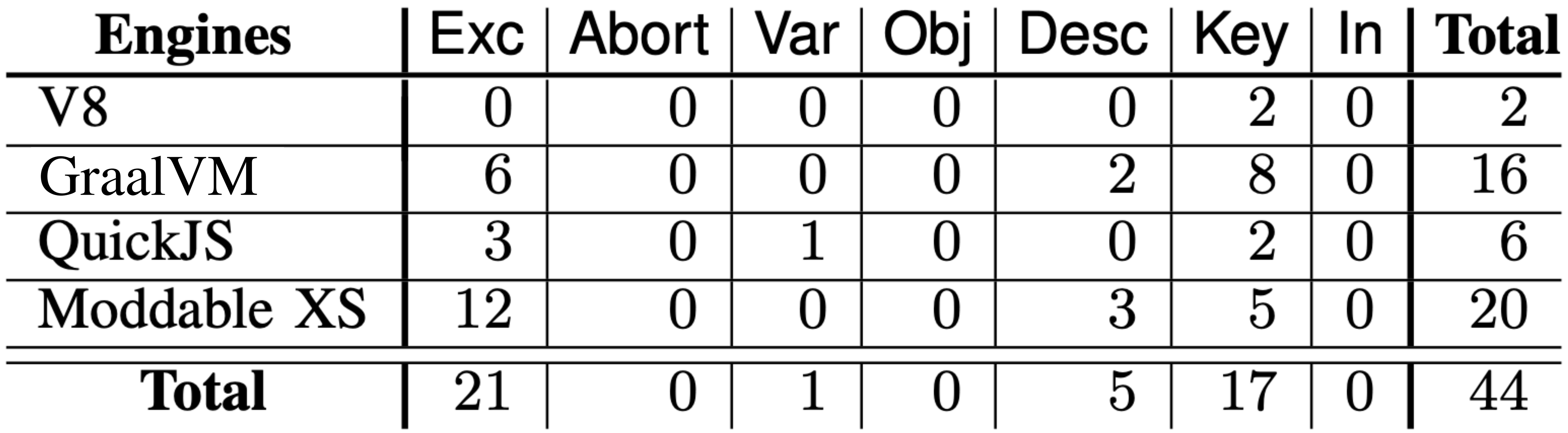
- JEST synthesized 1,700 conformance tests from ES2020

44 Bugs Detected

Crash

```
try { ++undefined; } catch (e) { }
```

TABLE II: The number of engine bugs detected by JEST



Engines	Exc	Abort	Var	Obj	Desc	Key	In	Total
V8	0	0	0	0	0	2	0	2
GraalVM	6	0	0	0	2	8	0	16
QuickJS	3	0	1	0	0	2	0	6
Moddable XS	12	0	0	0	3	5	0	20
Total	21	0	1	0	5	17	0	44

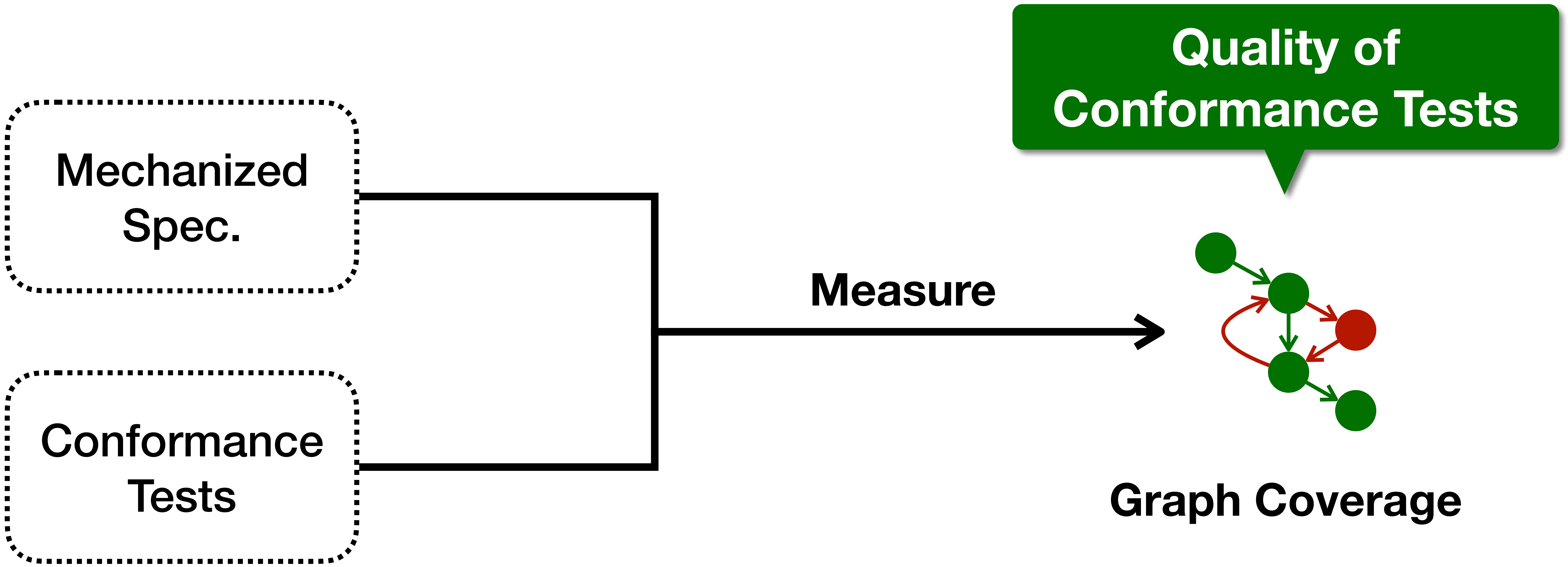
> *“Right now, we are running Test262 and the V8 and Nashorn unit test suites in our CI for every change, it might make sense to* ***add your suite as well****.”*

- A Developer of GraalVM™

PLDI 2023
Feature-Sens. Coverage
ASE 2020
ICSE 2021
ECMA-262 (JS Spec.)
Mech. Spec. Extraction
Mechanized Spec.
Conform. Test Synthesis
Conformance Tests
JavaScript Engines

[PLDI'23] J. Park, D. Youn, K, Lee, and S. Ryu, **Feature-Sensitive Coverage for Conformance Testing of Programming Language Implementations**

Graph Coverage for Language Specification



Mechanized Spec.

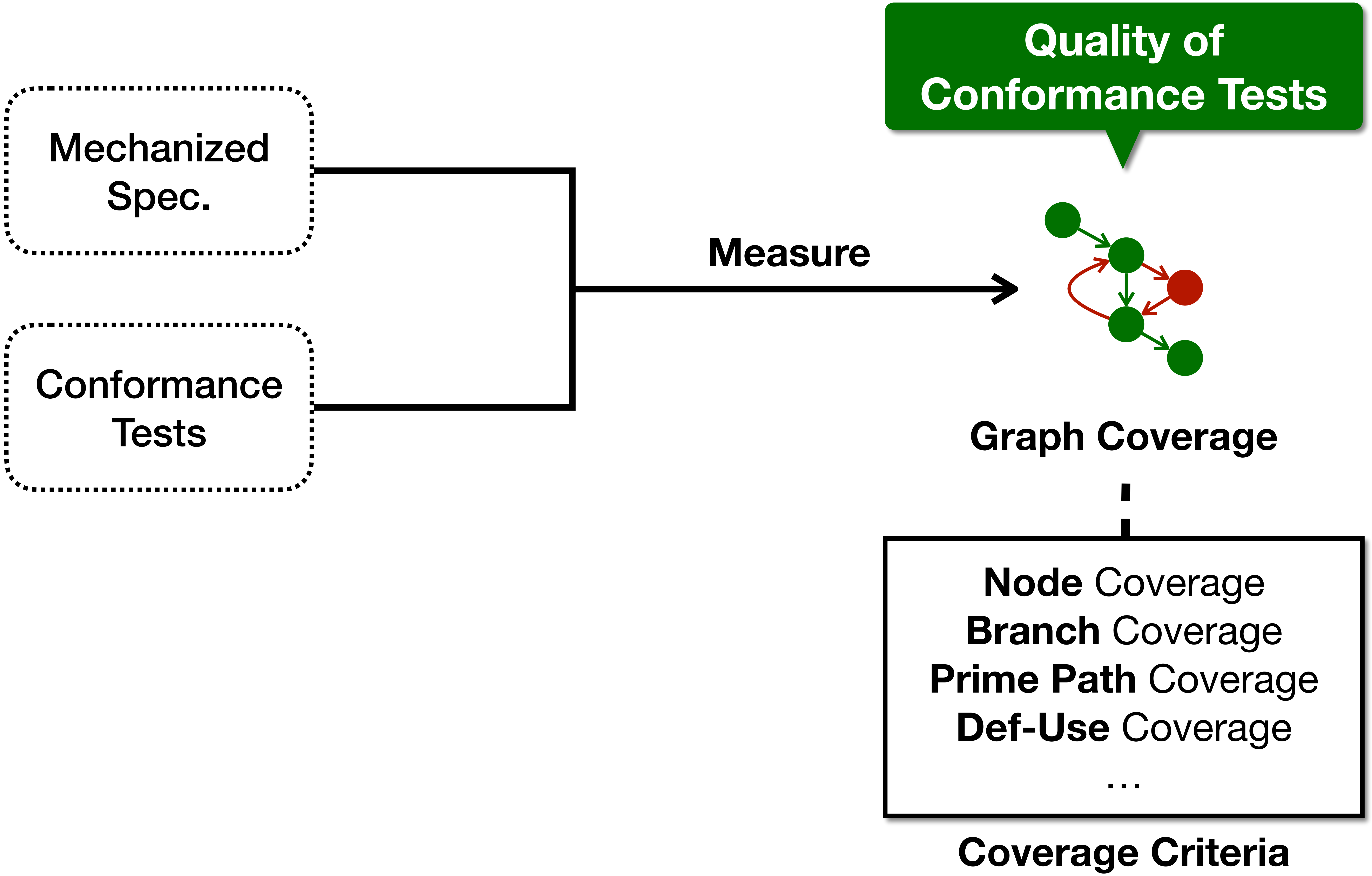
Conformance Tests

Measure

Quality of Conformance Tests

Graph Coverage

Graph Coverage for Language Specification



Mechanized Spec.

Conformance Tests

Measure

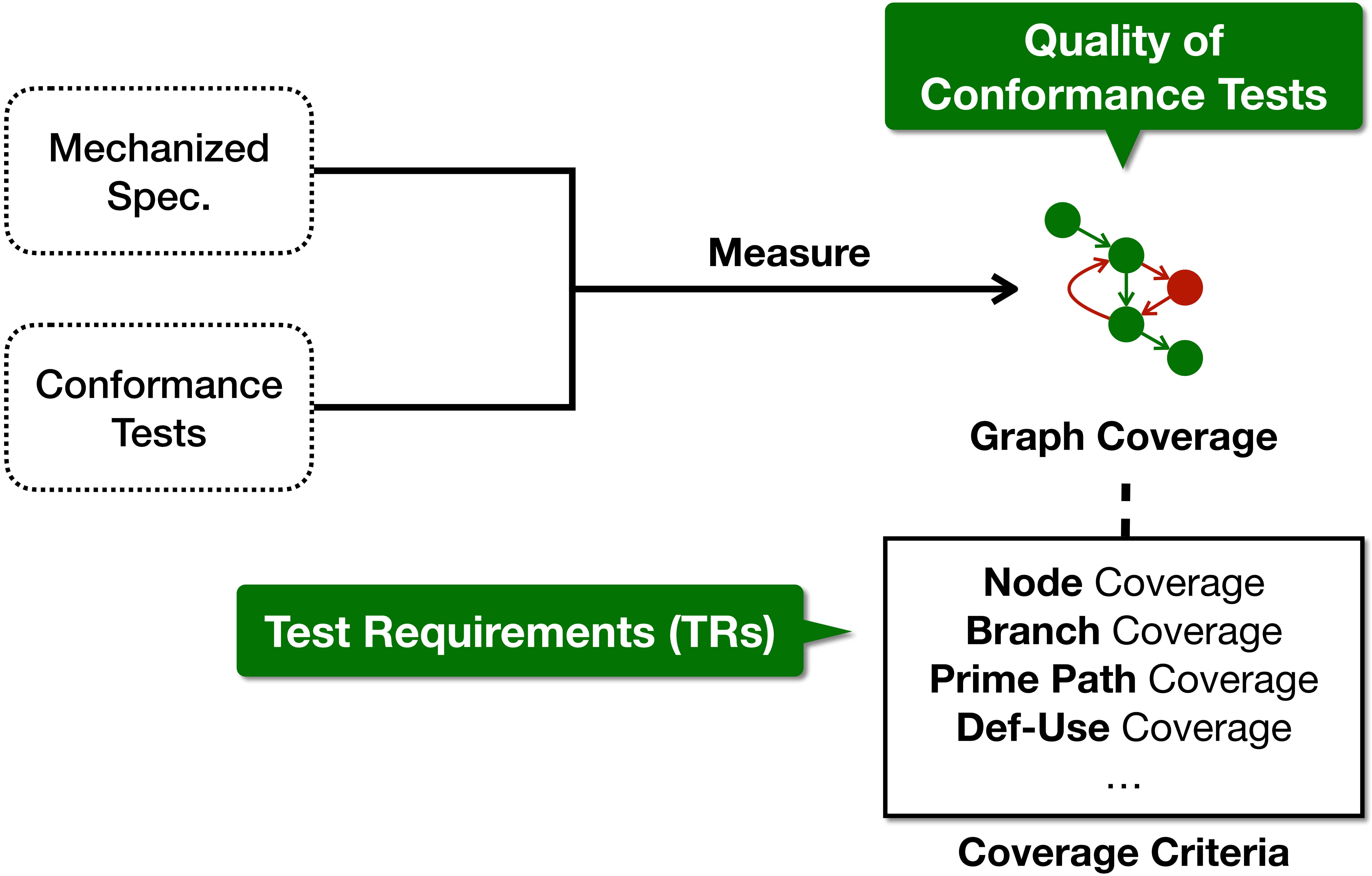
Quality of Conformance Tests

Graph Coverage

Node Coverage
Branch Coverage
Prime Path Coverage
Def-Use Coverage
…

Coverage Criteria

Graph Coverage for Language Specification



Mechanized Spec.

Conformance Tests

Measure

Quality of Conformance Tests

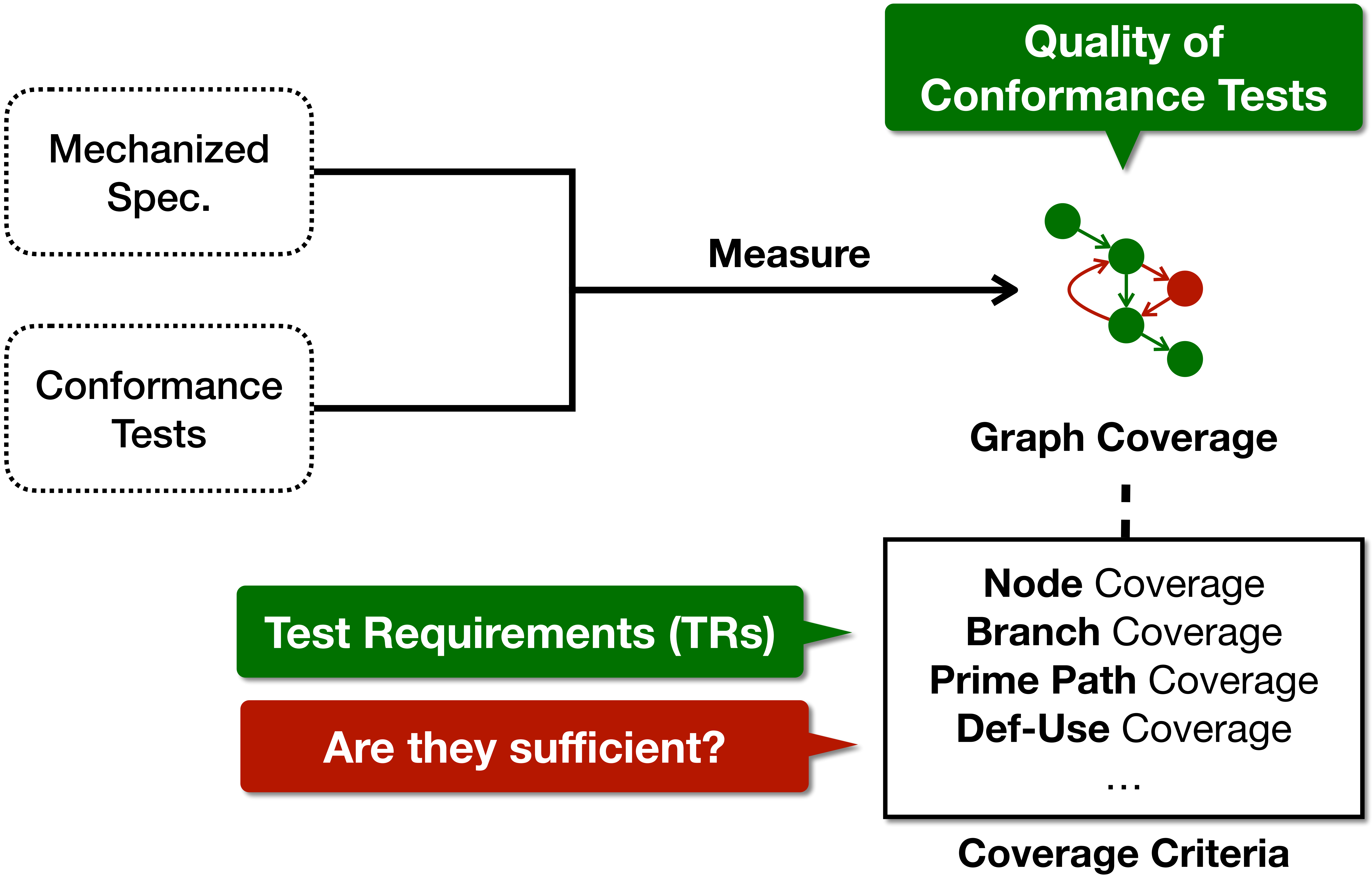
Graph Coverage

Test Requirements (TRs)

Node Coverage
Branch Coverage
Prime Path Coverage
Def-Use Coverage
…

Coverage Criteria

Graph Coverage for Language Specification



Mechanized Spec.

Conformance Tests

Measure

Quality of Conformance Tests

Graph Coverage

Test Requirements (TRs)

Are they sufficient?

Node Coverage
Branch Coverage
Prime Path Coverage
Def-Use Coverage
…

Coverage Criteria

Motivating Example 1 with **Node Coverage**

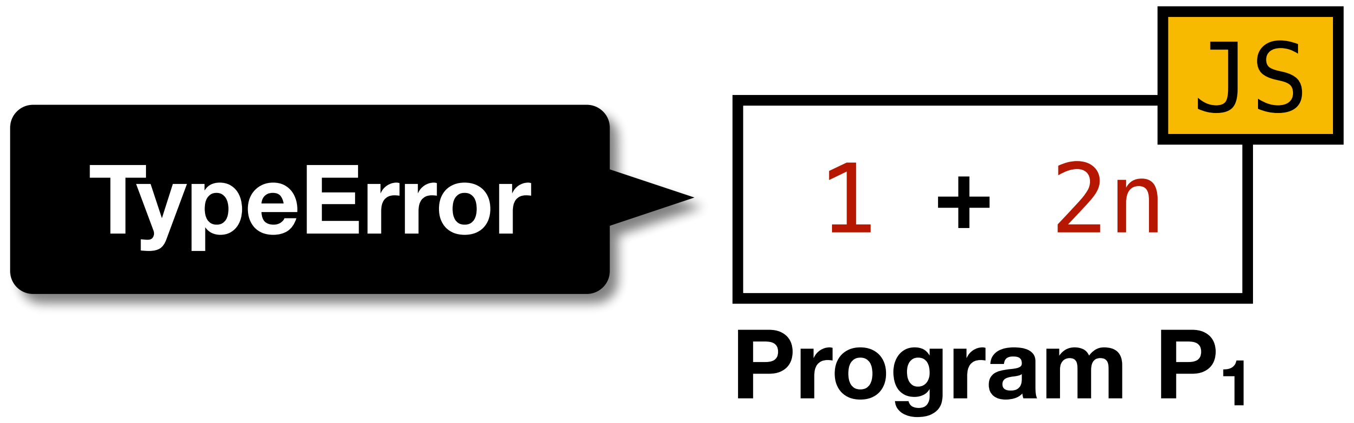
TypeError

JS

```
1 + 2n
```

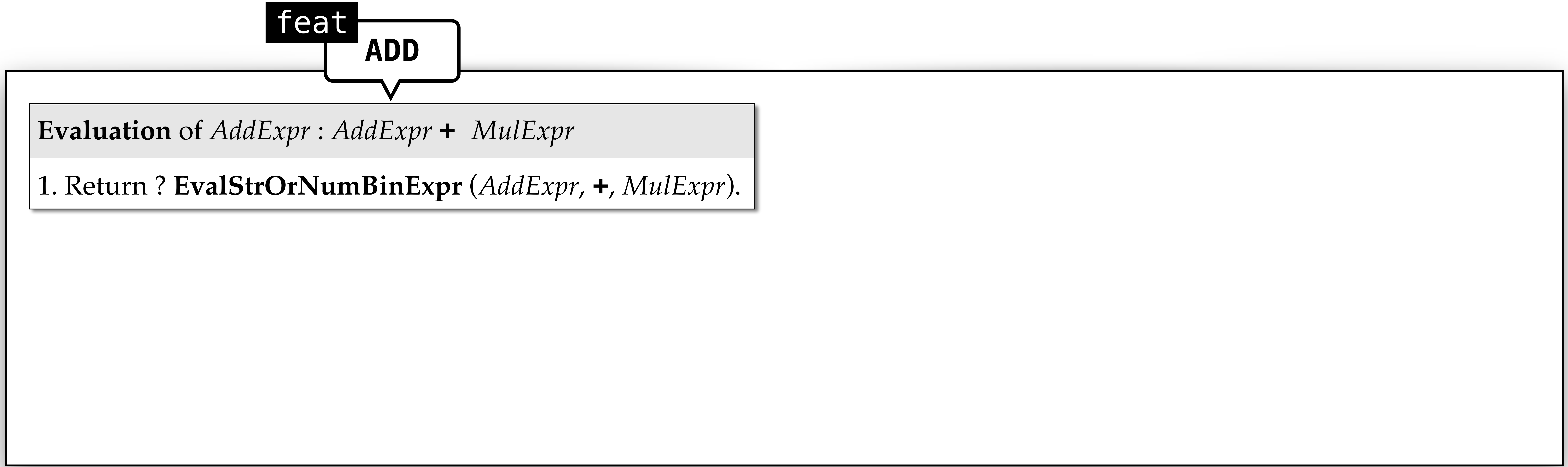
Program P_1

Motivating Example 1 with Node Coverage



```
1 + 2n
```

Program P_1

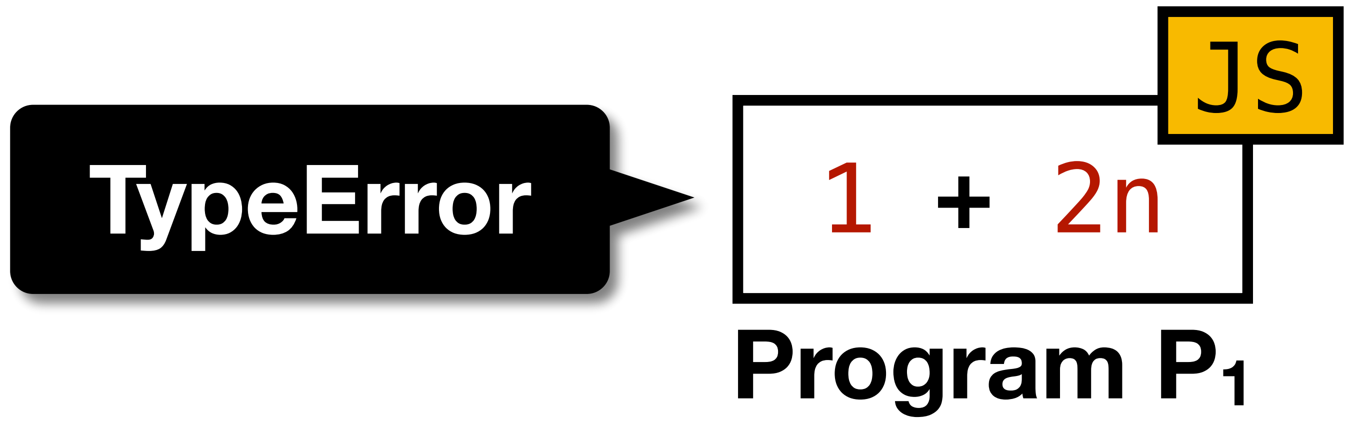


Evaluation of *AddExpr* : *AddExpr* **+** *MulExpr*

1. Return ? **EvalStrOrNumBinExpr** (*AddExpr*, **+**, *MulExpr*).

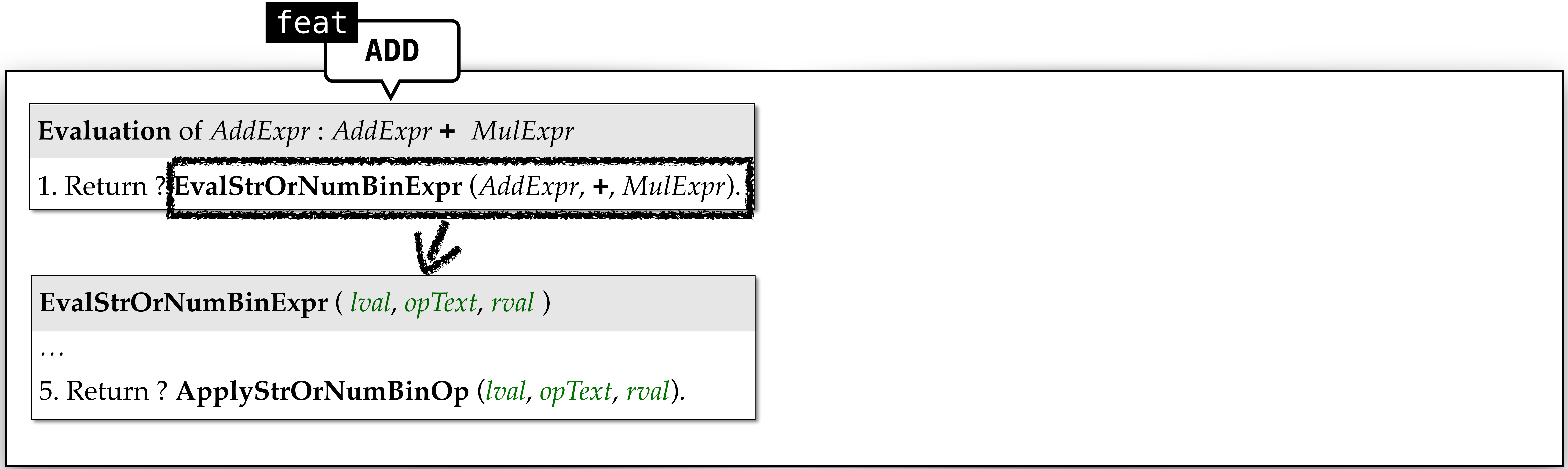
Abstract Algorithms in **ECMA-262** (ES13, 2022), **JavaScript Language Specification**

Motivating Example 1 with **Node Coverage**



```
1 + 2n
```

Program P$_1$



Evaluation of *AddExpr* : *AddExpr* **+** *MulExpr*

1. Return ? **EvalStrOrNumBinExpr** (*AddExpr*, **+**, *MulExpr*).

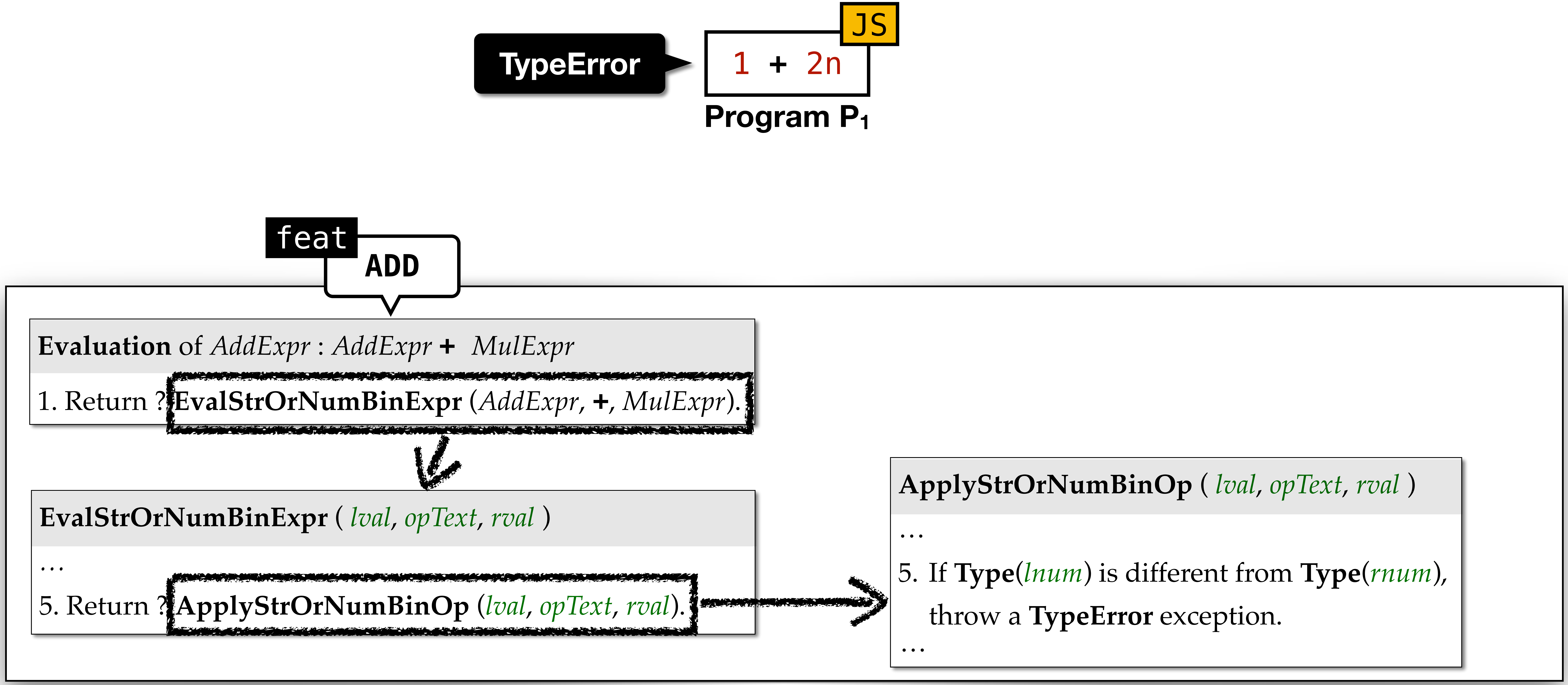
EvalStrOrNumBinExpr (*lval*, *opText*, *rval*)

…

5. Return ? **ApplyStrOrNumBinOp** (*lval*, *opText*, *rval*).

Abstract Algorithms in **ECMA-262** (ES13, 2022), **JavaScript Language Specification**

Motivating Example 1 with Node Coverage



TypeError

JS

```
1 + 2n
```

Program P₁

feat

ADD

Evaluation of *AddExpr* : *AddExpr* **+** *MulExpr*

1. Return ? **EvalStrOrNumBinExpr** (*AddExpr*, **+**, *MulExpr*).

EvalStrOrNumBinExpr (*lval*, *opText*, *rval*)

…

5. Return ? **ApplyStrOrNumBinOp** (*lval*, *opText*, *rval*).

ApplyStrOrNumBinOp (*lval*, *opText*, *rval*)

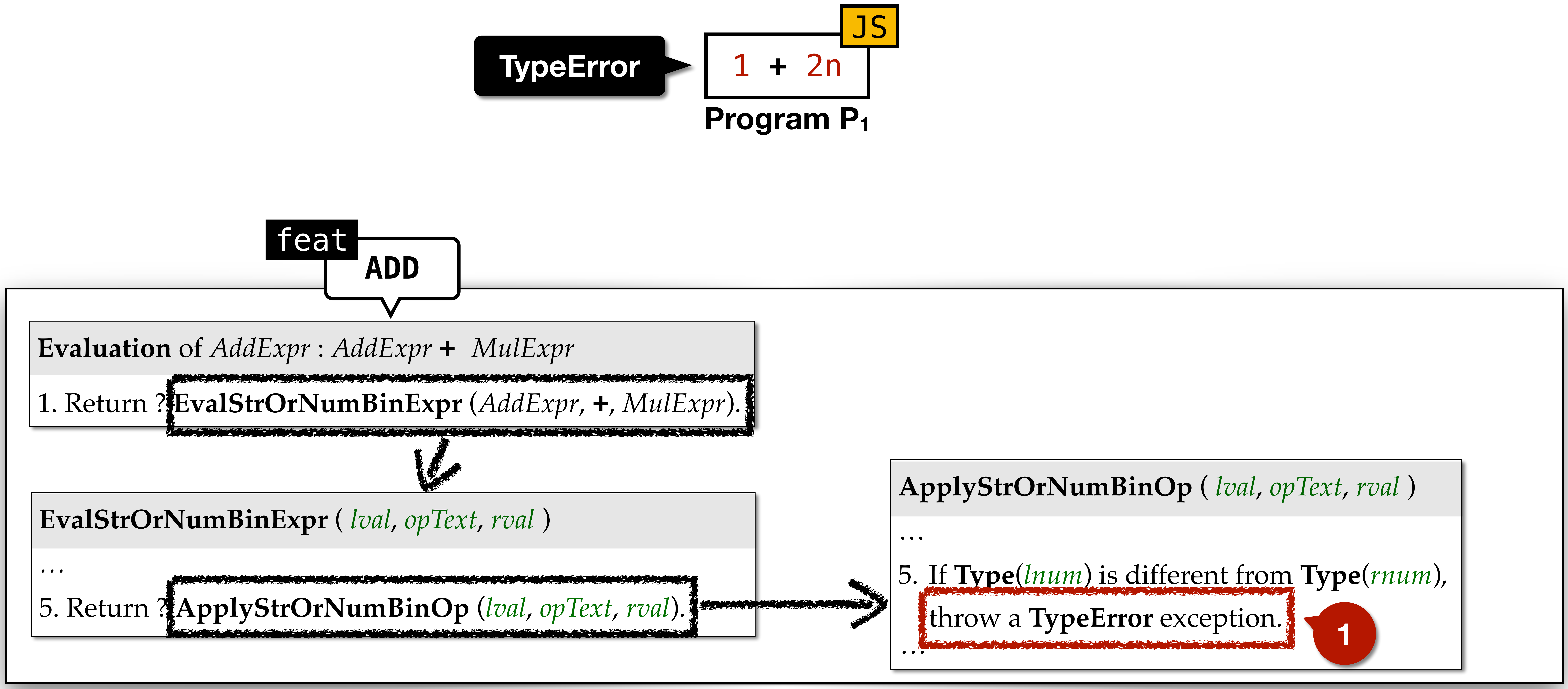
…

5. If **Type**(*lnum*) is different from **Type**(*rnum*), throw a **TypeError** exception.

…

Abstract Algorithms in **ECMA-262** (ES13, 2022), **JavaScript Language Specification**

Motivating Example 1 with **Node Coverage**



Evaluation of *AddExpr* : *AddExpr* **+** *MulExpr*

1. Return ? **EvalStrOrNumBinExpr** (*AddExpr*, **+**, *MulExpr*).

EvalStrOrNumBinExpr (*lval*, *opText*, *rval*)

…

5. Return ? **ApplyStrOrNumBinOp** (*lval*, *opText*, *rval*).

ApplyStrOrNumBinOp (*lval*, *opText*, *rval*)

…

5. If **Type**(*lnum*) is different from **Type**(*rnum*), throw a **TypeError** exception.

…

Abstract Algorithms in **ECMA-262** (ES13, 2022), **JavaScript Language Specification**

Motivating Example 1 with Node Coverage

Node Coverage

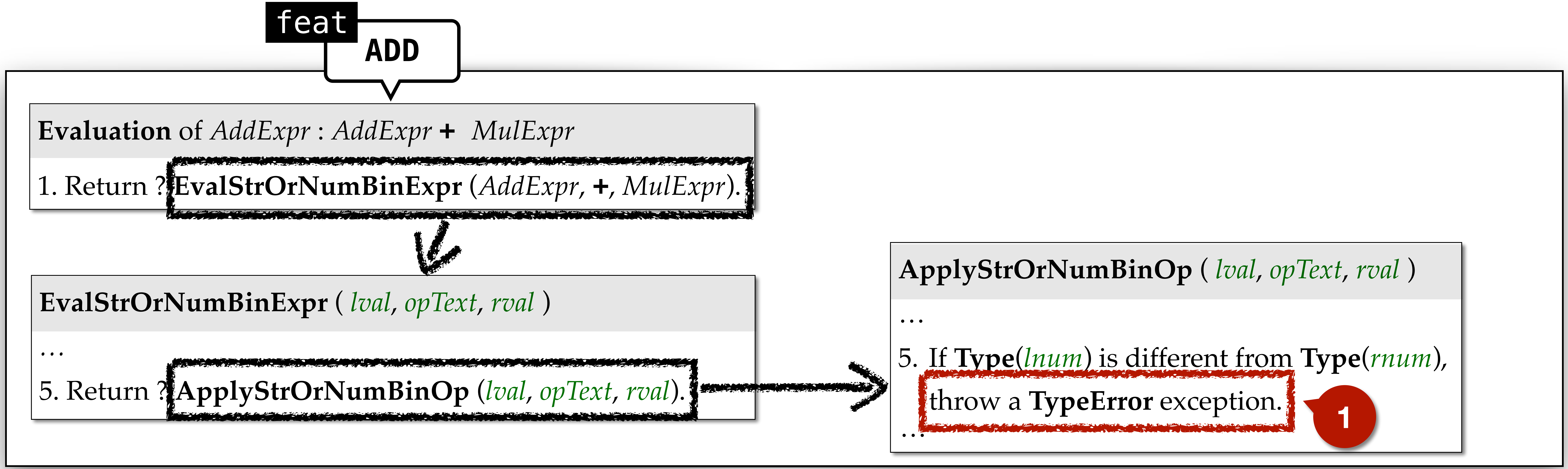
TR = Node

TypeError

JS

```
1 + 2n
```

Program P_1



Evaluation of *AddExpr* : *AddExpr* **+** *MulExpr*

1. Return ? **EvalStrOrNumBinExpr** (*AddExpr*, **+**, *MulExpr*).

EvalStrOrNumBinExpr (*lval*, *opText*, *rval*)

…

5. Return ? **ApplyStrOrNumBinOp** (*lval*, *opText*, *rval*).

ApplyStrOrNumBinOp (*lval*, *opText*, *rval*)

…

5. If **Type**(*lnum*) is different from **Type**(*rnum*), throw a **TypeError** exception.

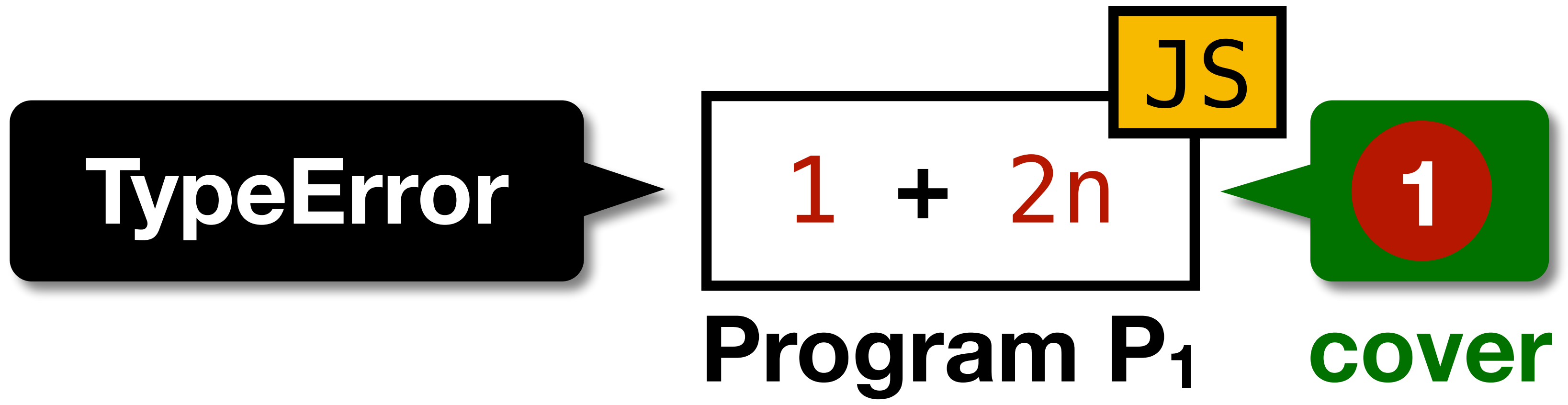
…

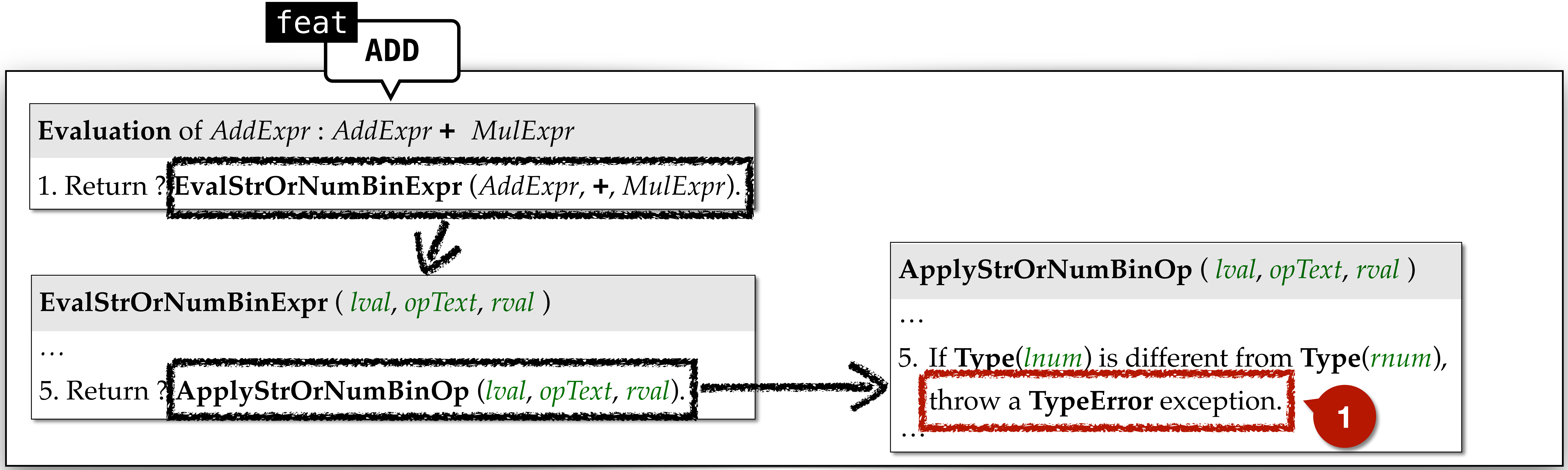
Abstract Algorithms in **ECMA-262** (ES13, 2022), **JavaScript Language Specification**

Motivating Example 1 with **Node Coverage**

Node Coverage

TR = Node





Evaluation of *AddExpr* : *AddExpr* **+** *MulExpr*

1. Return ? **EvalStrOrNumBinExpr** (*AddExpr*, **+**, *MulExpr*).

EvalStrOrNumBinExpr (*lval*, *opText*, *rval*)

…

5. Return ? **ApplyStrOrNumBinOp** (*lval*, *opText*, *rval*).

ApplyStrOrNumBinOp (*lval*, *opText*, *rval*)

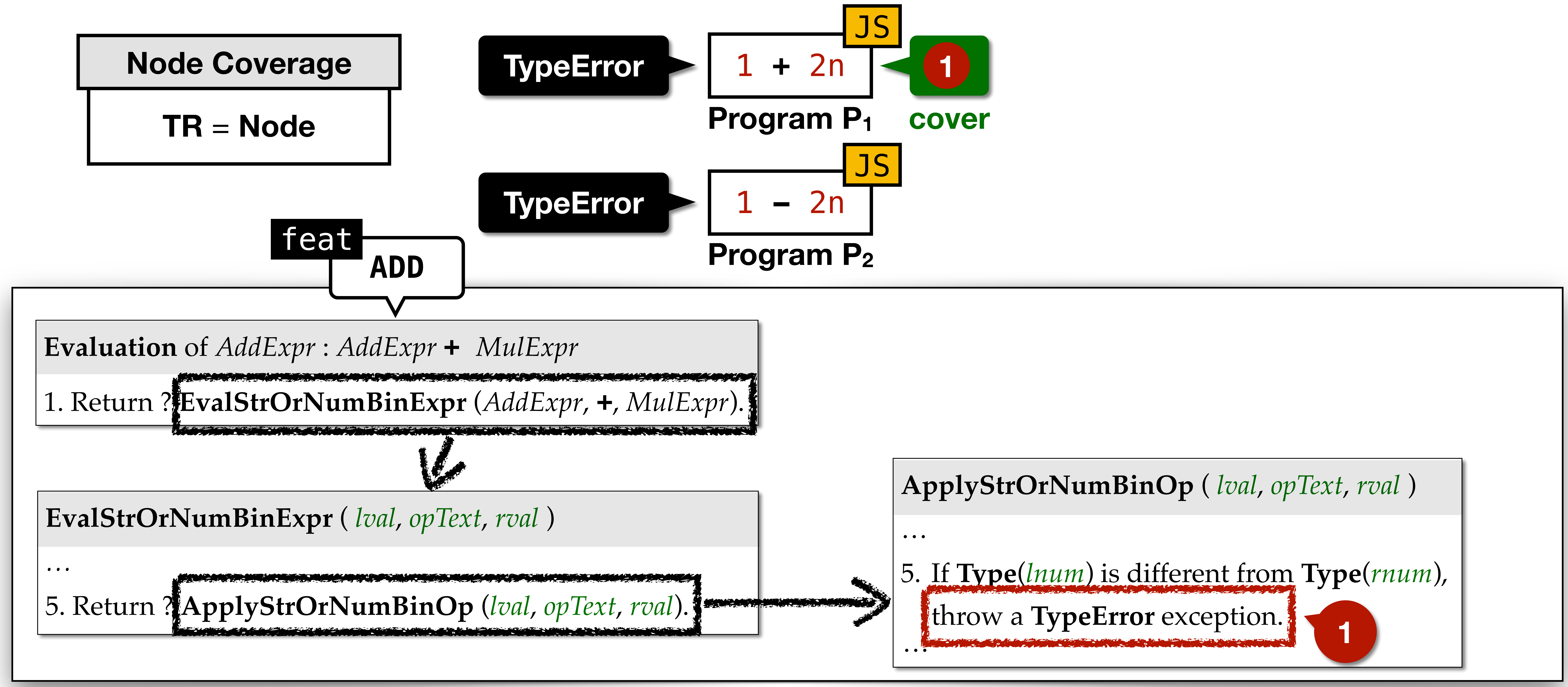
…

5. If **Type**(*lnum*) is different from **Type**(*rnum*), throw a **TypeError** exception.

…

Abstract Algorithms in **ECMA-262** (ES13, 2022), **JavaScript Language Specification**

Motivating Example 1 with Node Coverage



```
1 + 2n
```

```
1 - 2n
```

Evaluation of *AddExpr* : *AddExpr* **+** *MulExpr*

1. Return ? **EvalStrOrNumBinExpr** (*AddExpr*, **+**, *MulExpr*).

EvalStrOrNumBinExpr (*lval*, *opText*, *rval*)

…

5. Return ? **ApplyStrOrNumBinOp** (*lval*, *opText*, *rval*).

ApplyStrOrNumBinOp (*lval*, *opText*, *rval*)

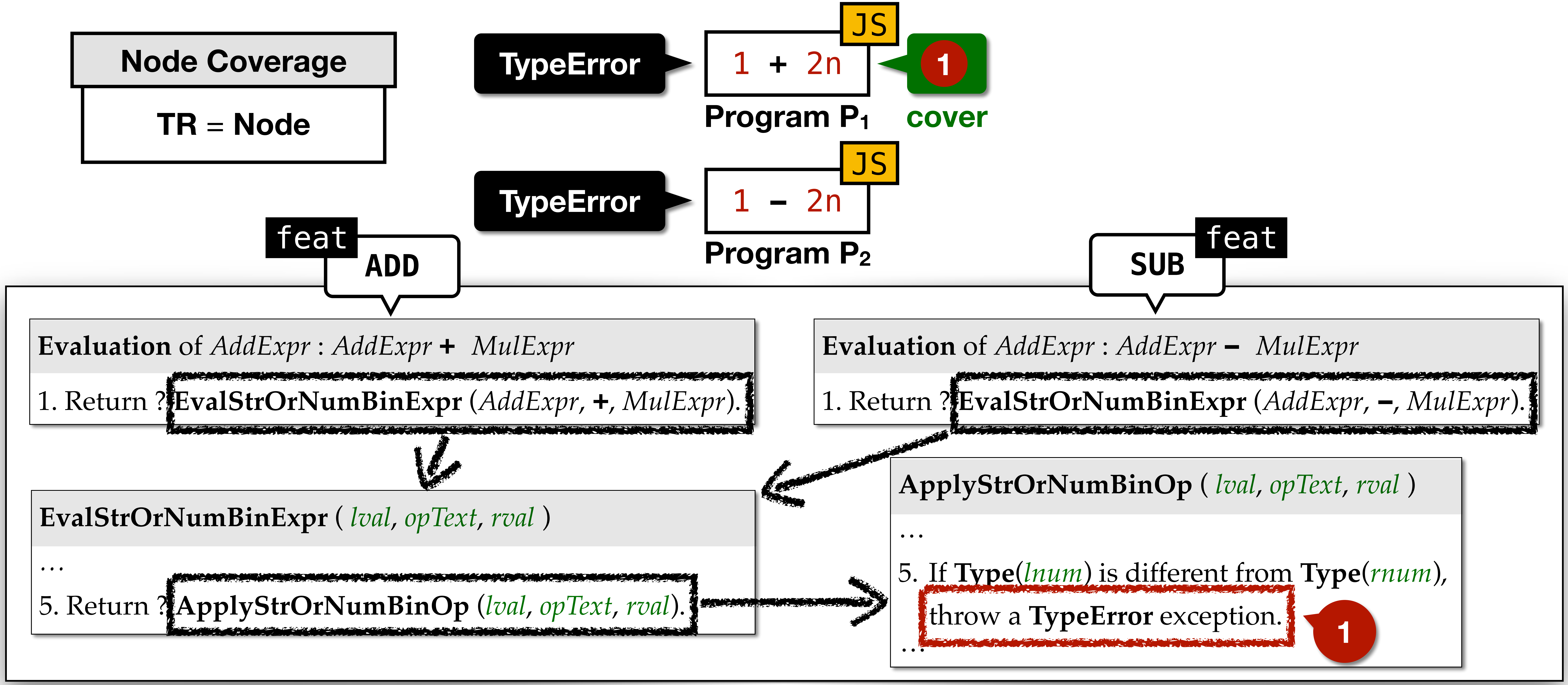
…

5. If **Type**(*lnum*) is different from **Type**(*rnum*), throw a **TypeError** exception.

…

Abstract Algorithms in **ECMA-262** (ES13, 2022), **JavaScript Language Specification**

Motivating Example 1 with **Node Coverage**



Node Coverage

TR = Node

Evaluation of *AddExpr* : *AddExpr* **+** *MulExpr*

1. Return ? **EvalStrOrNumBinExpr** (*AddExpr*, **+**, *MulExpr*).

Evaluation of *AddExpr* : *AddExpr* **–** *MulExpr*

1. Return ? **EvalStrOrNumBinExpr** (*AddExpr*, **–**, *MulExpr*).

EvalStrOrNumBinExpr (*lval*, *opText*, *rval*)

…

5. Return ? **ApplyStrOrNumBinOp** (*lval*, *opText*, *rval*).

ApplyStrOrNumBinOp (*lval*, *opText*, *rval*)

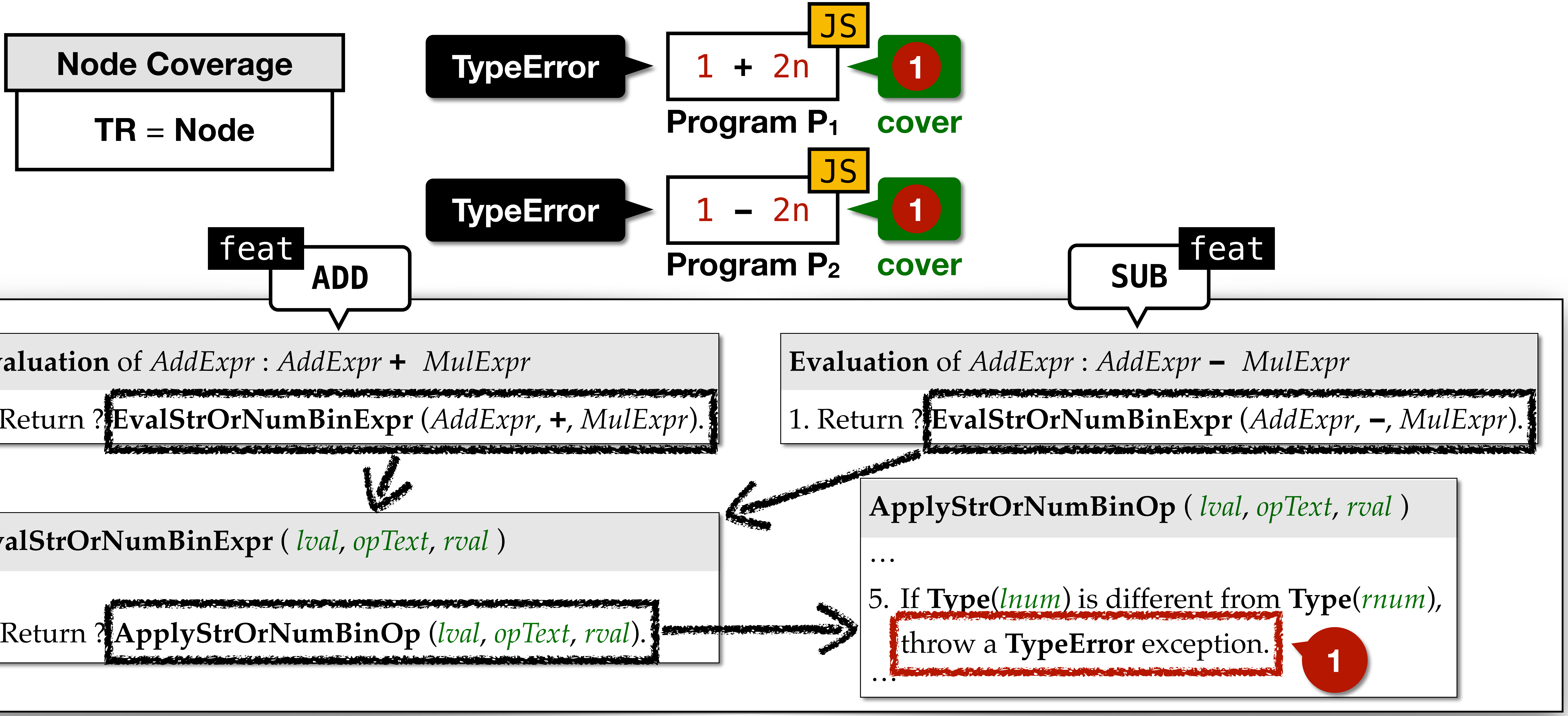
…

5. If **Type**(*lnum*) is different from **Type**(*rnum*), throw a **TypeError** exception.

…

Abstract Algorithms in **ECMA-262** (ES13, 2022), **JavaScript Language Specification**

Motivating Example 1 with **Node Coverage**



Node Coverage

TR = **Node**

TypeError → Program P_1: `1 + 2n` (JS) ← 1 cover

TypeError → Program P_2: `1 - 2n` (JS) ← 1 cover

Evaluation of *AddExpr* : *AddExpr* **+** *MulExpr*

1. Return ? **EvalStrOrNumBinExpr** (*AddExpr*, **+**, *MulExpr*).

Evaluation of *AddExpr* : *AddExpr* **–** *MulExpr*

1. Return ? **EvalStrOrNumBinExpr** (*AddExpr*, **–**, *MulExpr*).

EvalStrOrNumBinExpr (*lval*, *opText*, *rval*)

…

5. Return ? **ApplyStrOrNumBinOp** (*lval*, *opText*, *rval*).

ApplyStrOrNumBinOp (*lval*, *opText*, *rval*)

…

5. If **Type**(*lnum*) is different from **Type**(*rnum*), throw a **TypeError** exception.

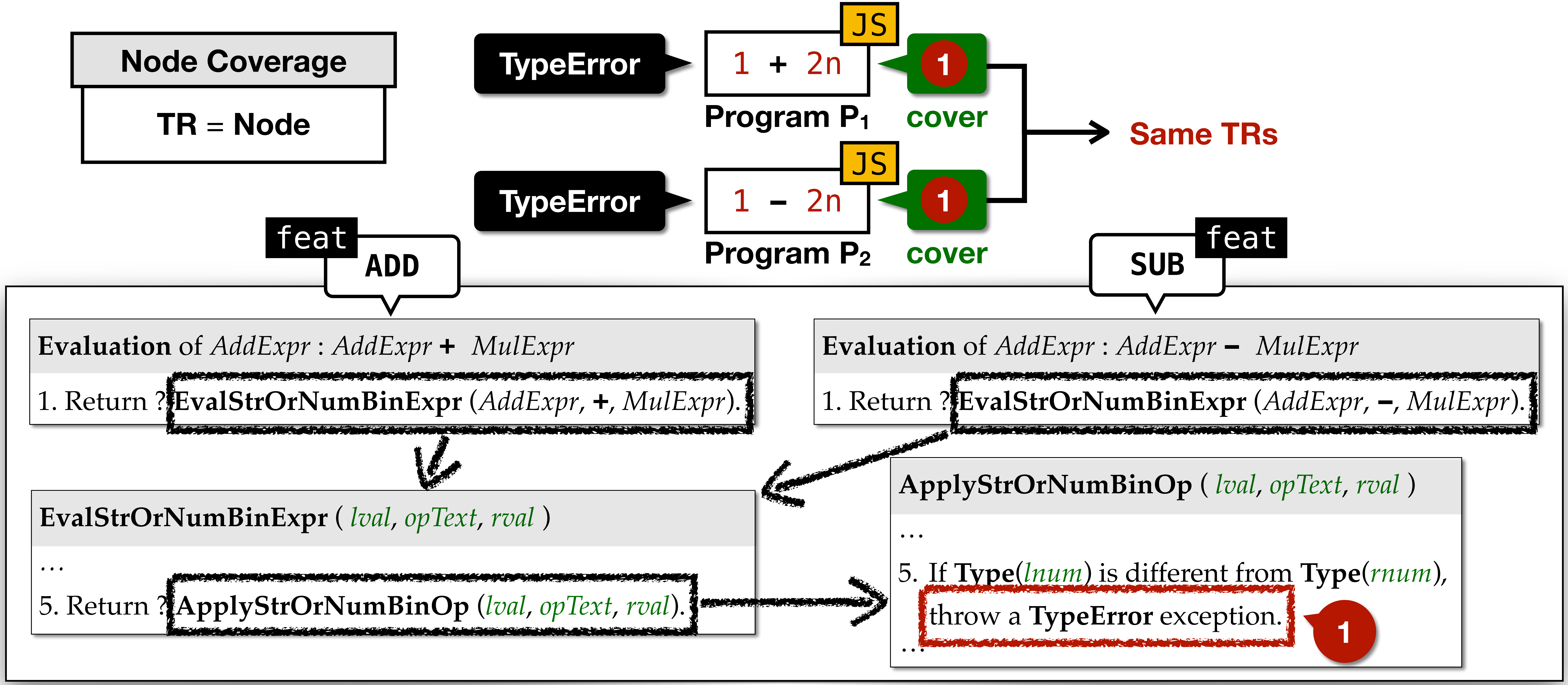
…

Abstract Algorithms in **ECMA-262** (ES13, 2022), **JavaScript Language Specification**

Motivating Example 1 with **Node Coverage**

Node Coverage

TR = **Node**



Evaluation of *AddExpr* : *AddExpr* **+** *MulExpr*

1. Return ? **EvalStrOrNumBinExpr** (*AddExpr*, **+**, *MulExpr*).

Evaluation of *AddExpr* : *AddExpr* **–** *MulExpr*

1. Return ? **EvalStrOrNumBinExpr** (*AddExpr*, **–**, *MulExpr*).

EvalStrOrNumBinExpr (*lval*, *opText*, *rval*)

…

5. Return ? **ApplyStrOrNumBinOp** (*lval*, *opText*, *rval*).

ApplyStrOrNumBinOp (*lval*, *opText*, *rval*)

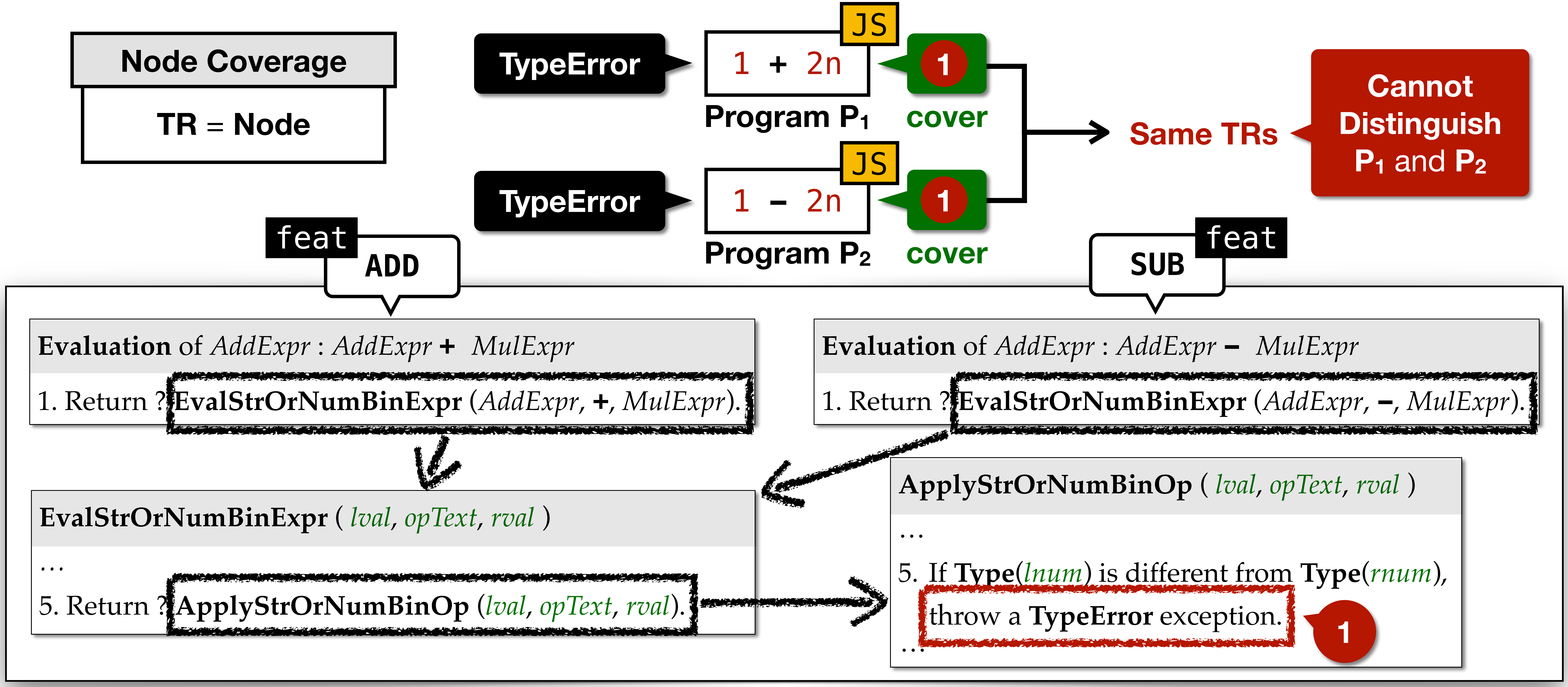
…

5. If **Type**(*lnum*) is different from **Type**(*rnum*), throw a **TypeError** exception.

…

Abstract Algorithms in **ECMA-262** (ES13, 2022), **JavaScript Language Specification**

Motivating Example 1 with **Node Coverage**



Node Coverage

TR = Node

TypeError

JS

```
1 + 2n
```

Program P_1

1 cover

TypeError

JS

```
1 - 2n
```

Program P_2

1 cover

Same TRs

Cannot Distinguish P_1 and P_2

feat ADD

feat SUB

Evaluation of *AddExpr* : *AddExpr* **+** *MulExpr*

1. Return ? **EvalStrOrNumBinExpr** (*AddExpr*, **+**, *MulExpr*).

Evaluation of *AddExpr* : *AddExpr* **–** *MulExpr*

1. Return ? **EvalStrOrNumBinExpr** (*AddExpr*, **–**, *MulExpr*).

EvalStrOrNumBinExpr (*lval*, *opText*, *rval*)

…

5. Return ? **ApplyStrOrNumBinOp** (*lval*, *opText*, *rval*).

ApplyStrOrNumBinOp (*lval*, *opText*, *rval*)

…

5. If **Type**(*lnum*) is different from **Type**(*rnum*), throw a **TypeError** exception.

1

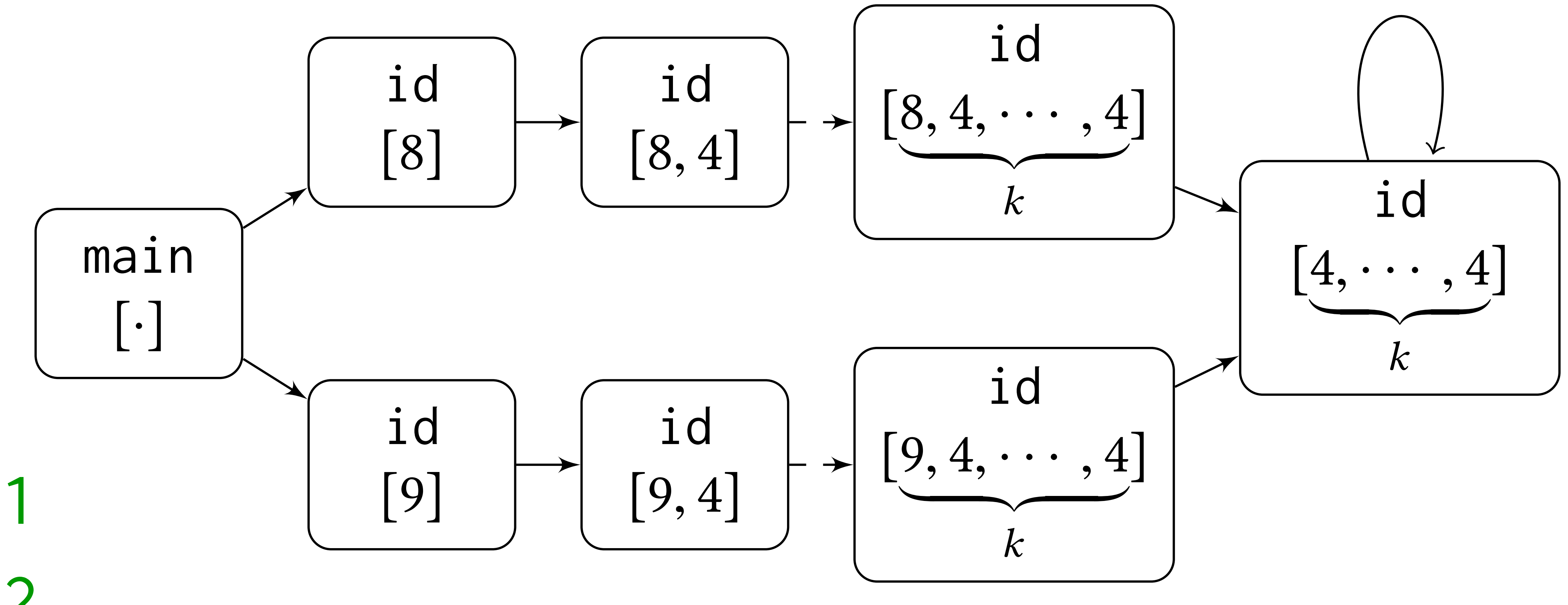
…

Abstract Algorithms in **ECMA-262** (ES13, 2022), **JavaScript Language Specification**

PLRG

Insight from **Context Tunneling** [OOPSLA'18]

```
1  class A {} class B {}
2  class C {
3    static Object id (Object v, int i){
4        return i >= 0 ? id(v, i-1) : v;
5    }
6    public static void main (){
7      int i = input();
8      A a = (A) id(new A(), i); //Query 1
9      B b = (B) id(new B(), i); //Query 2
10   }
11 }
```


main
$[\cdot]$
id
$[8]$
id
$[8,4]$
id
$[\underbrace{8,4,\cdots,4}_{k}]$
id
$[9]$
id
$[9,4]$
id
$[\underbrace{9,4,\cdots,4}_{k}]$
id
$[\underbrace{4,\cdots,4}_{k}]$

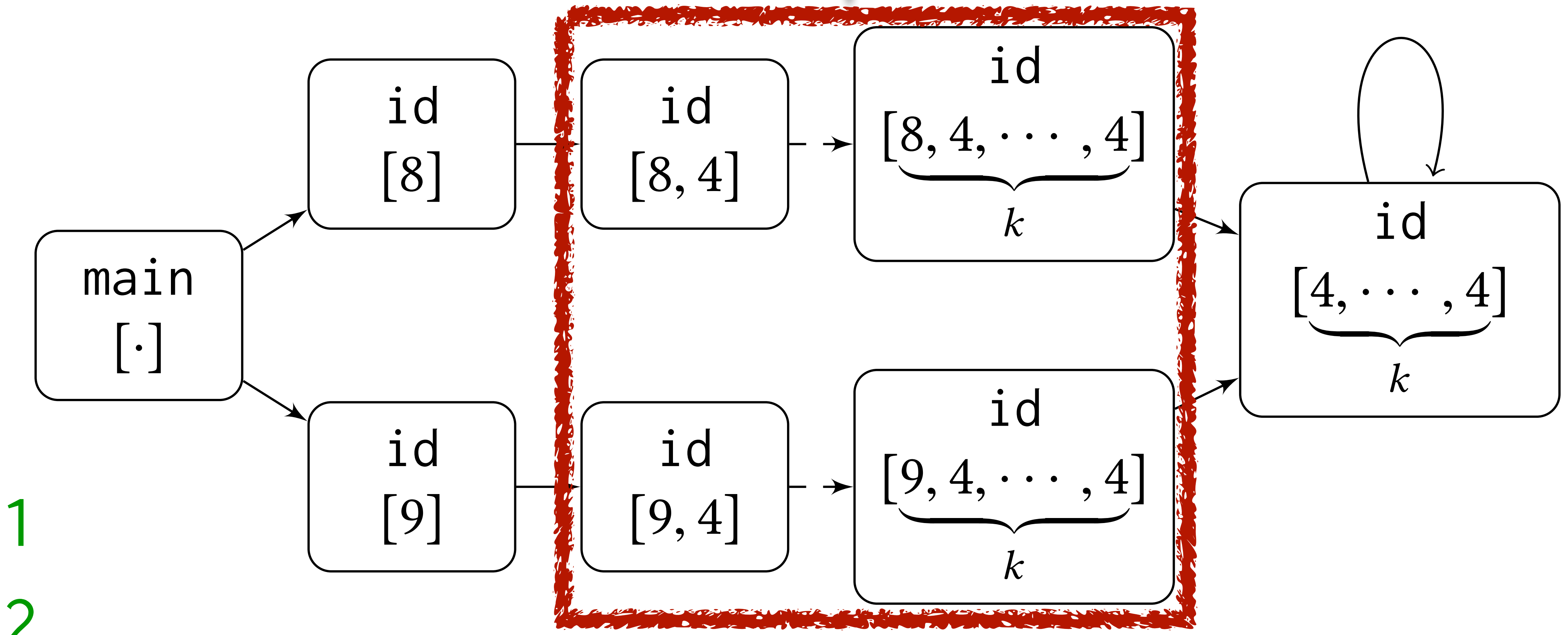
k-**Callsite Sensitivity**

[OOPSLA'18] M. Jeon, S. Jeong, and H, Oh, **Precise and Scalable Points-to Analysis via Data-Driven Context Tunneling**

Insight from **Context Tunneling** [OOPSLA’18]

```
 1 class A {} class B {}
 2 class C {
 3   static Object id (Object v, int i){
 4       return i >= 0 ? id(v, i-1) : v;
 5   }
 6   public static void main (){
 7     int i = input();
 8     A a = (A) id(new A(), i); //Query 1
 9     B b = (B) id(new B(), i); //Query 2
10   }
11 }
```

Unnecessary Split


main
$[\cdot]$
id
$[8]$
id
$[9]$
id
$[8, 4]$
id
$[9, 4]$
id
$[\underbrace{8, 4, \cdots, 4}_{k}]$
id
$[\underbrace{9, 4, \cdots, 4}_{k}]$
id
$[\underbrace{4, \cdots, 4}_{k}]$

k**-Callsite Sensitivity**

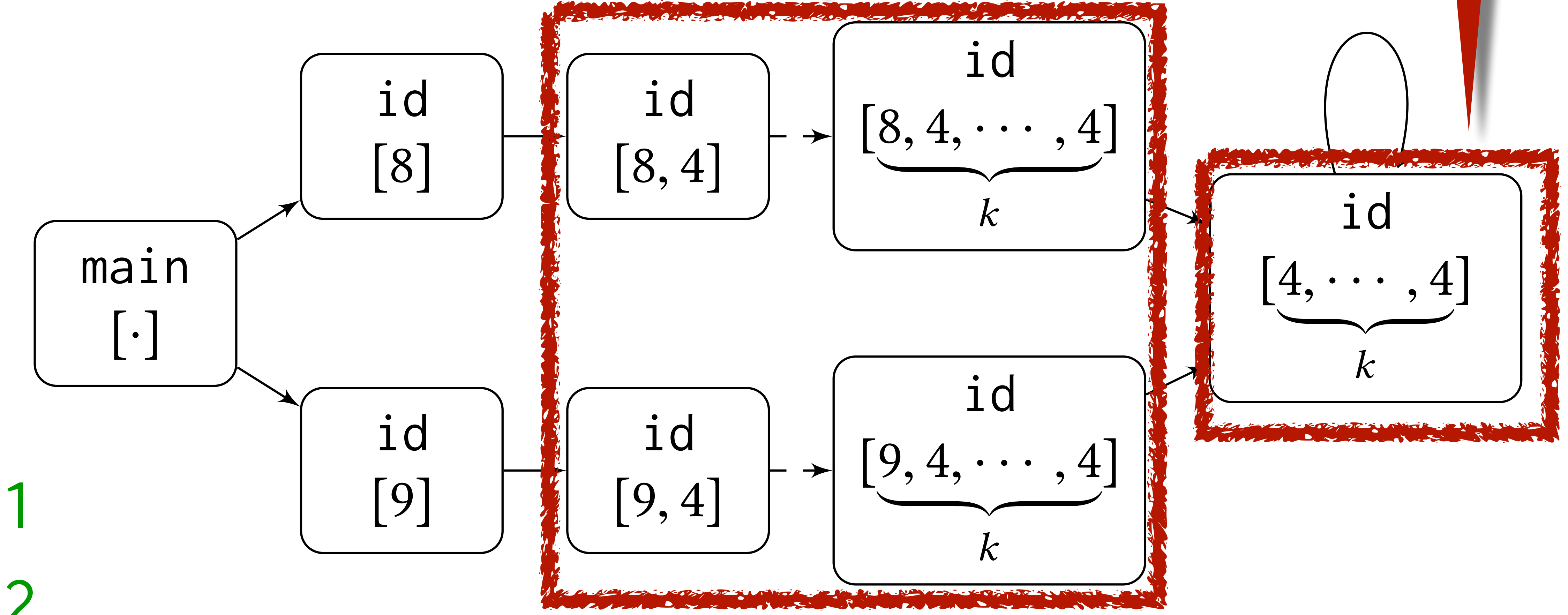
[OOPSLA’18] M. Jeon, S. Jeong, and H, Oh, **Precise and Scalable Points-to Analysis via Data-Driven Context Tunneling**

Insight from **Context Tunneling** [OOPSLA'18]

```
 1 class A {} class B {}
 2 class C {
 3   static Object id (Object v, int i){
 4       return i >= 0 ? id(v, i-1) : v;
 5   }
 6   public static void main (){
 7     int i = input();
 8     A a = (A) id(new A(), i); //Query 1
 9     B b = (B) id(new B(), i); //Query 2
10   }
11 }
```

Unnecessary Split

Unnecessary Merge



main $[\cdot]$
id $[8]$
id $[9]$
id $[8,4]$
id $[\underbrace{8,4,\cdots,4}_{k}]$
id $[9,4]$
id $[\underbrace{9,4,\cdots,4}_{k}]$
id $[\underbrace{4,\cdots,4}_{k}]$

***k*-Callsite Sensitivity**

[OOPSLA'18] M. Jeon, S. Jeong, and H, Oh, **Precise and Scalable Points-to Analysis via Data-Driven Context Tunneling**

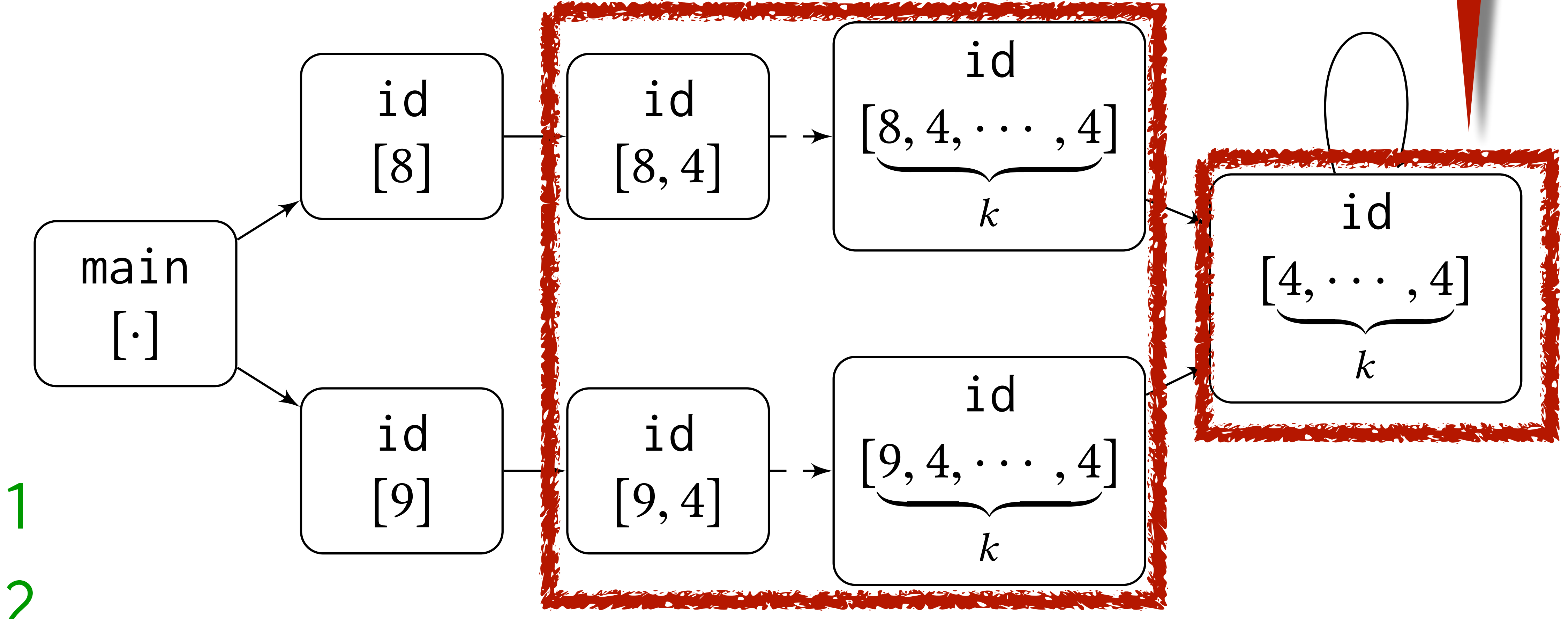
Insight from **Context Tunneling** [OOPSLA'18]

Not Important

```
class A {} class B {}
class C {
  static Object id (Object v, int i){
      return i >= 0 ? id(v, i-1) : v;
  }
  public static void main (){
    int i = input();
    A a = (A) id(new A(), i); //Query 1
    B b = (B) id(new B(), i); //Query 2
  }
}
```

Unnecessary Split

Unnecessary Merge


main
$[\cdot]$
id
$[8]$
id
$[9]$
id
$[8,4]$
id
$[9,4]$
id
$[\underbrace{8,4,\cdots,4}_{k}]$
id
$[\underbrace{9,4,\cdots,4}_{k}]$
id
$[\underbrace{4,\cdots,4}_{k}]$

k-Callsite Sensitivity

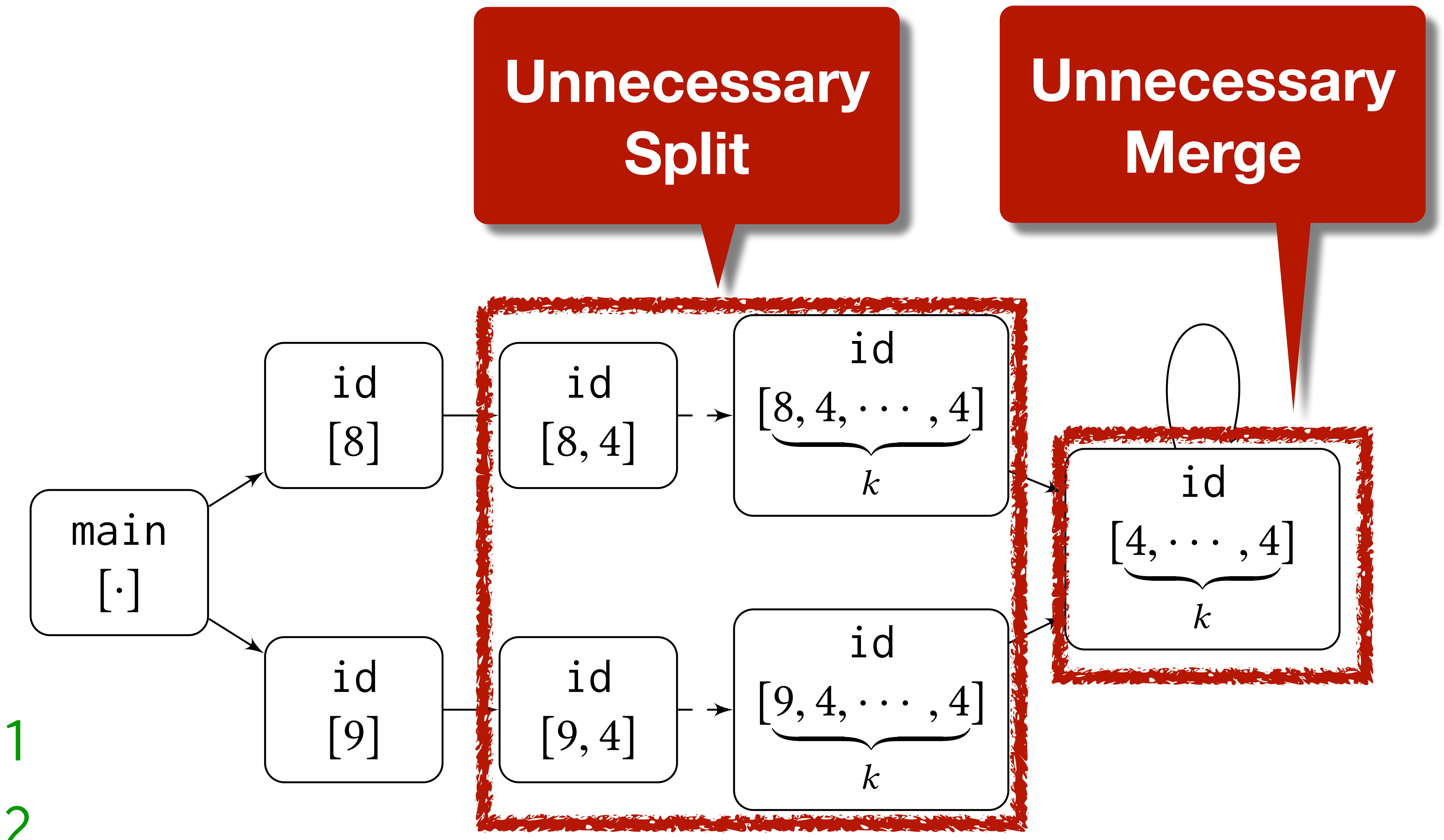
[OOPSLA'18] M. Jeon, S. Jeong, and H, Oh, **Precise and Scalable Points-to Analysis via Data-Driven Context Tunneling**

Insight from **Context Tunneling** [OOPSLA'18]

Not Important

Important

```
 1 class A {} class B {}
 2 class C {
 3   static Object id (Object v, int i){
 4       return i >= 0 ? id(v, i-1) : v;
 5   }
 6   public static void main (){
 7     int i = input();
 8     A a = (A) id(new A(), i); //Query 1
 9     B b = (B) id(new B(), i); //Query 2
10   }
11 }
```



Unnecessary Split
Unnecessary Merge
main $[\cdot]$
id $[8]$
id $[9]$
id $[8,4]$
id $[9,4]$
id $[8,\underbrace{4,\cdots,4}_{k}]$
id $[9,\underbrace{4,\cdots,4}_{k}]$
id $[\underbrace{4,\cdots,4}_{k}]$

k**-Callsite Sensitivity**

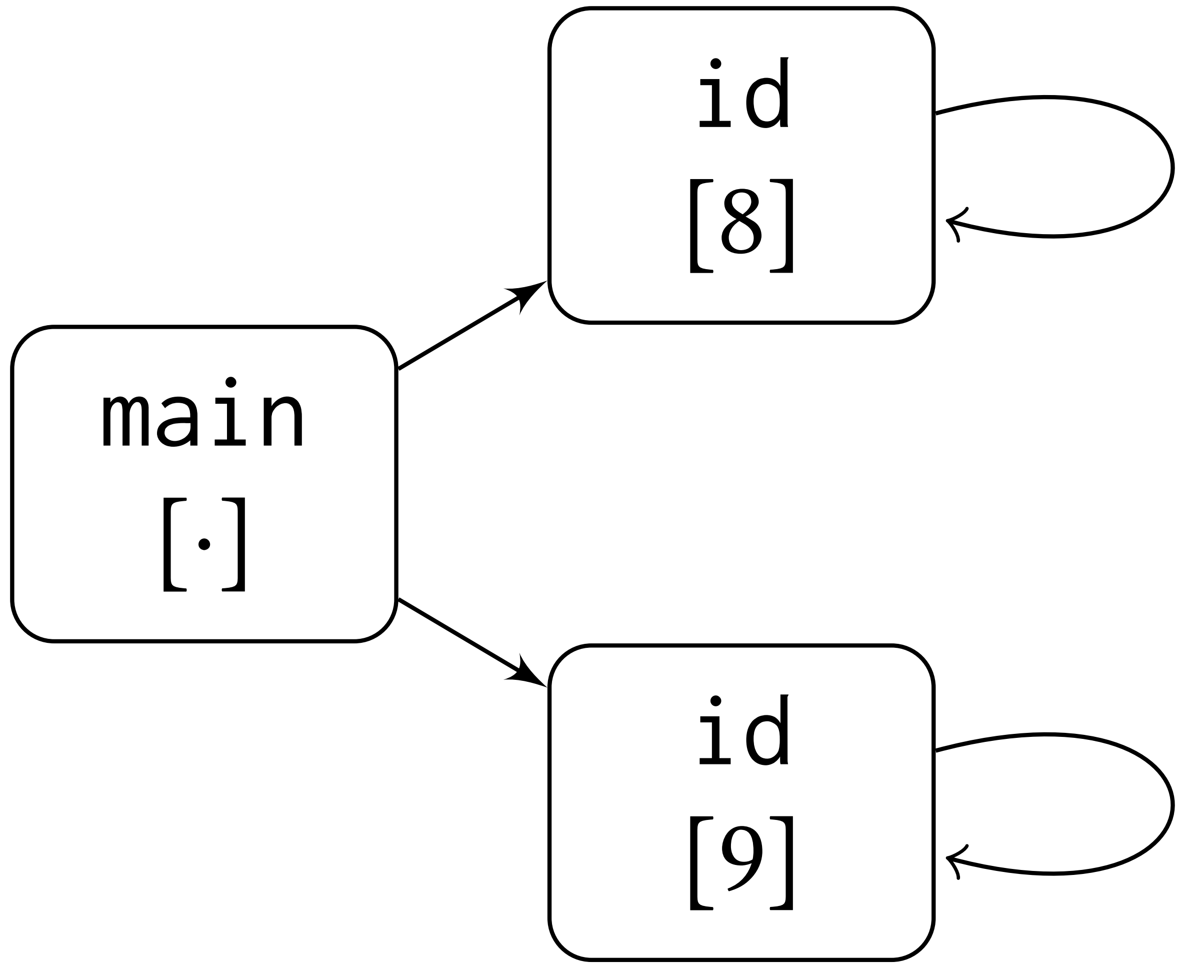
[OOPSLA'18] M. Jeon, S. Jeong, and H, Oh, **Precise and Scalable Points-to Analysis via Data-Driven Context Tunneling**

Insight from **Context Tunneling** [OOPSLA'18]

Not Important

Important

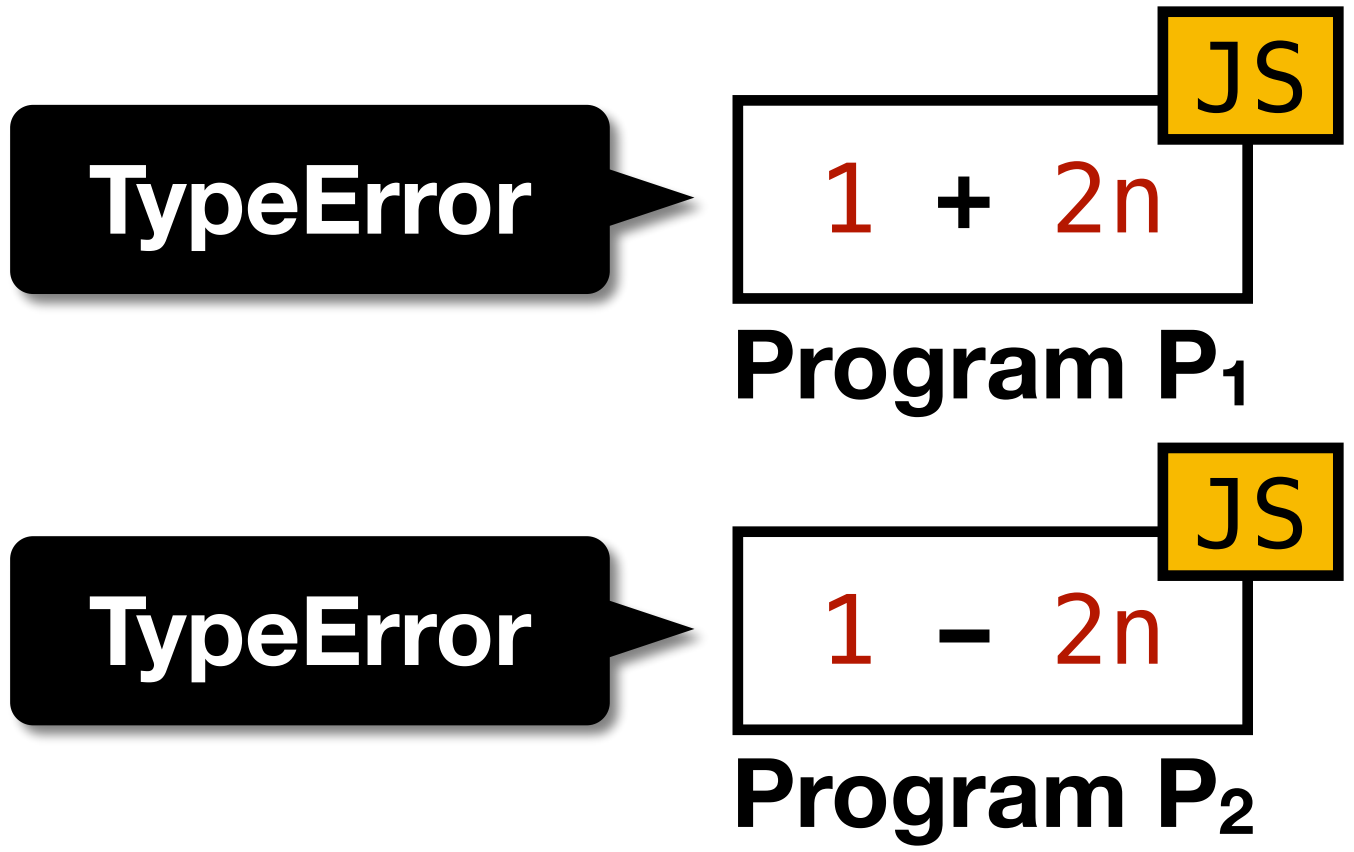
```
 1 class A {} class B {}
 2 class C {
 3   static Object id (Object v, int i){
 4       return i >= 0 ? id(v, i-1) : v;
 5   }
 6   public static void main (){
 7     int i = input();
 8     A a = (A) id(new A(), i); //Query 1
 9     B b = (B) id(new B(), i); //Query 2
10   }
11 }
```



Context Tunneling

[OOPSLA'18] M. Jeon, S. Jeong, and H, Oh, **Precise and Scalable Points-to Analysis via Data-Driven Context Tunneling**

Feature-Sensitive (FS) Coverage



TypeError

JS

```
1 + 2n
```

Program P_1

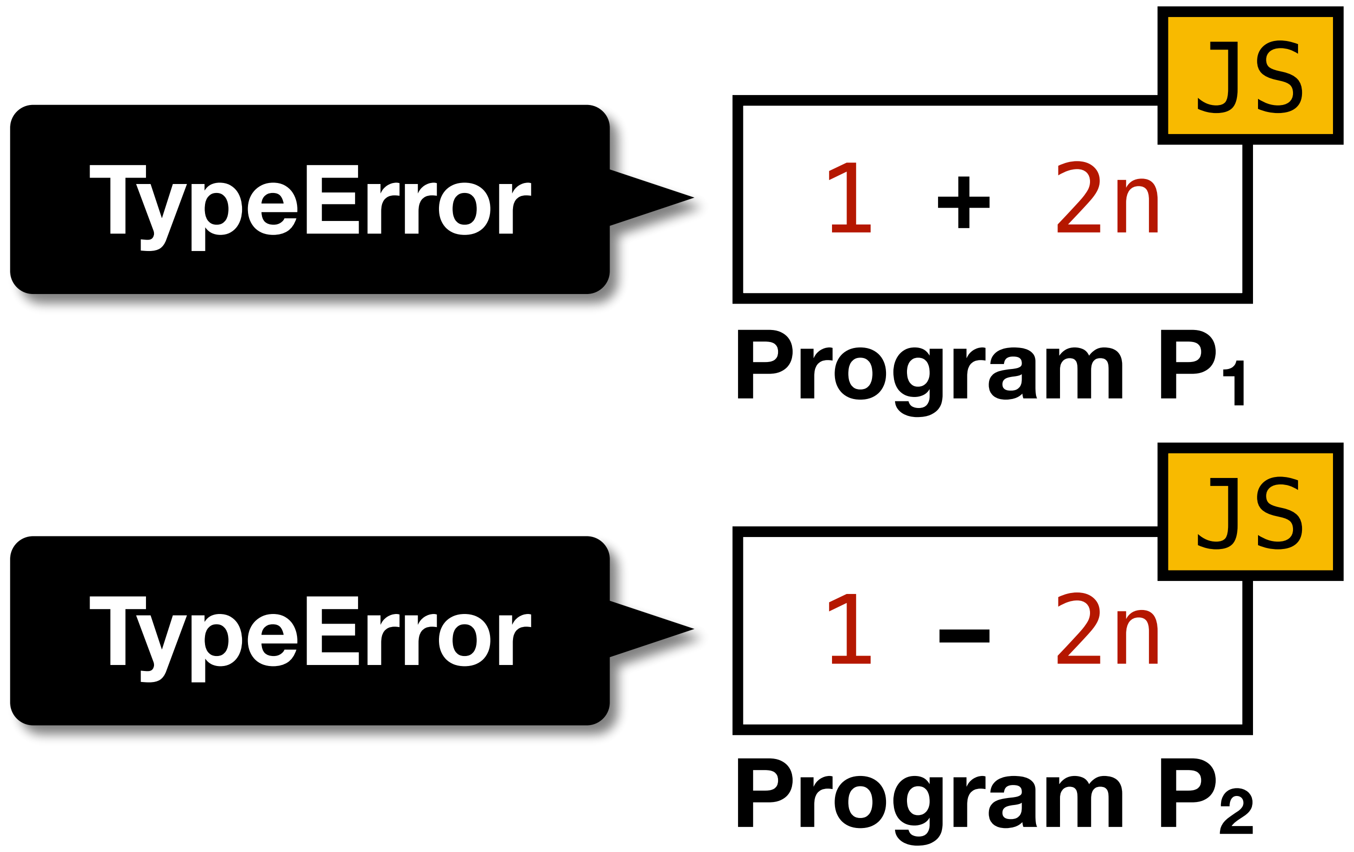
TypeError

JS

```
1 - 2n
```

Program P_2

Feature-Sensitive (FS) Coverage

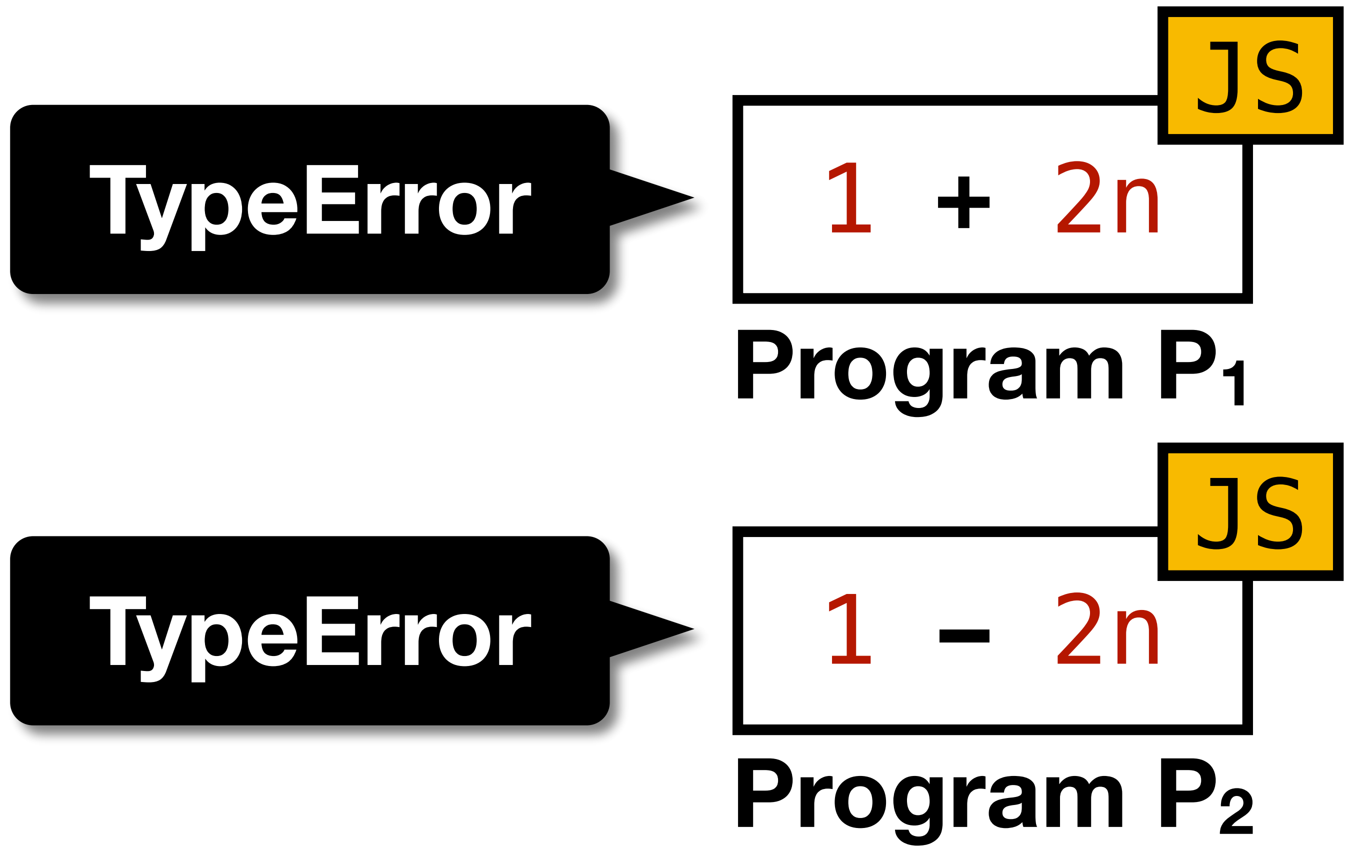


- **Feature-Sensitive (FS)** coverage criterion **divides** the given TRs with the **innermost enclosing** language **features**

FS Coverage

TR = (Feature, given **TR)**

Feature-Sensitive (FS) Coverage



TypeError

```
1 + 2n
```

Program P_1

TypeError

```
1 - 2n
```

Program P_2

- **Feature-Sensitive (FS)** coverage criterion **divides** the given TRs with the **innermost enclosing** language **features**

FS Coverage

TR = (Feature, given **TR)**

feat **ADD**

Evaluation of *AddExpr* : *AddExpr* **+** *MulExpr*

1. Return ? **EvalStrOrNumBinExpr** (*AddExpr*, **+**, *MulExpr*).

feat **SUB**

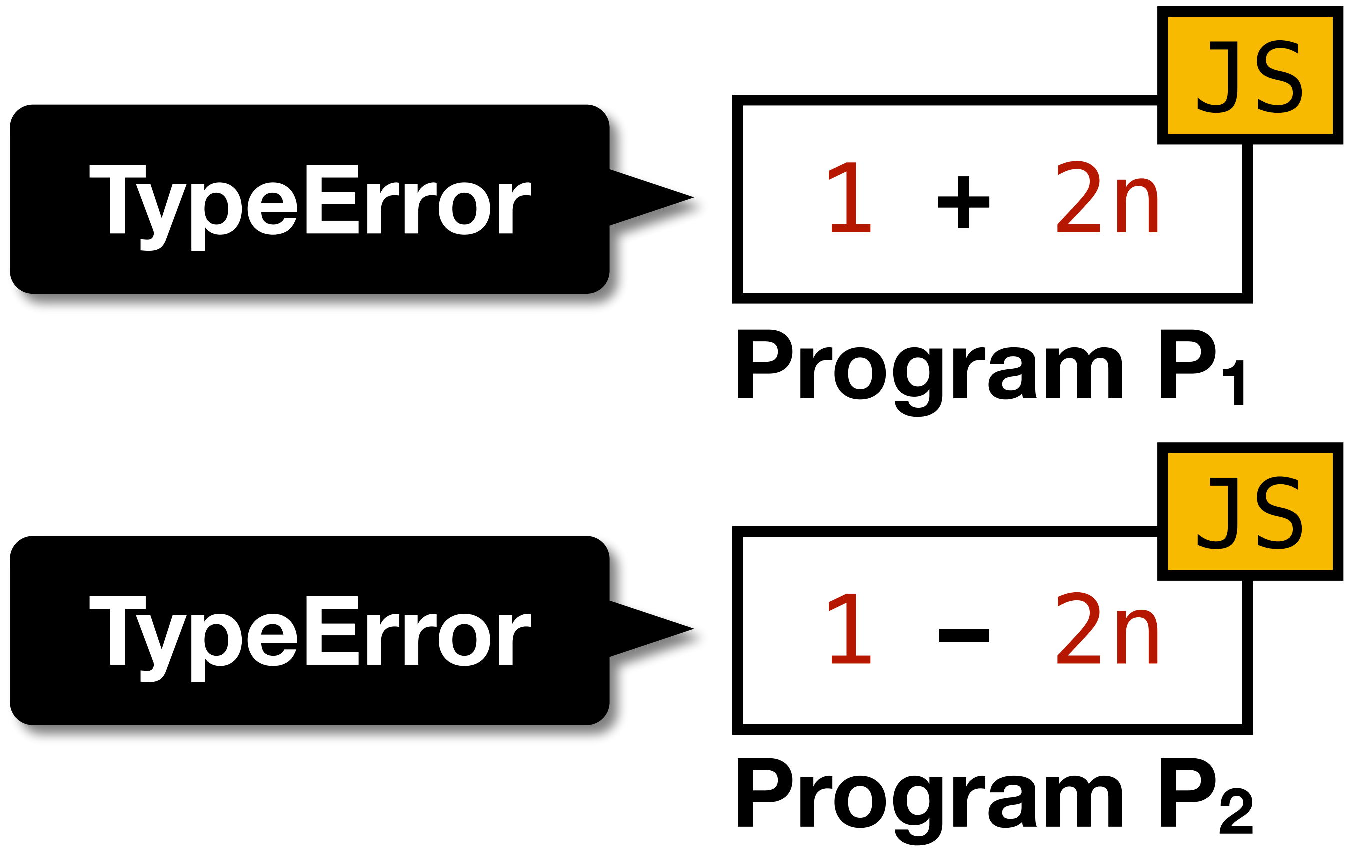
Evaluation of *AddExpr* : *AddExpr* **–** *MulExpr*

1. Return ? **EvalStrOrNumBinExpr** (*AddExpr*, **–**, *MulExpr*).

Feature-Sensitive (FS) Coverage

FS Node Coverage

TR = (Feature, Node)



TypeError

```
1 + 2n
```

JS

Program P_1

TypeError

```
1 - 2n
```

JS

Program P_2

- **Feature-Sensitive (FS)** coverage criterion **divides** the given TRs with the **innermost enclosing** language **features**

FS Coverage

TR = (Feature, given TR)

ADD feat

Evaluation of *AddExpr* : *AddExpr* **+** *MulExpr*

1. Return ? **EvalStrOrNumBinExpr** (*AddExpr*, **+**, *MulExpr*).

SUB feat

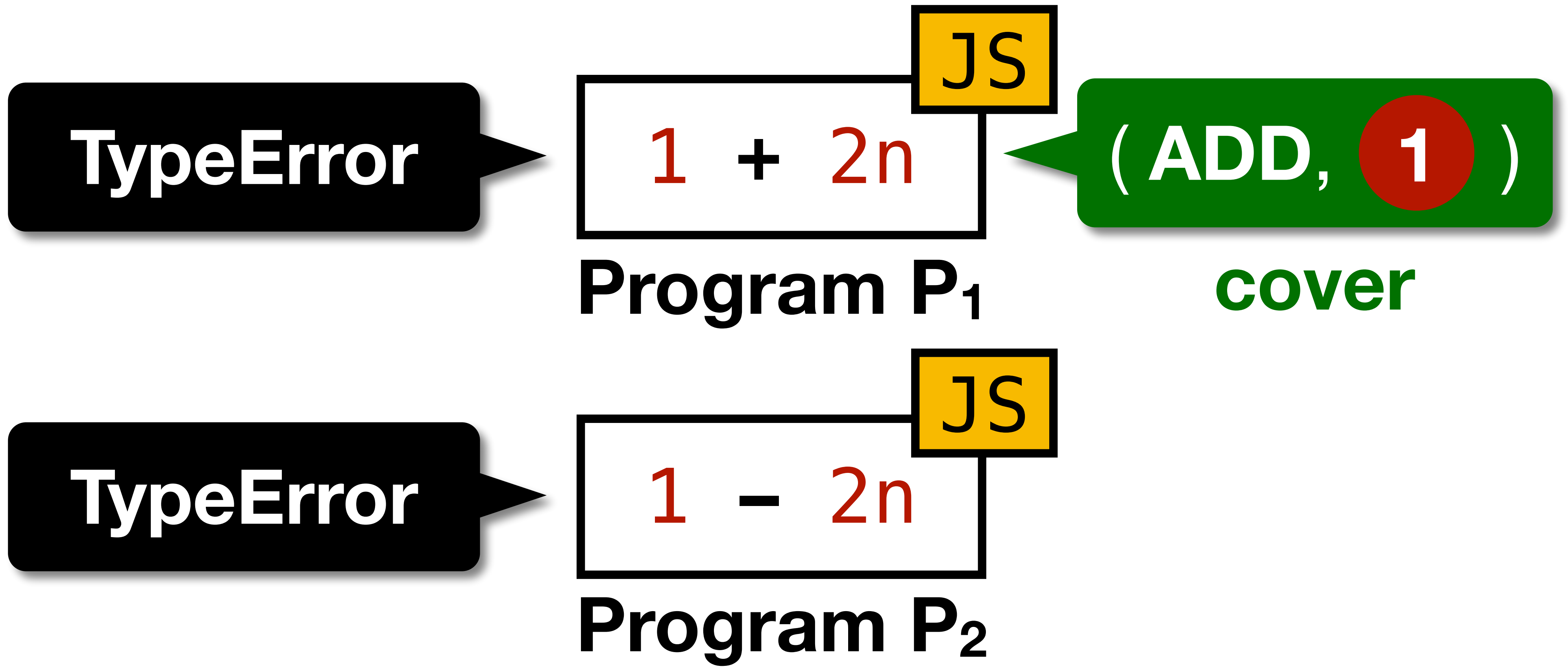
Evaluation of *AddExpr* : *AddExpr* **−** *MulExpr*

1. Return ? **EvalStrOrNumBinExpr** (*AddExpr*, **−**, *MulExpr*).

Feature-Sensitive (FS) Coverage

FS Node Coverage

TR = (Feature, Node)



- **Feature-Sensitive (FS)** coverage criterion **divides** the given TRs with the **innermost enclosing** language **features**

FS Coverage

TR = (Feature, given TR)

feat **ADD**

Evaluation of *AddExpr* : *AddExpr* **+** *MulExpr*

1. Return ? **EvalStrOrNumBinExpr** (*AddExpr*, **+**, *MulExpr*).

feat **SUB**

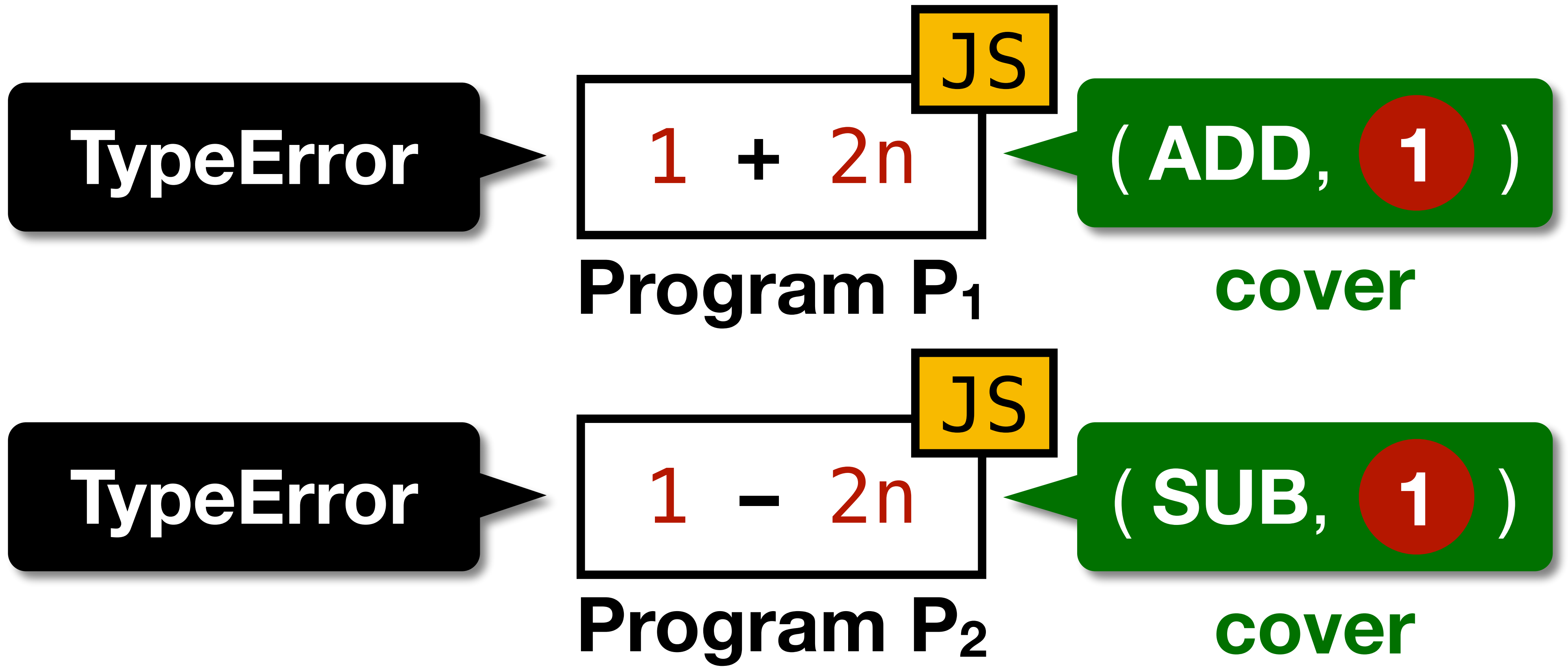
Evaluation of *AddExpr* : *AddExpr* **–** *MulExpr*

1. Return ? **EvalStrOrNumBinExpr** (*AddExpr*, **–**, *MulExpr*).

Feature-Sensitive (FS) Coverage

FS Node Coverage

TR = (Feature, Node)


TypeError

JS

1 + 2n

Program P_1

(ADD, 1)

cover

TypeError

JS

1 – 2n

Program P_2

(SUB, 1)

cover

- **Feature-Sensitive (FS)** coverage criterion **divides** the given TRs with the **innermost enclosing** language **features**

FS Coverage

TR = (Feature, given TR)

`ADD` `feat`

Evaluation of *AddExpr* : *AddExpr* **+** *MulExpr*

1. Return ? **EvalStrOrNumBinExpr** (*AddExpr*, **+**, *MulExpr*).

`SUB` `feat`

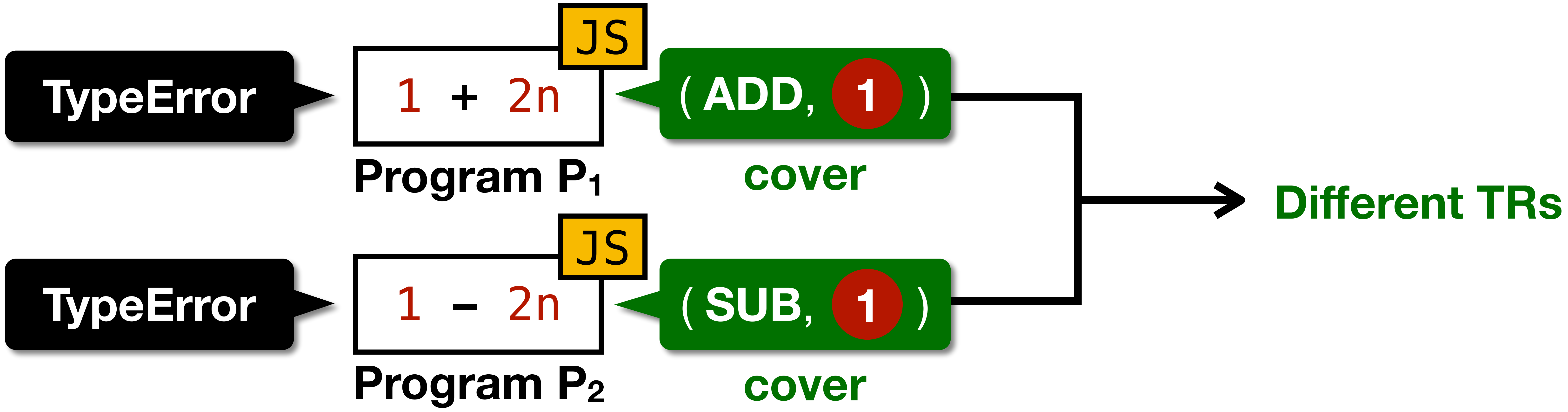
Evaluation of *AddExpr* : *AddExpr* **–** *MulExpr*

1. Return ? **EvalStrOrNumBinExpr** (*AddExpr*, **–**, *MulExpr*).

Feature-Sensitive (FS) Coverage

FS Node Coverage

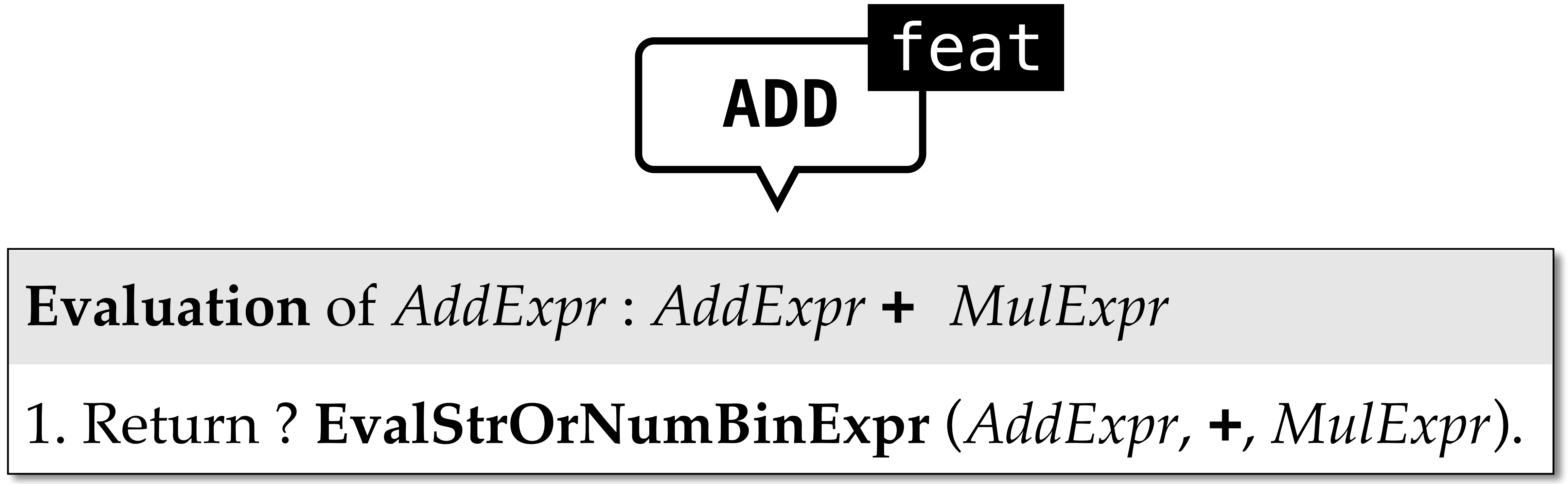
TR = **(Feature, Node)**



- **Feature-Sensitive (FS)** coverage criterion **divides** the given TRs with the **innermost enclosing** language **features**

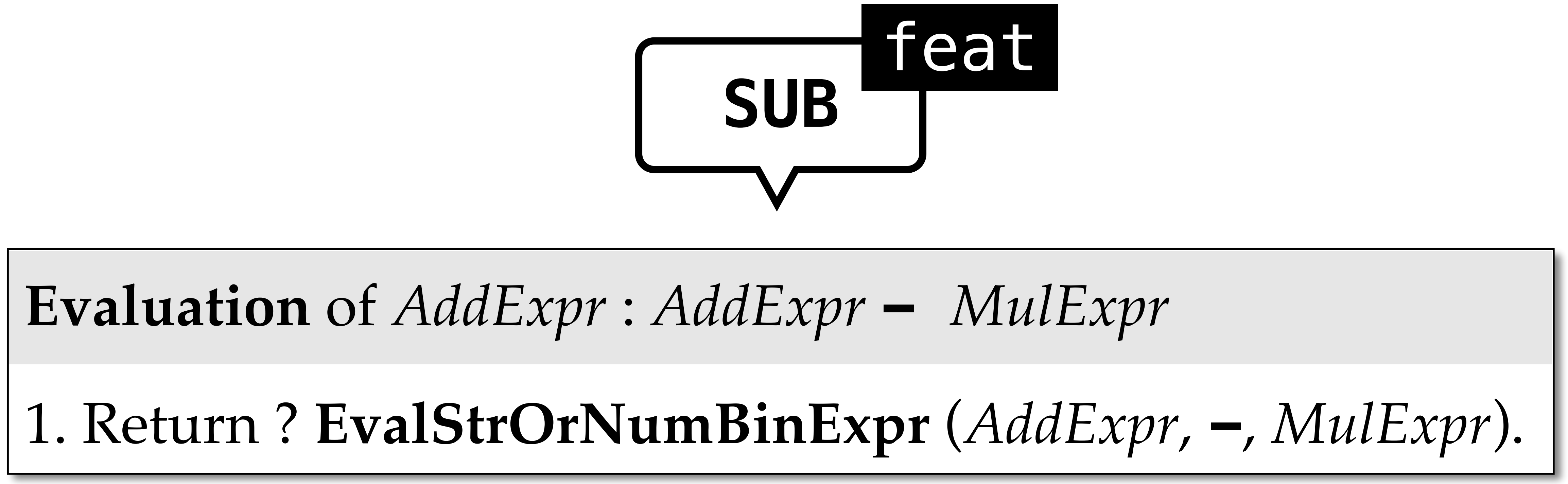
FS Coverage

TR = (Feature, given TR)



Evaluation of *AddExpr* : *AddExpr* **+** *MulExpr*

1. Return ? **EvalStrOrNumBinExpr** (*AddExpr*, **+**, *MulExpr*).



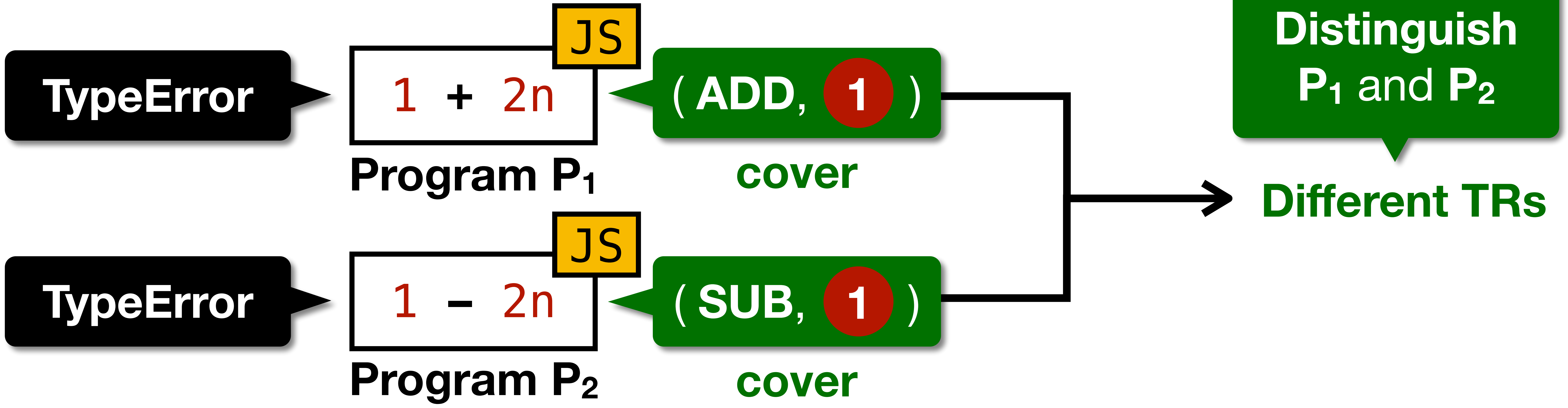
Evaluation of *AddExpr* : *AddExpr* **–** *MulExpr*

1. Return ? **EvalStrOrNumBinExpr** (*AddExpr*, **–**, *MulExpr*).

Feature-Sensitive (FS) Coverage

FS Node Coverage

TR = (Feature, Node)



- **Feature-Sensitive (FS)** coverage criterion **divides** the given TRs with the **innermost enclosing** language **features**

FS Coverage

TR = (Feature, given TR)

feat **ADD**

Evaluation of *AddExpr* : *AddExpr* **+** *MulExpr*

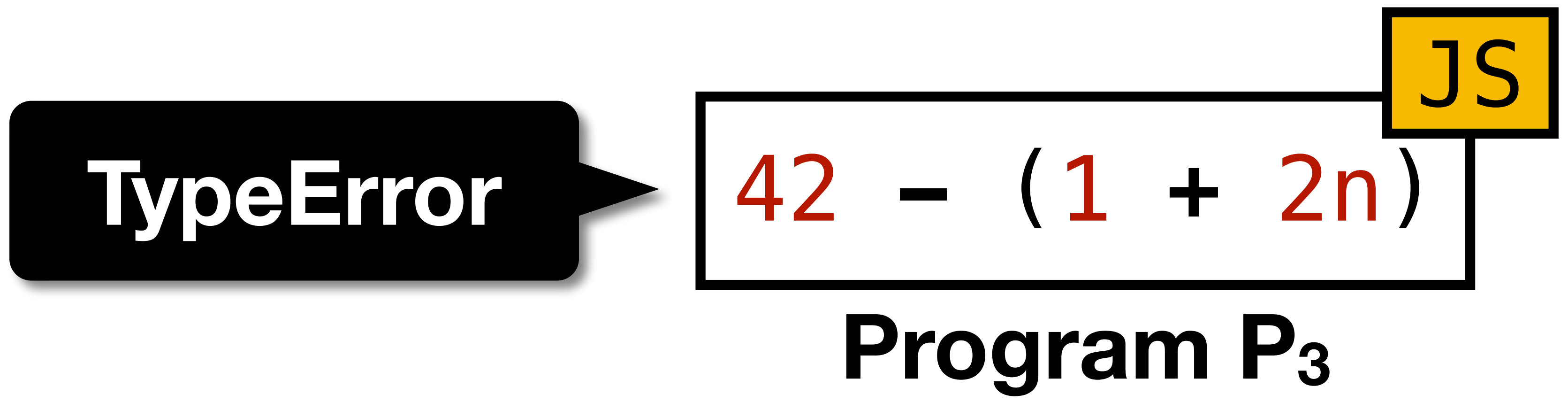
1. Return ? **EvalStrOrNumBinExpr** (*AddExpr*, **+**, *MulExpr*).

feat **SUB**

Evaluation of *AddExpr* : *AddExpr* **–** *MulExpr*

1. Return ? **EvalStrOrNumBinExpr** (*AddExpr*, **–**, *MulExpr*).

k-**Feature-Sensitive (**k**-FS)** Coverage



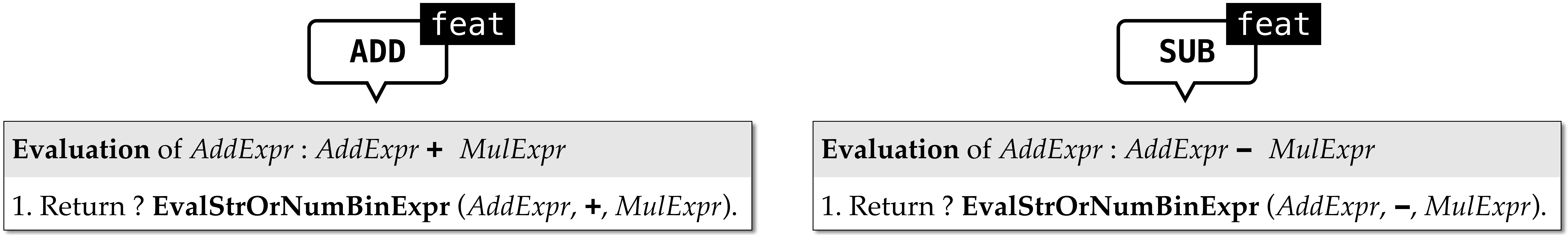
```
42 - (1 + 2n)
```

Program P_3

- k-**F**eature-**S**ensitive (**k-FS**) coverage criterion **divides** the given TRs with at most k**-innermost enclosing** language **features**

k-FS Coverage

TR = (Feature$^{\leq k}$, given **TR)**



Evaluation of *AddExpr* : *AddExpr* **+** *MulExpr*

1. Return ? **EvalStrOrNumBinExpr** (*AddExpr*, **+**, *MulExpr*).

Evaluation of *AddExpr* : *AddExpr* **–** *MulExpr*

1. Return ? **EvalStrOrNumBinExpr** (*AddExpr*, **–**, *MulExpr*).

k-Feature-Sensitive (k-FS) Coverage

2-FS Node Coverage

TR = **(Feature$^{\le 2}$, Node)**

TypeError

JS

```
42 - (1 + 2n)
```

Program P_3

- k-Feature-Sensitive (k-FS) coverage criterion **divides** the given TRs with at most k**-innermost enclosing** language **features**

k-FS Coverage

TR = (Feature$^{\le k}$, given **TR)**

ADD feat

Evaluation of *AddExpr* : *AddExpr* **+** *MulExpr*

1. Return ? **EvalStrOrNumBinExpr** (*AddExpr*, **+**, *MulExpr*).

SUB feat

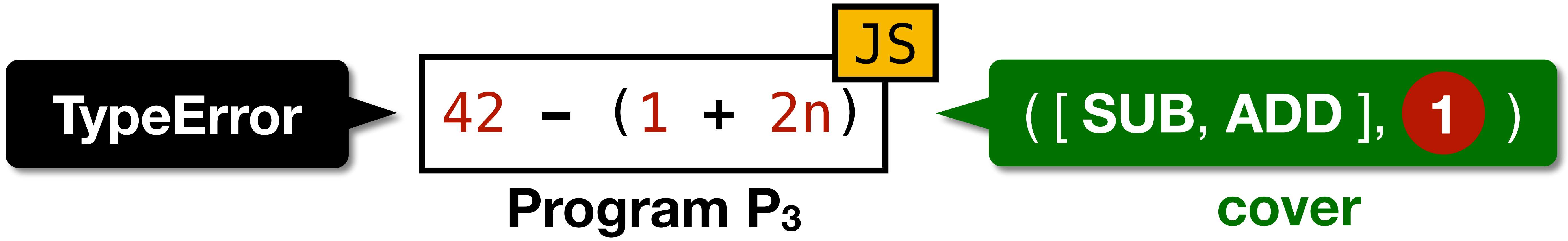
Evaluation of *AddExpr* : *AddExpr* **–** *MulExpr*

1. Return ? **EvalStrOrNumBinExpr** (*AddExpr*, **–**, *MulExpr*).

k-Feature-Sensitive (k-FS) Coverage

2-FS Node Coverage

TR = **(Feature$^{\leq 2}$, Node)**



TypeError

```
42 - (1 + 2n)
```

JS

Program P$_3$

([**SUB**, **ADD**], **1**)

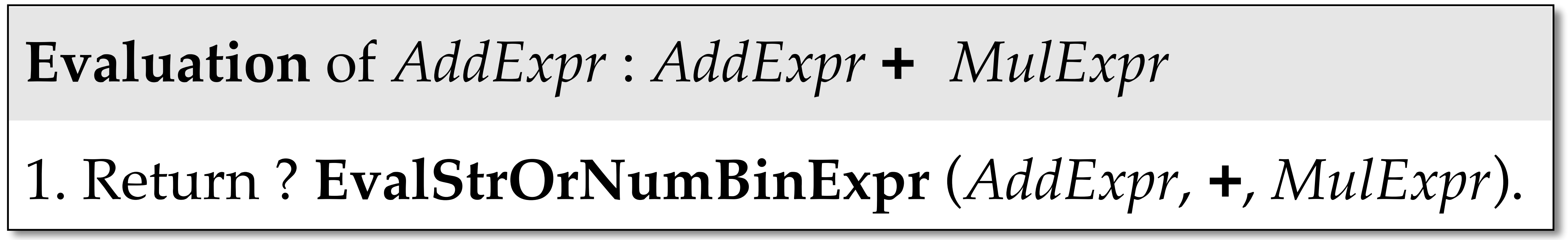
cover

- k-**F**eature-**S**ensitive (**k-FS**) coverage criterion **divides** the given TRs with at most **k-innermost enclosing** language **features**

k-FS Coverage

TR = (Feature$^{\leq k}$, given **TR)**

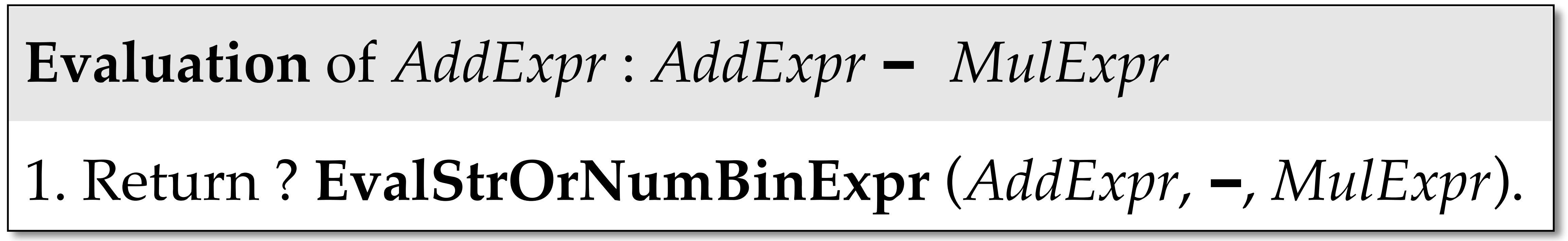
`feat` **ADD**



Evaluation of *AddExpr* : *AddExpr* **+** *MulExpr*

1. Return ? **EvalStrOrNumBinExpr** (*AddExpr*, **+**, *MulExpr*).

`feat` **SUB**



Evaluation of *AddExpr* : *AddExpr* **–** *MulExpr*

1. Return ? **EvalStrOrNumBinExpr** (*AddExpr*, **–**, *MulExpr*).

Motivating Example 2

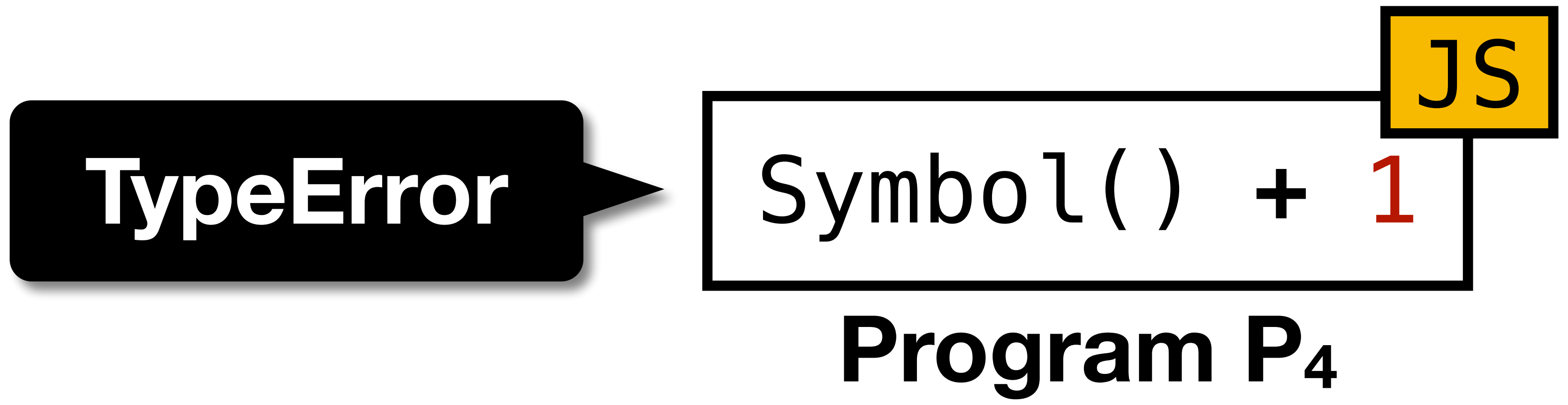
TypeError

JS

```
Symbol() + 1
```

Program P_4

Motivating Example 2

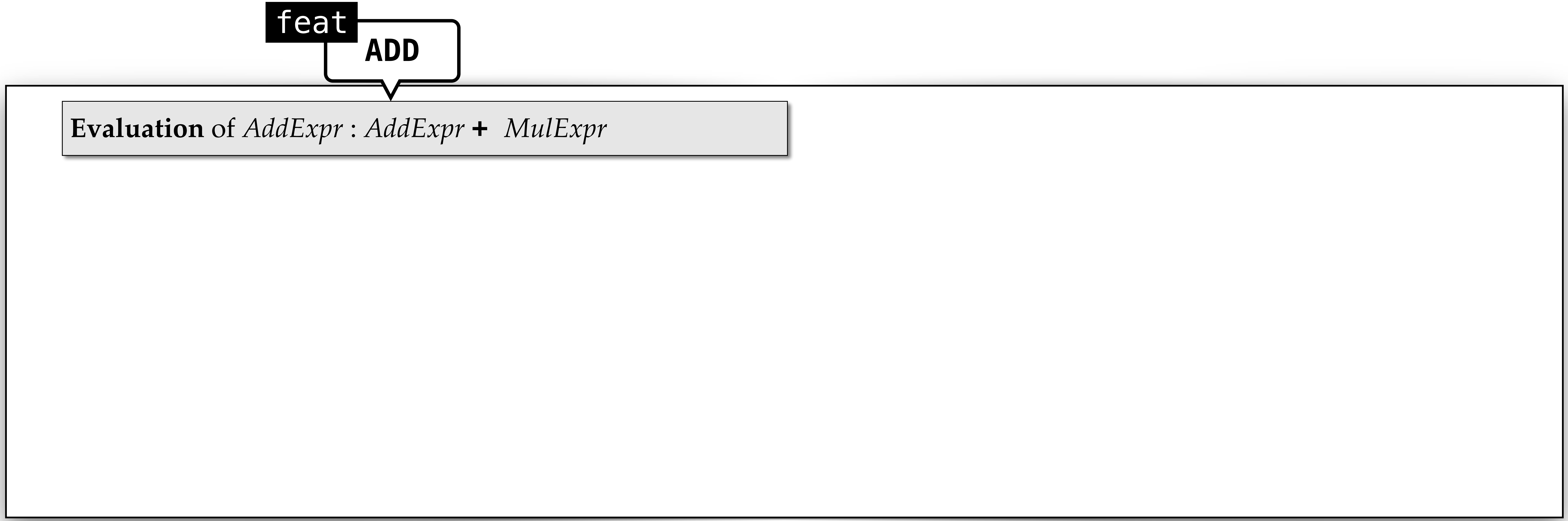


TypeError

JS

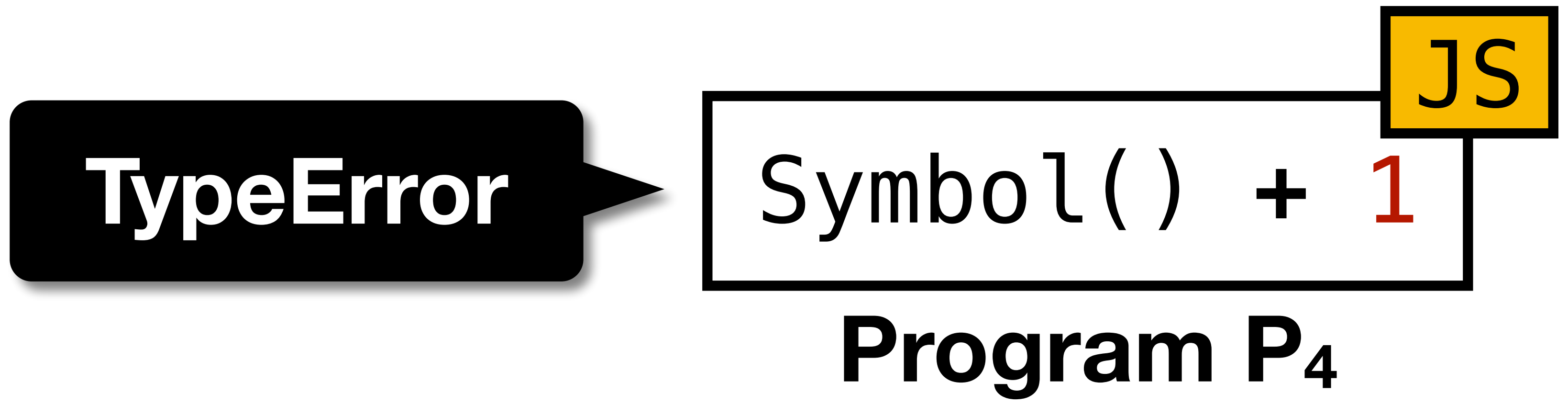
```
Symbol() + 1
```

Program P_4



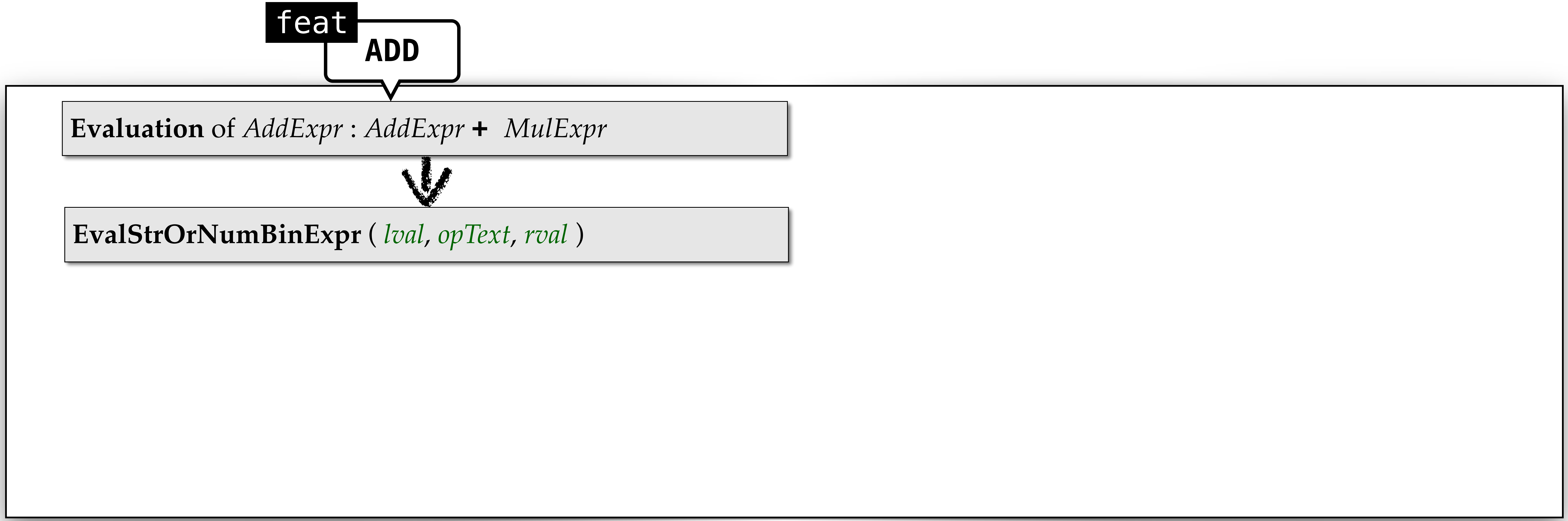
Evaluation of *AddExpr* : *AddExpr* **+** *MulExpr*

Motivating Example 2



```
Symbol() + 1
```

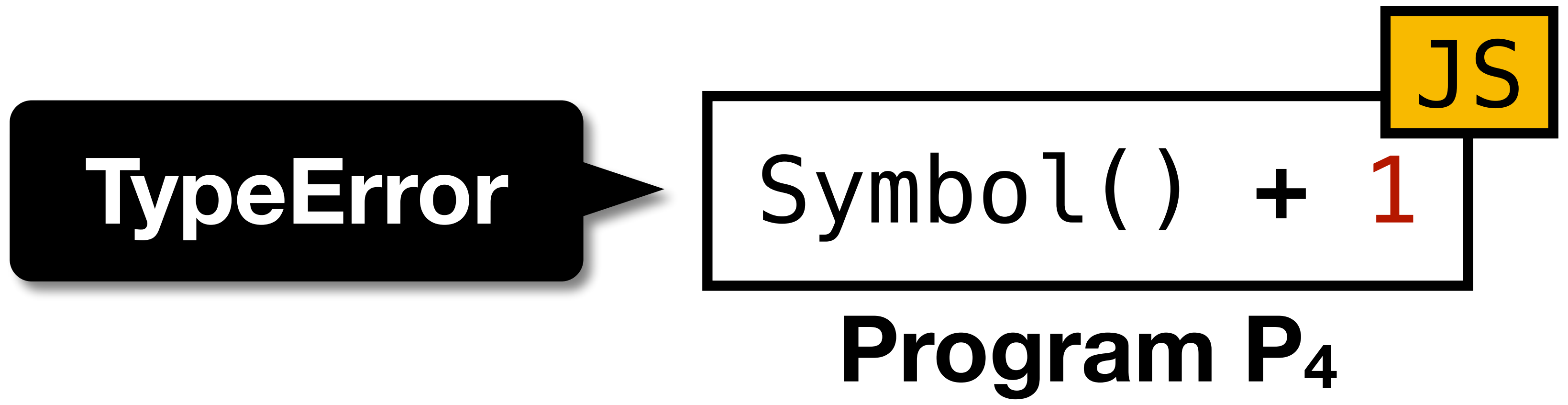
Program P_4



Evaluation of *AddExpr* : *AddExpr* **+** *MulExpr*

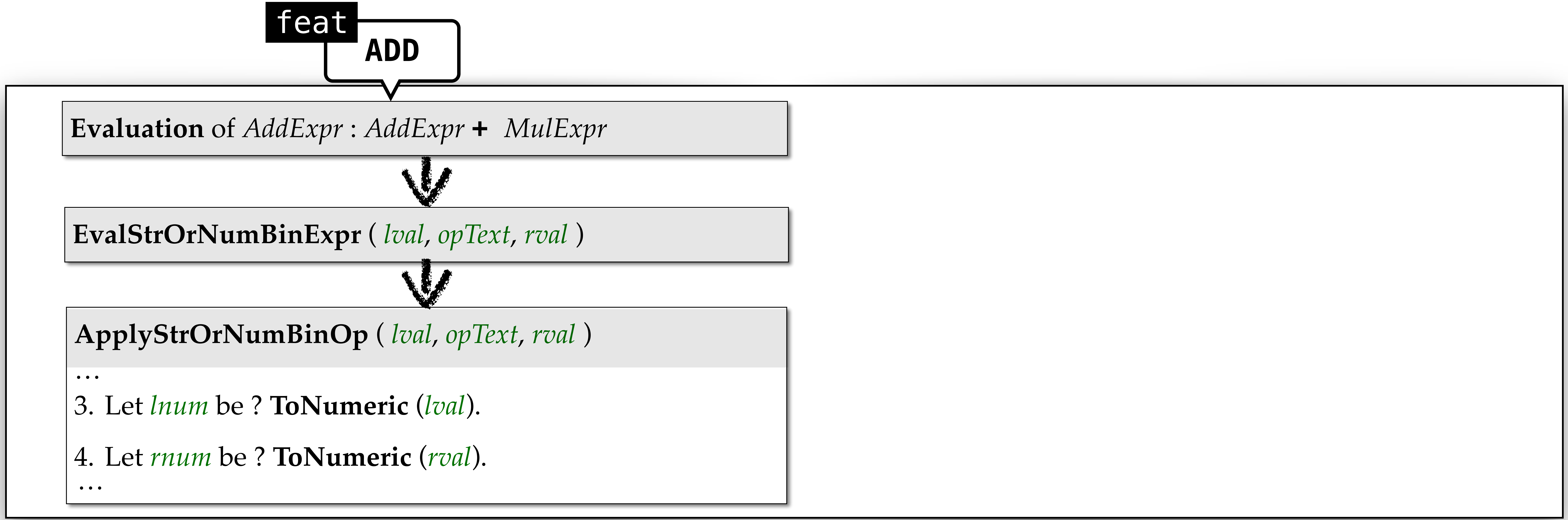
EvalStrOrNumBinExpr (*lval*, *opText*, *rval*)

Motivating Example 2



```
Symbol() + 1
```

Program P_4



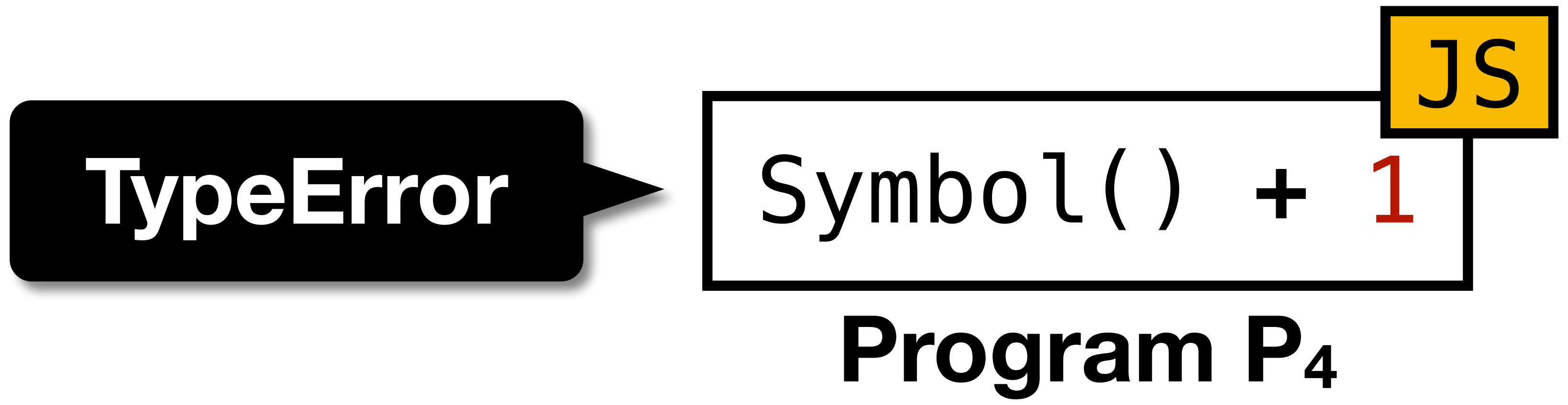
Evaluation of *AddExpr* : *AddExpr* **+** *MulExpr*

EvalStrOrNumBinExpr (*lval*, *opText*, *rval*)

ApplyStrOrNumBinOp (*lval*, *opText*, *rval*)

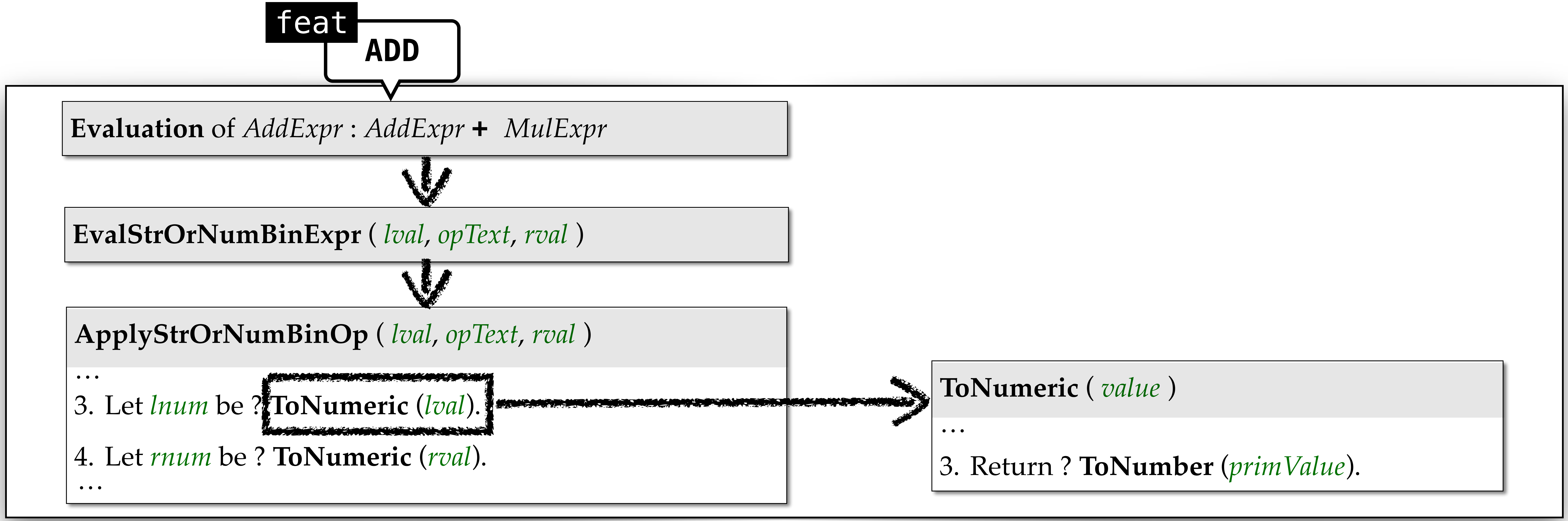
```
…
3. Let lnum be ? ToNumeric (lval).
4. Let rnum be ? ToNumeric (rval).
…
```

Motivating Example 2



```
Symbol() + 1
```

Program P$_4$



Evaluation of *AddExpr* : *AddExpr* **+** *MulExpr*

EvalStrOrNumBinExpr (*lval*, *opText*, *rval*)

ApplyStrOrNumBinOp (*lval*, *opText*, *rval*)

```
…
3. Let lnum be ? ToNumeric (lval).
4. Let rnum be ? ToNumeric (rval).
…
```

ToNumeric (*value*)

```
…
3. Return ? ToNumber (primValue).
```

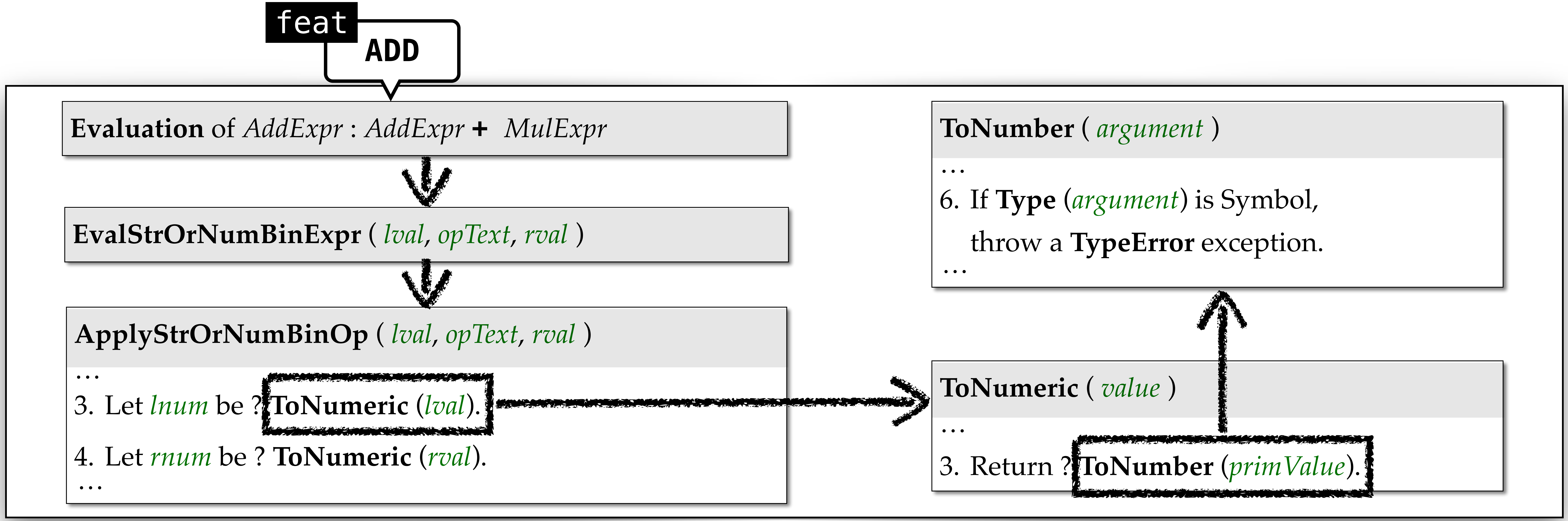
Motivating Example 2

TypeError

JS

```
Symbol() + 1
```

Program P_4



feat **ADD**

Evaluation of *AddExpr* : *AddExpr* **+** *MulExpr*

EvalStrOrNumBinExpr (*lval*, *opText*, *rval*)

ApplyStrOrNumBinOp (*lval*, *opText*, *rval*)

…

3. Let *lnum* be ? **ToNumeric** (*lval*).

4. Let *rnum* be ? **ToNumeric** (*rval*).

…

ToNumeric (*value*)

…

3. Return ? **ToNumber** (*primValue*).

ToNumber (*argument*)

…

6. If **Type** (*argument*) is Symbol,
   throw a **TypeError** exception.

…

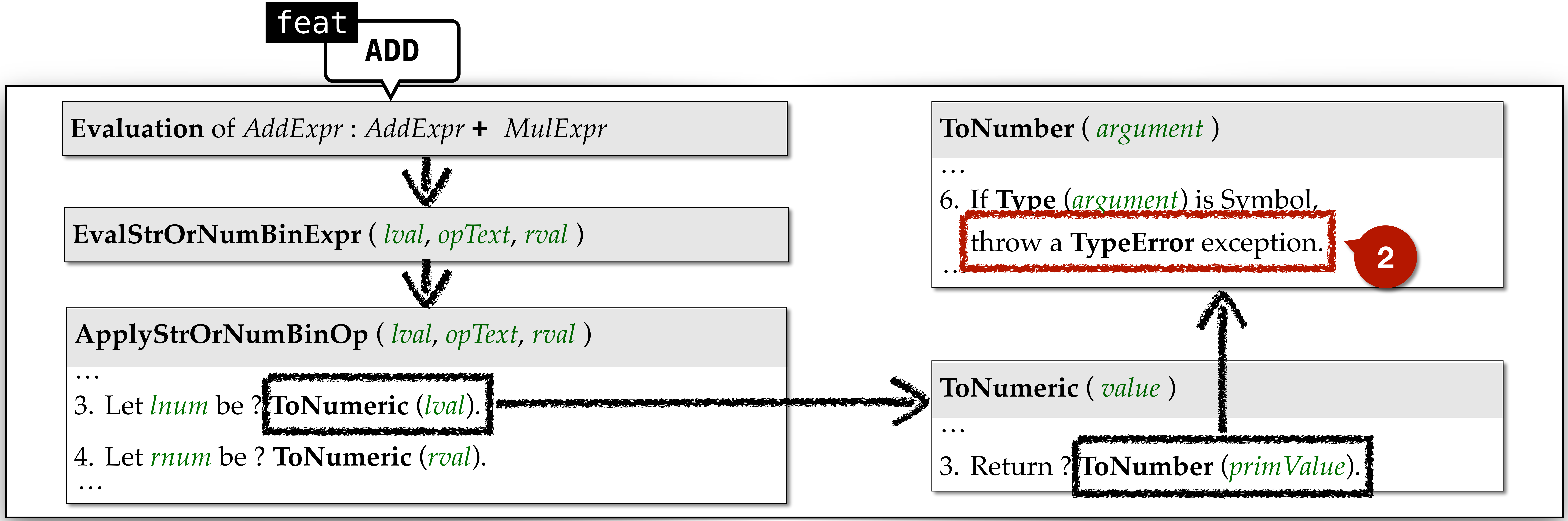
Motivating Example 2

TypeError ▶

```
Symbol() + 1
```

JS

Program P_4



feat

ADD

Evaluation of *AddExpr* : *AddExpr* **+** *MulExpr*

EvalStrOrNumBinExpr (*lval*, *opText*, *rval*)

ApplyStrOrNumBinOp (*lval*, *opText*, *rval*)

```
…
3. Let lnum be ? ToNumeric (lval).
4. Let rnum be ? ToNumeric (rval).
…
```

ToNumeric (*value*)

```
…
3. Return ? ToNumber (primValue).
```

ToNumber (*argument*)

```
…
6. If Type (argument) is Symbol,
   throw a TypeError exception.
…
```

2

Motivating Example 2

1-FS Node Coverage

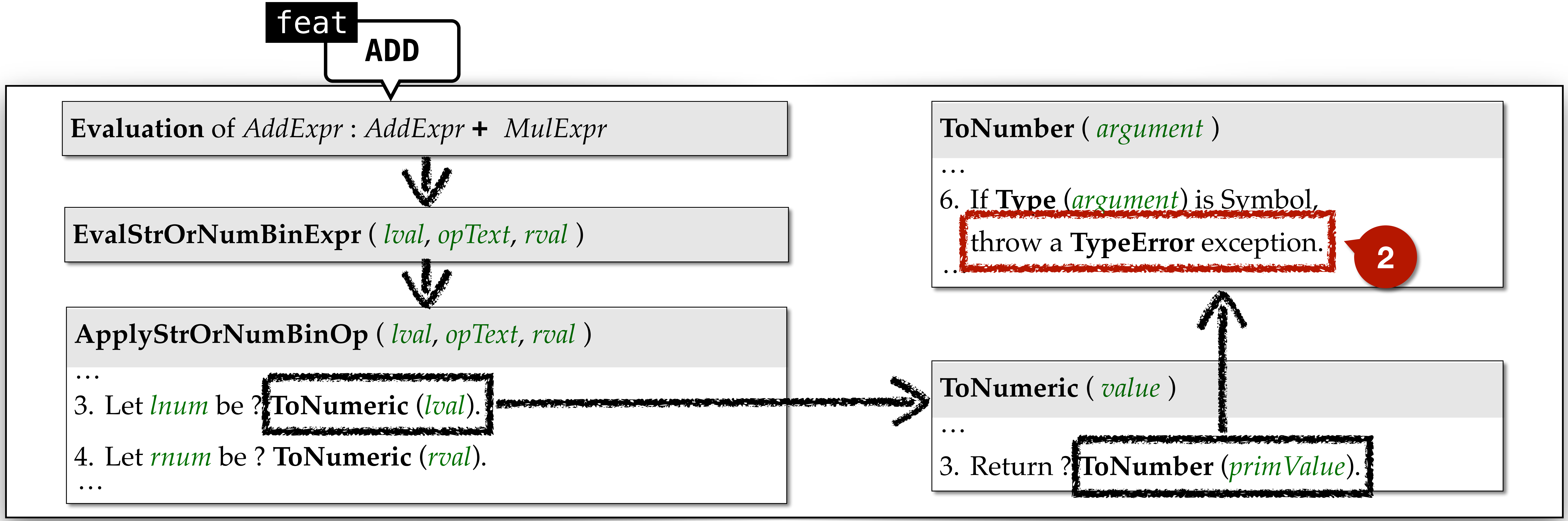
TR = (Feature, Node)

TypeError

JS

```
Symbol() + 1
```

Program P_4



Evaluation of *AddExpr* : *AddExpr* **+** *MulExpr*

EvalStrOrNumBinExpr (*lval*, *opText*, *rval*)

ApplyStrOrNumBinOp (*lval*, *opText*, *rval*)

```
…
3. Let lnum be ? ToNumeric (lval).
4. Let rnum be ? ToNumeric (rval).
…
```

ToNumeric (*value*)

```
…
3. Return ? ToNumber (primValue).
```

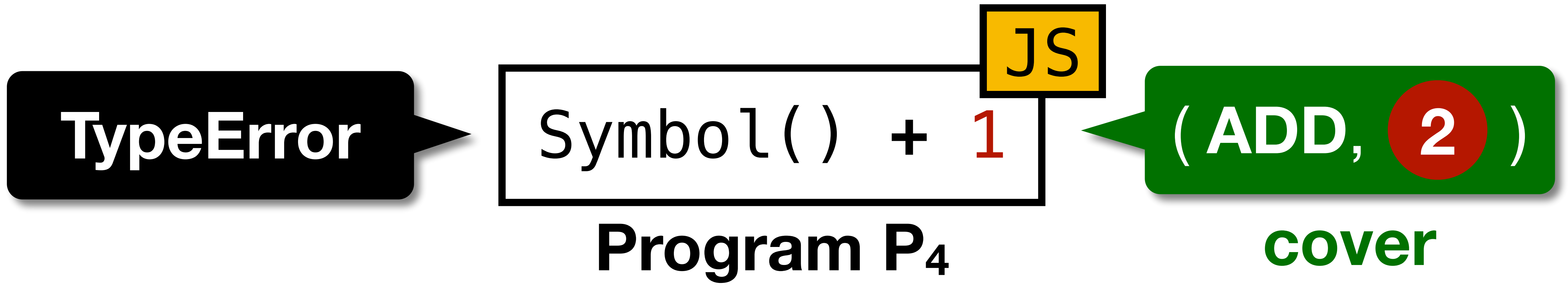
ToNumber (*argument*)

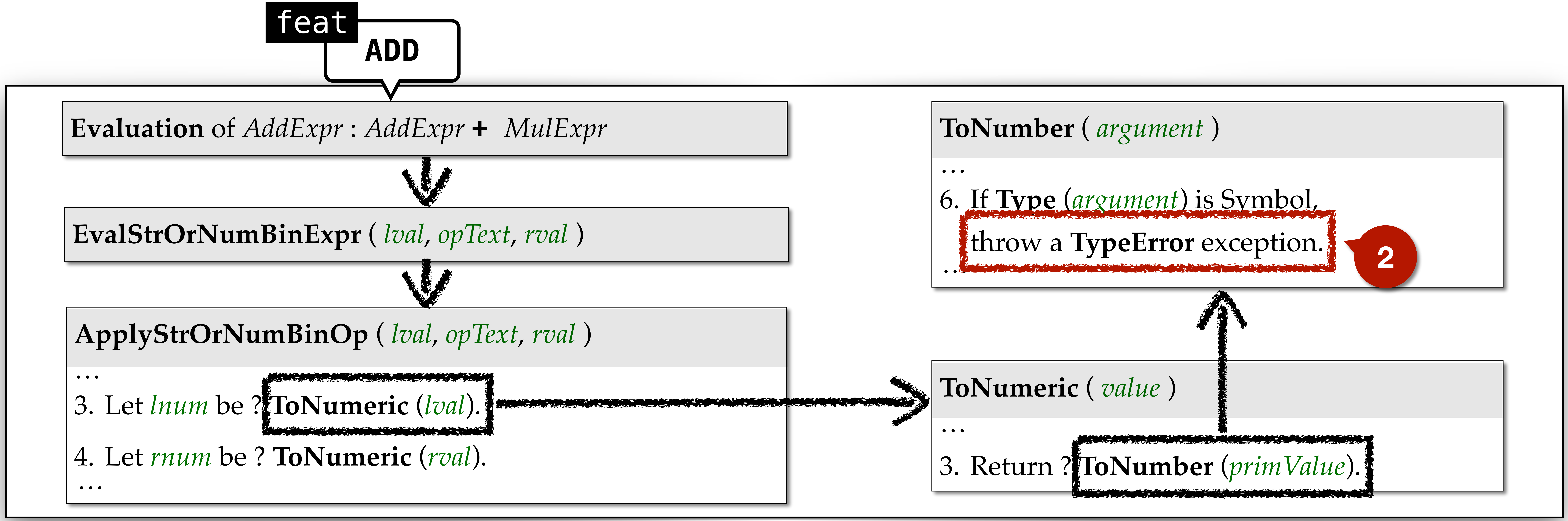
```
…
6. If Type (argument) is Symbol,
   throw a TypeError exception.
…
```

Motivating Example 2

1-FS Node Coverage

TR = (Feature, Node)





Evaluation of *AddExpr* : *AddExpr* **+** *MulExpr*

EvalStrOrNumBinExpr (*lval*, *opText*, *rval*)

ApplyStrOrNumBinOp (*lval*, *opText*, *rval*)

…

3. Let *lnum* be ? **ToNumeric** (*lval*).

4. Let *rnum* be ? **ToNumeric** (*rval*).

…

ToNumeric (*value*)

…

3. Return ? **ToNumber** (*primValue*).

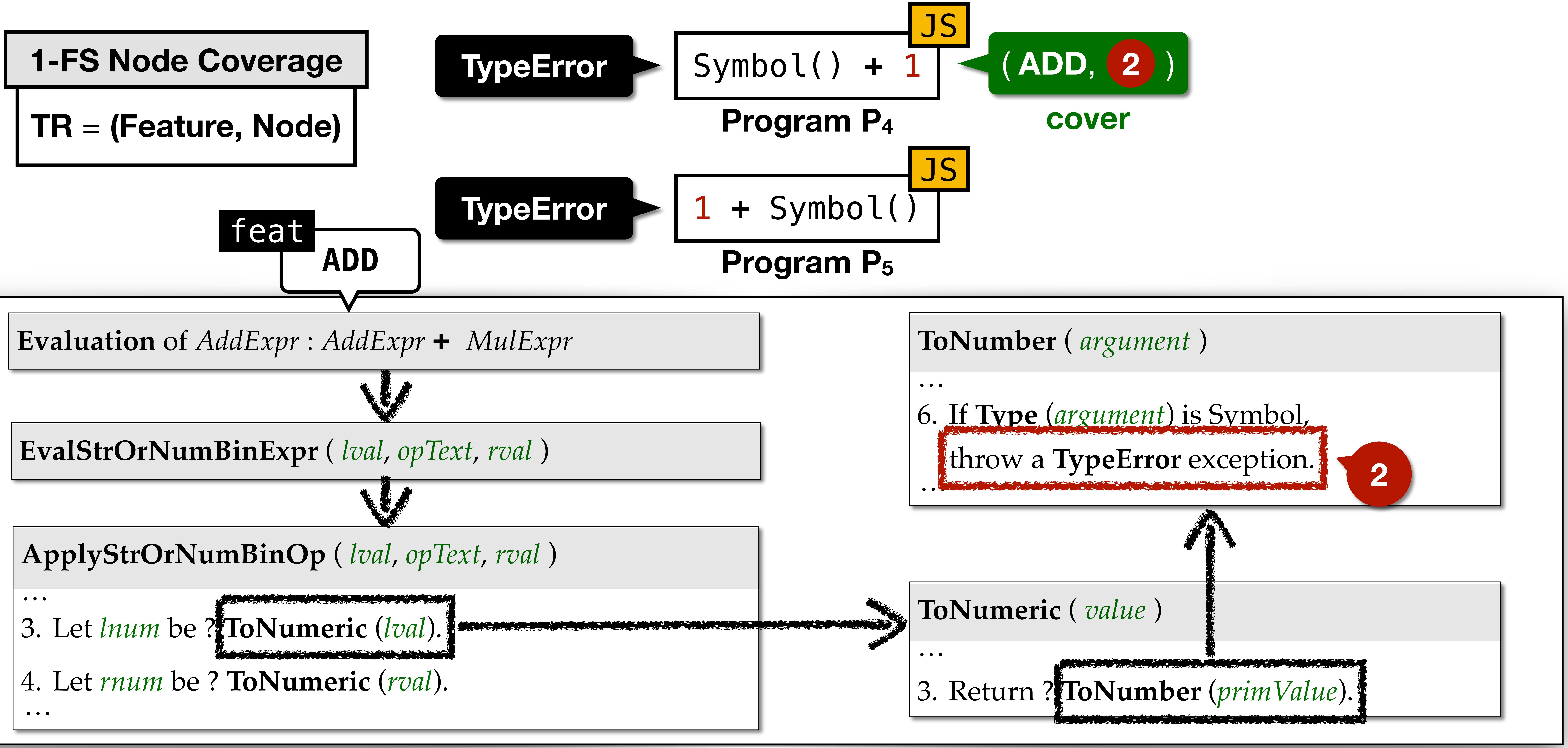
ToNumber (*argument*)

…

6. If **Type** (*argument*) is Symbol,
throw a **TypeError** exception.

…

Motivating Example 2



1-FS Node Coverage

TR = (Feature, Node)

```
Symbol() + 1
```

Program P_4

```
1 + Symbol()
```

Program P_5

Evaluation of *AddExpr* : *AddExpr* **+** *MulExpr*

EvalStrOrNumBinExpr (*lval*, *opText*, *rval*)

ApplyStrOrNumBinOp (*lval*, *opText*, *rval*)

…

3. Let *lnum* be ? **ToNumeric** (*lval*).

4. Let *rnum* be ? **ToNumeric** (*rval*).

…

ToNumeric (*value*)

…

3. Return ? **ToNumber** (*primValue*).

ToNumber (*argument*)

…

6. If **Type** (*argument*) is Symbol,
   throw a **TypeError** exception.

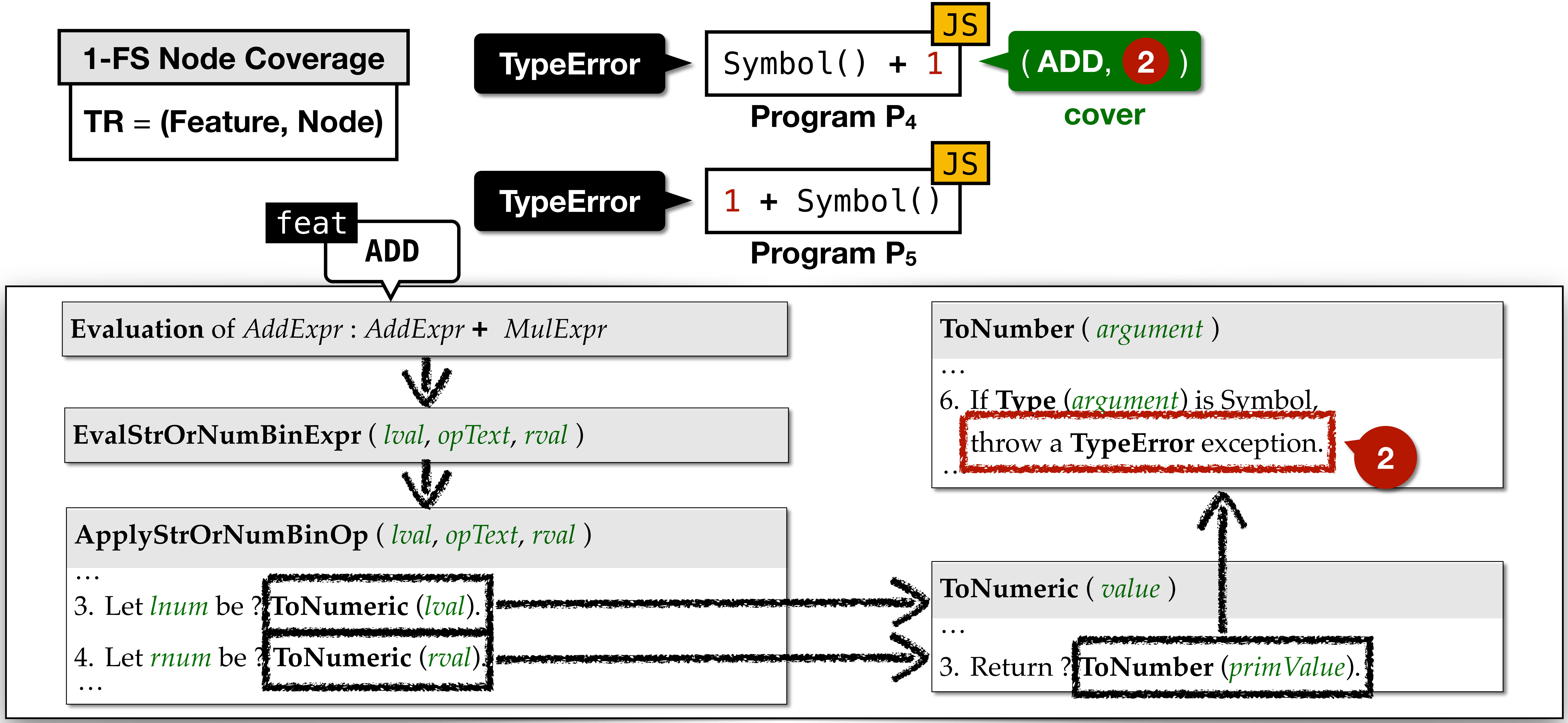
…

PLRG Abstract Algorithms in **ECMA-262** (ES13, 2022), **JavaScript Language Specification**

Motivating Example 2

1-FS Node Coverage

TR = (Feature, Node)



Evaluation of *AddExpr* : *AddExpr* **+** *MulExpr*

EvalStrOrNumBinExpr (*lval*, *opText*, *rval*)

ApplyStrOrNumBinOp (*lval*, *opText*, *rval*)

…
3. Let *lnum* be ? **ToNumeric** (*lval*).
4. Let *rnum* be ? **ToNumeric** (*rval*).
…

ToNumeric (*value*)

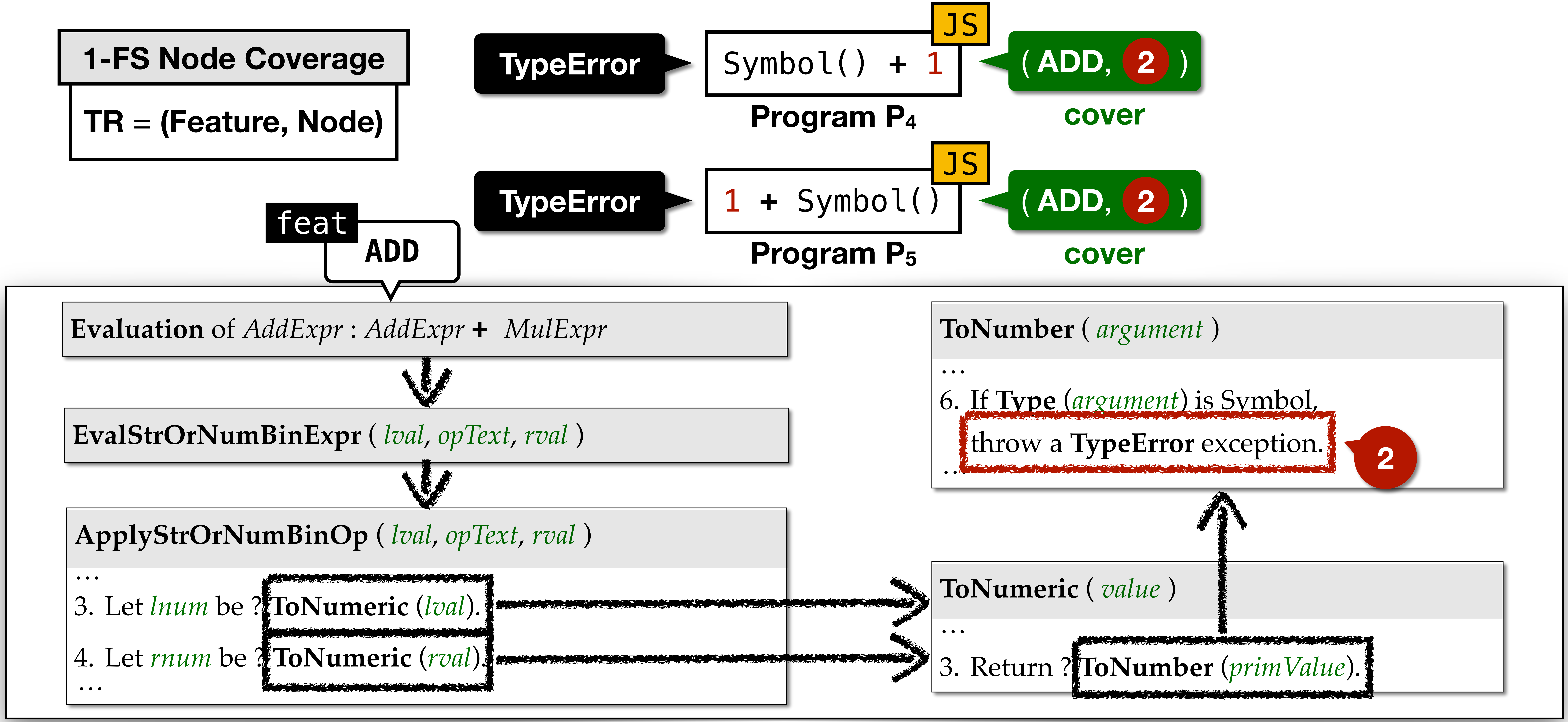
…
3. Return ? **ToNumber** (*primValue*).

ToNumber (*argument*)

…
6. If **Type** (*argument*) is Symbol,
throw a **TypeError** exception.
…

Abstract Algorithms in **ECMA-262** (ES13, 2022), **JavaScript Language Specification**

Motivating Example 2



1-FS Node Coverage

TR = (Feature, Node)

```
Symbol() + 1
```

Program P$_4$

```
1 + Symbol()
```

Program P$_5$

Evaluation of *AddExpr* : *AddExpr* **+** *MulExpr*

EvalStrOrNumBinExpr (*lval*, *opText*, *rval*)

ApplyStrOrNumBinOp (*lval*, *opText*, *rval*)

…

3. Let *lnum* be ? **ToNumeric** (*lval*).

4. Let *rnum* be ? **ToNumeric** (*rval*).

…

ToNumeric (*value*)

…

3. Return ? **ToNumber** (*primValue*).

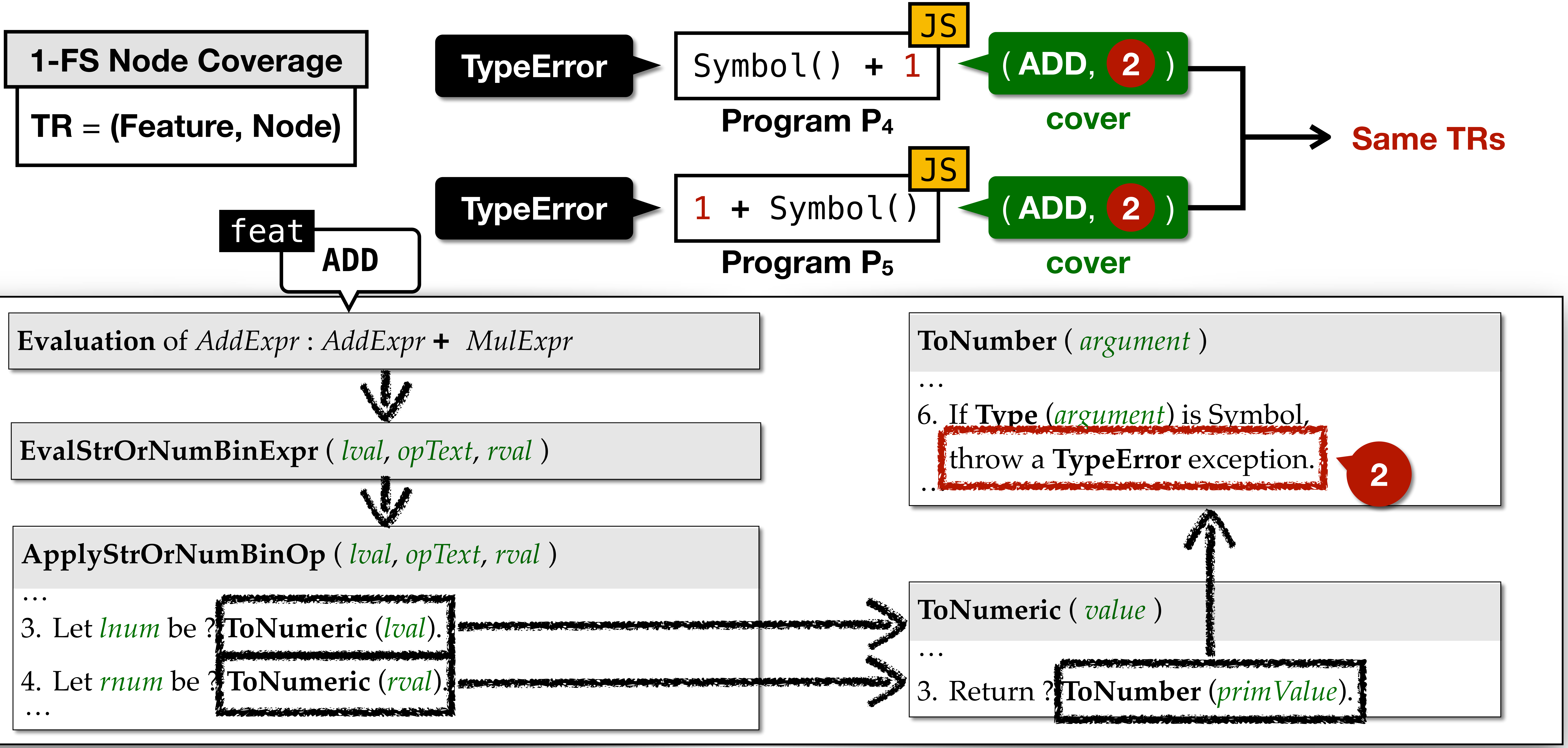
ToNumber (*argument*)

…

6. If **Type** (*argument*) is Symbol, throw a **TypeError** exception.

…

Motivating Example 2



1-FS Node Coverage

TR = (Feature, Node)

Evaluation of *AddExpr* : *AddExpr* **+** *MulExpr*

EvalStrOrNumBinExpr (*lval*, *opText*, *rval*)

ApplyStrOrNumBinOp (*lval*, *opText*, *rval*)

…

3. Let *lnum* be ? **ToNumeric** (*lval*).

4. Let *rnum* be ? **ToNumeric** (*rval*).

…

ToNumeric (*value*)

…

3. Return ? **ToNumber** (*primValue*).

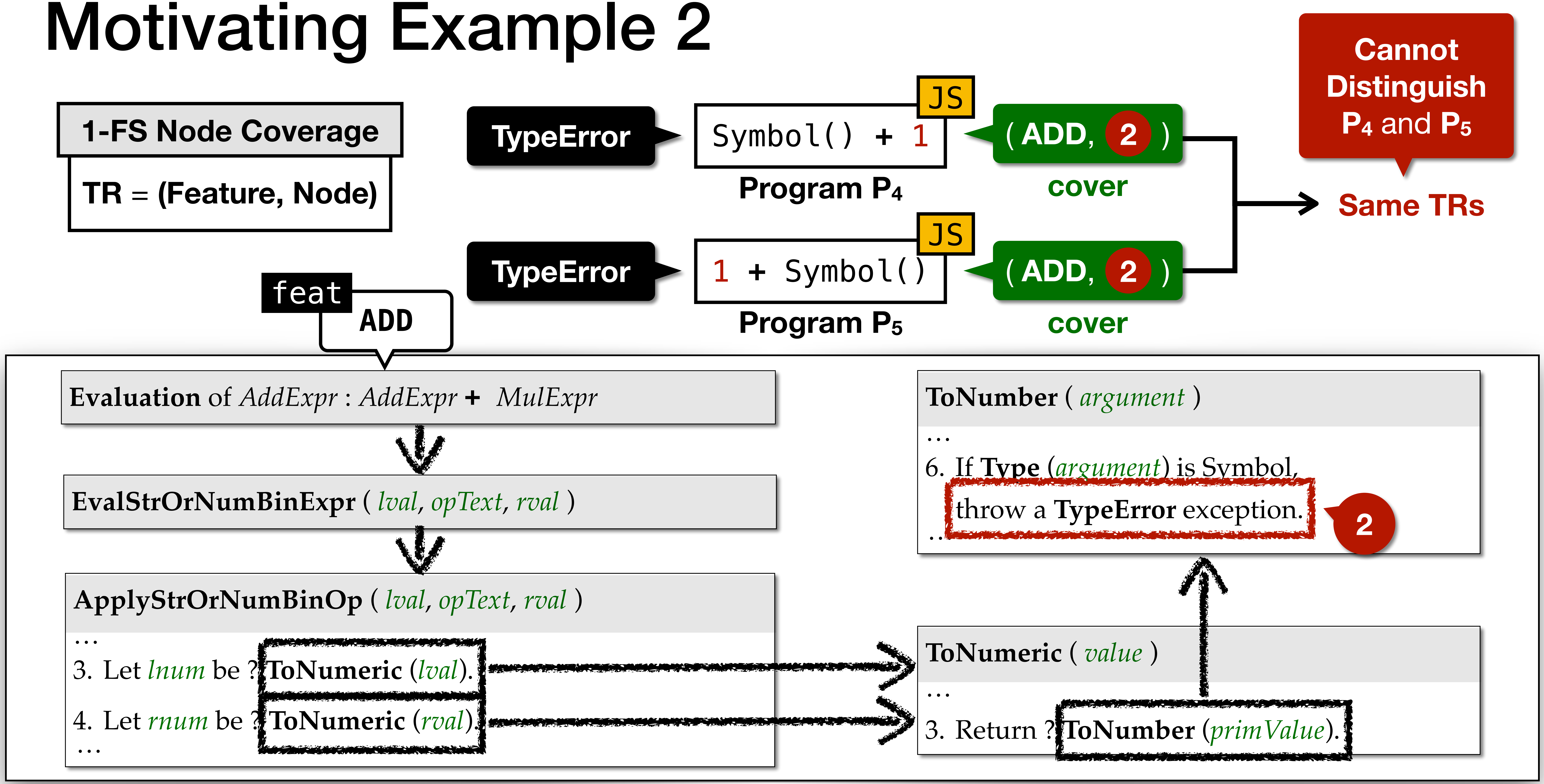
ToNumber (*argument*)

…

6. If **Type** (*argument*) is Symbol, throw a **TypeError** exception.

…

Motivating Example 2



1-FS Node Coverage

TR = (Feature, Node)

TypeError

```
Symbol() + 1
```

Program P_4

(ADD, 2) cover

TypeError

```
1 + Symbol()
```

Program P_5

(ADD, 2) cover

Same TRs

Cannot Distinguish P_4 and P_5

Evaluation of *AddExpr* : *AddExpr* **+** *MulExpr*

EvalStrOrNumBinExpr (*lval*, *opText*, *rval*)

ApplyStrOrNumBinOp (*lval*, *opText*, *rval*)

```
…
3. Let lnum be ? ToNumeric (lval).
4. Let rnum be ? ToNumeric (rval).
…
```

ToNumeric (*value*)

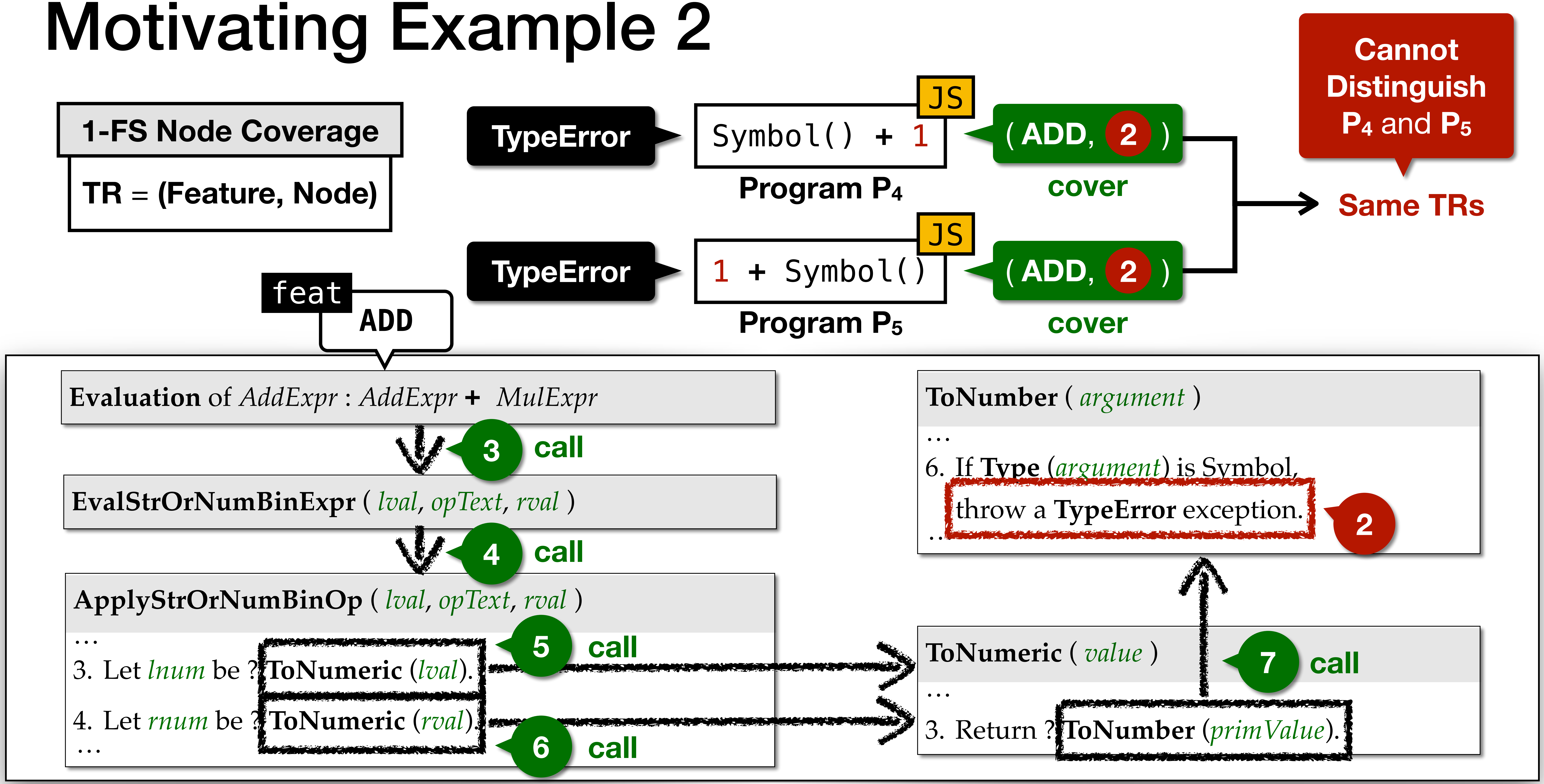
```
…
3. Return ? ToNumber (primValue).
```

ToNumber (*argument*)

```
…
6. If Type (argument) is Symbol,
   throw a TypeError exception.
…
```

Abstract Algorithms in **ECMA-262** (ES13, 2022), **JavaScript Language Specification**

Motivating Example 2



Cannot Distinguish P₄ and P₅

1-FS Node Coverage

TR = (Feature, Node)

TypeError

JS
```
Symbol() + 1
```
Program P_4

(ADD, 2) cover

TypeError

JS
```
1 + Symbol()
```
Program P_5

(ADD, 2) cover

Same TRs

feat ADD

Evaluation of *AddExpr* : *AddExpr* **+** *MulExpr*

3 call

EvalStrOrNumBinExpr (*lval*, *opText*, *rval*)

4 call

ApplyStrOrNumBinOp (*lval*, *opText*, *rval*)

```
…
3. Let lnum be ? ToNumeric (lval).
4. Let rnum be ? ToNumeric (rval).
…
```

5 call

6 call

ToNumeric (*value*)

```
…
3. Return ? ToNumber (primValue).
```

7 call

ToNumber (*argument*)

```
…
6. If Type (argument) is Symbol,
   throw a TypeError exception.
…
```

2

k-Feature-Call-Path-Sensitive (k-FCPS) Coverage

TypeError

JS

```
Symbol() + 1
```

Program P_4

TypeError

JS

```
1 + Symbol()
```

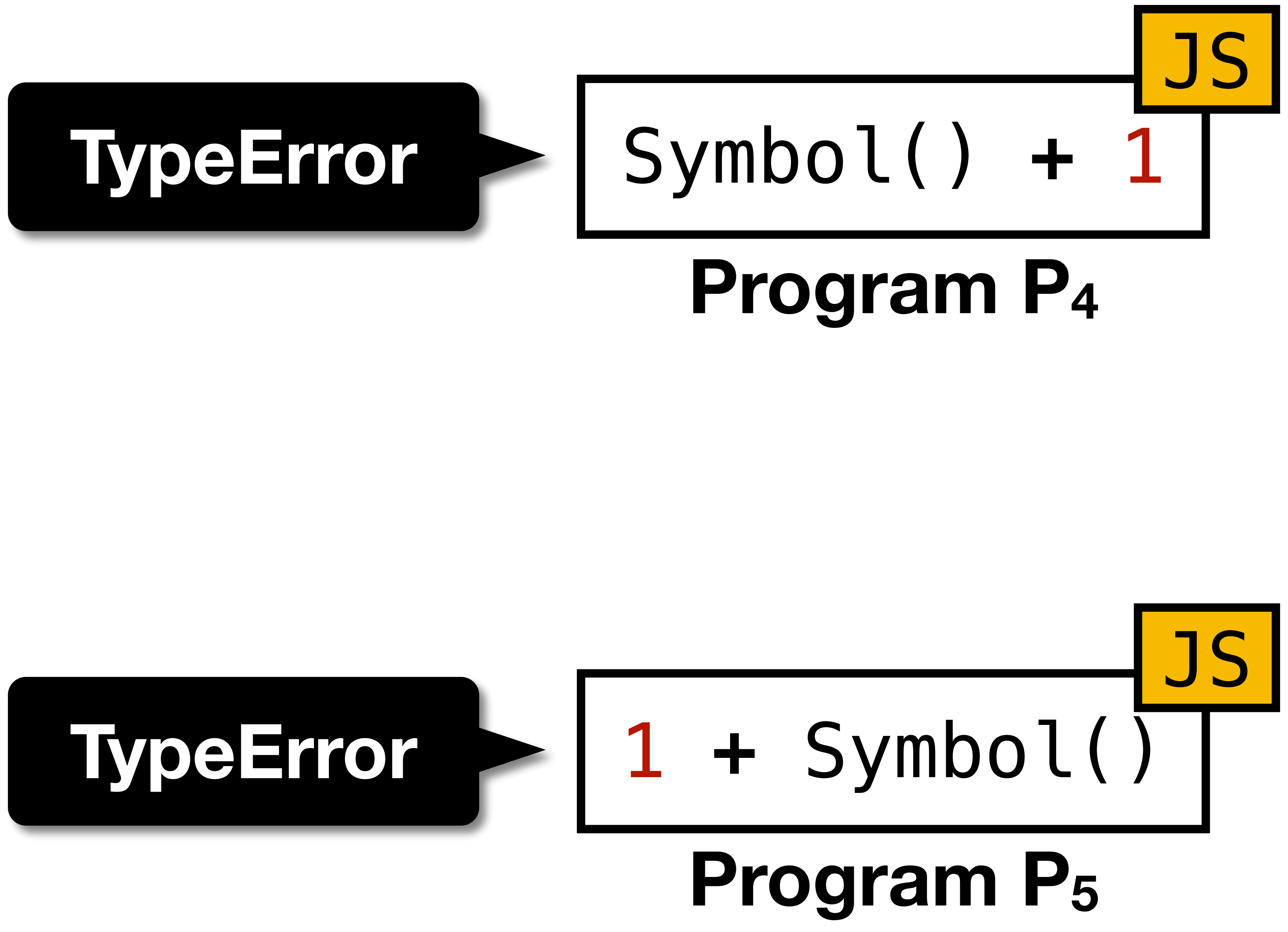
Program P_5

- k-**F**eature-**C**all-**P**ath-**S**ensitive (k-**FCPS**) coverage criterion divides the k-FS TRs with the **call-paths from** the innermost enclosing language feature

k-FCPS Coverage

TR = (Feature$^{\leq k}$, Call-Path, given TR)

k-Feature-Call-Path-Sensitive (k-FCPS) Coverage



```
Symbol() + 1
```

Program P4

```
1 + Symbol()
```

Program P5

1-FCPS Node Coverage

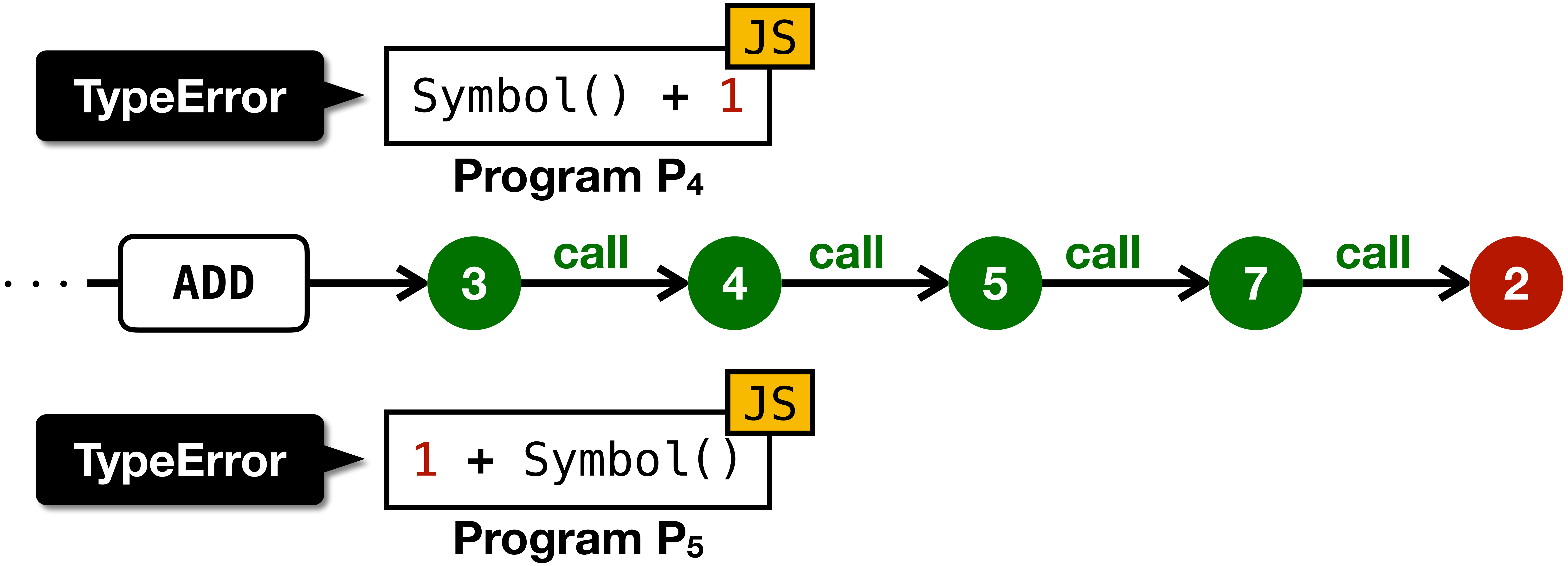
TR = (Feature, Call-Path, Node)

- k-**F**eature-**C**all-**P**ath-**S**ensitive (k-**FCPS**) coverage criterion divides the k-FS TRs with the **call-paths from** the innermost enclosing language feature

k-FCPS Coverage

TR = (Feature$^{\leq k}$, Call-Path, given **TR)**

k-Feature-Call-Path-Sensitive (k-FCPS) Coverage



```
Symbol() + 1
```

```
1 + Symbol()
```

1-FCPS Node Coverage

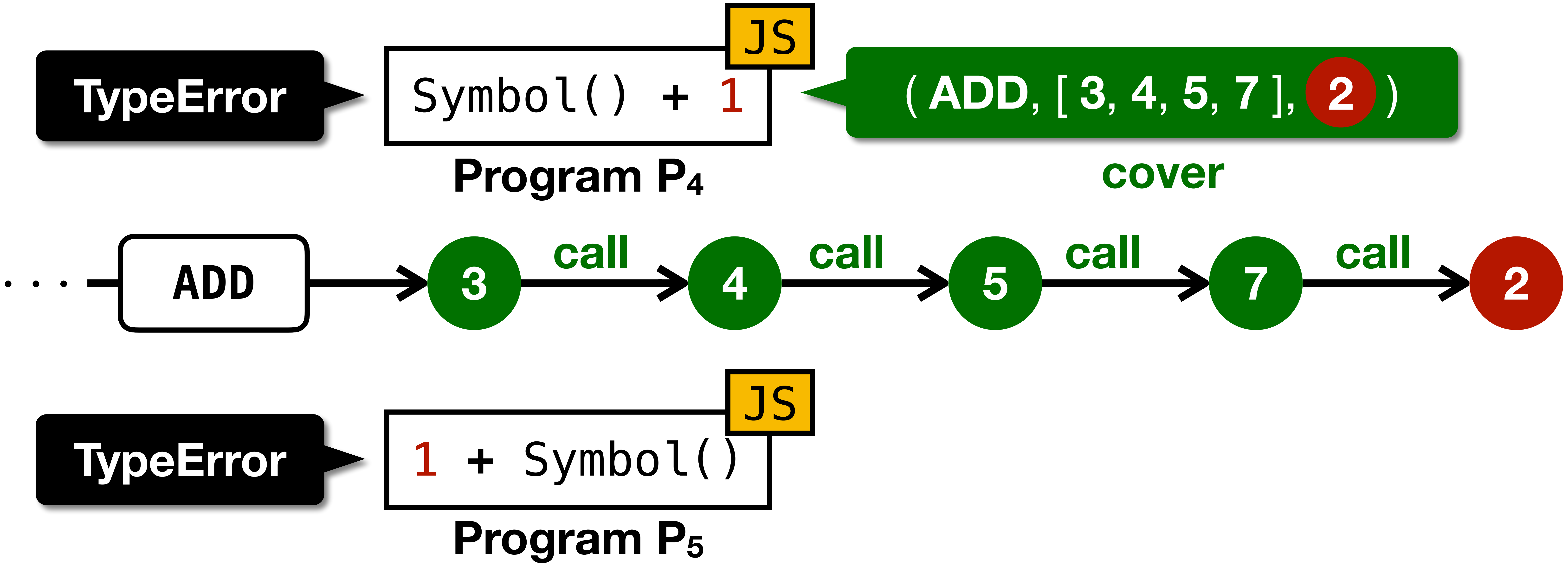
TR = (Feature, Call-Path, Node)

- k-Feature-Call-Path-Sensitive (k-**FCPS**) coverage criterion divides the k-FS TRs with the **call-paths from** the innermost enclosing language feature

k-FCPS Coverage

TR = (Feature$^{\leq k}$, Call-Path, given **TR)**

k-Feature-Call-Path-Sensitive (k-FCPS) Coverage



TypeError

JS

```
Symbol() + 1
```

Program P4

(ADD, [3, 4, 5, 7], 2)

cover

. . . ADD → 3 call→ 4 call→ 5 call→ 7 call→ 2

TypeError

JS

```
1 + Symbol()
```

Program P5

1-FCPS Node Coverage

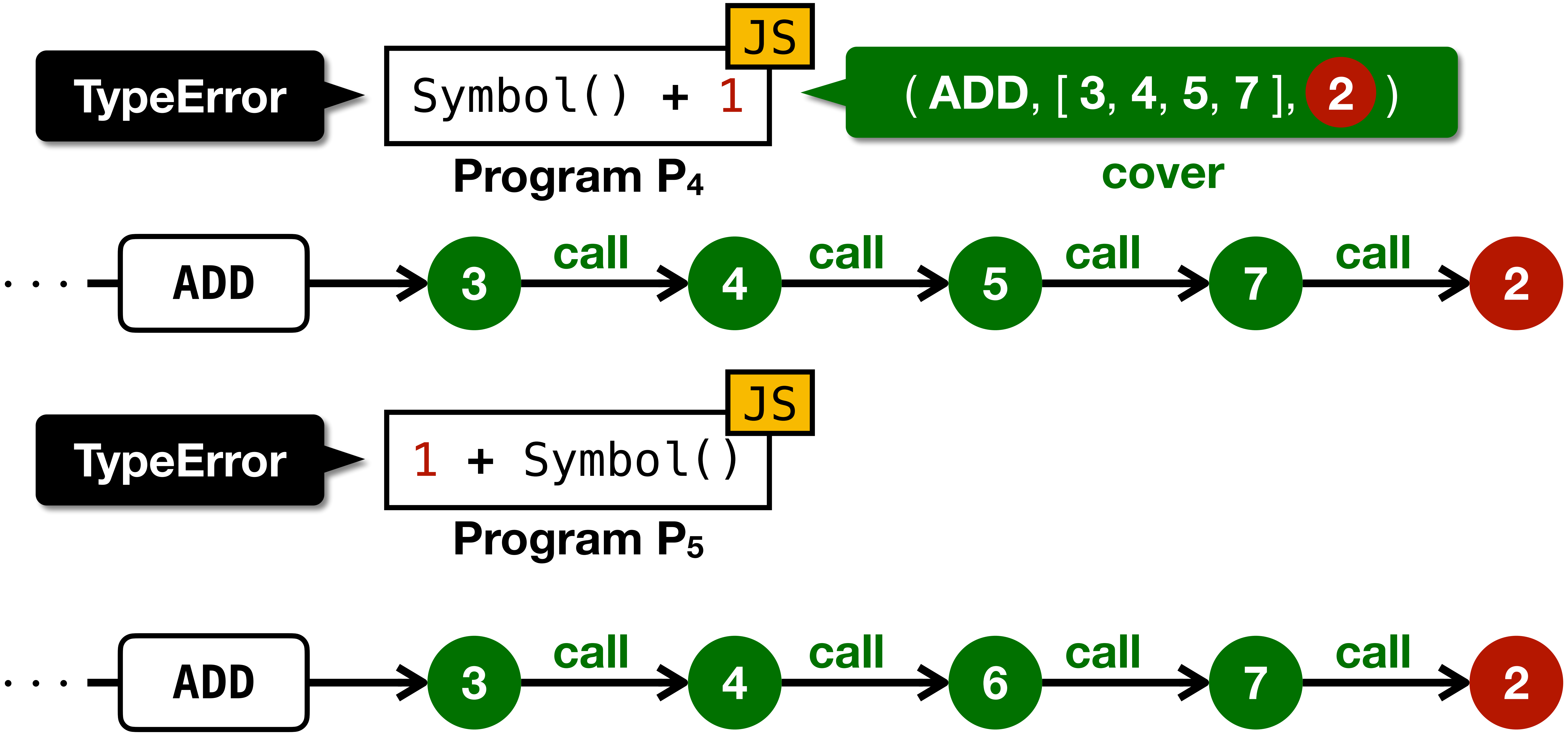
TR = (Feature, Call-Path, Node)

- *k-**F**eature-**C**all-**P**ath-**S**ensitive (k-**FCPS**)* coverage criterion divides the k-FS TRs with the **call-paths from** the innermost enclosing language feature

k-FCPS Coverage

TR = (Feature$^{\leq k}$, Call-Path, given TR)

k-Feature-Call-Path-Sensitive (k-FCPS) Coverage



TypeError

```
Symbol() + 1
```

JS

Program P_4

(ADD, [3, 4, 5, 7], 2)

cover

· · · ADD → 3 call → 4 call → 5 call → 7 call → 2

TypeError

```
1 + Symbol()
```

JS

Program P_5

· · · ADD → 3 call → 4 call → 6 call → 7 call → 2

1-FCPS Node Coverage

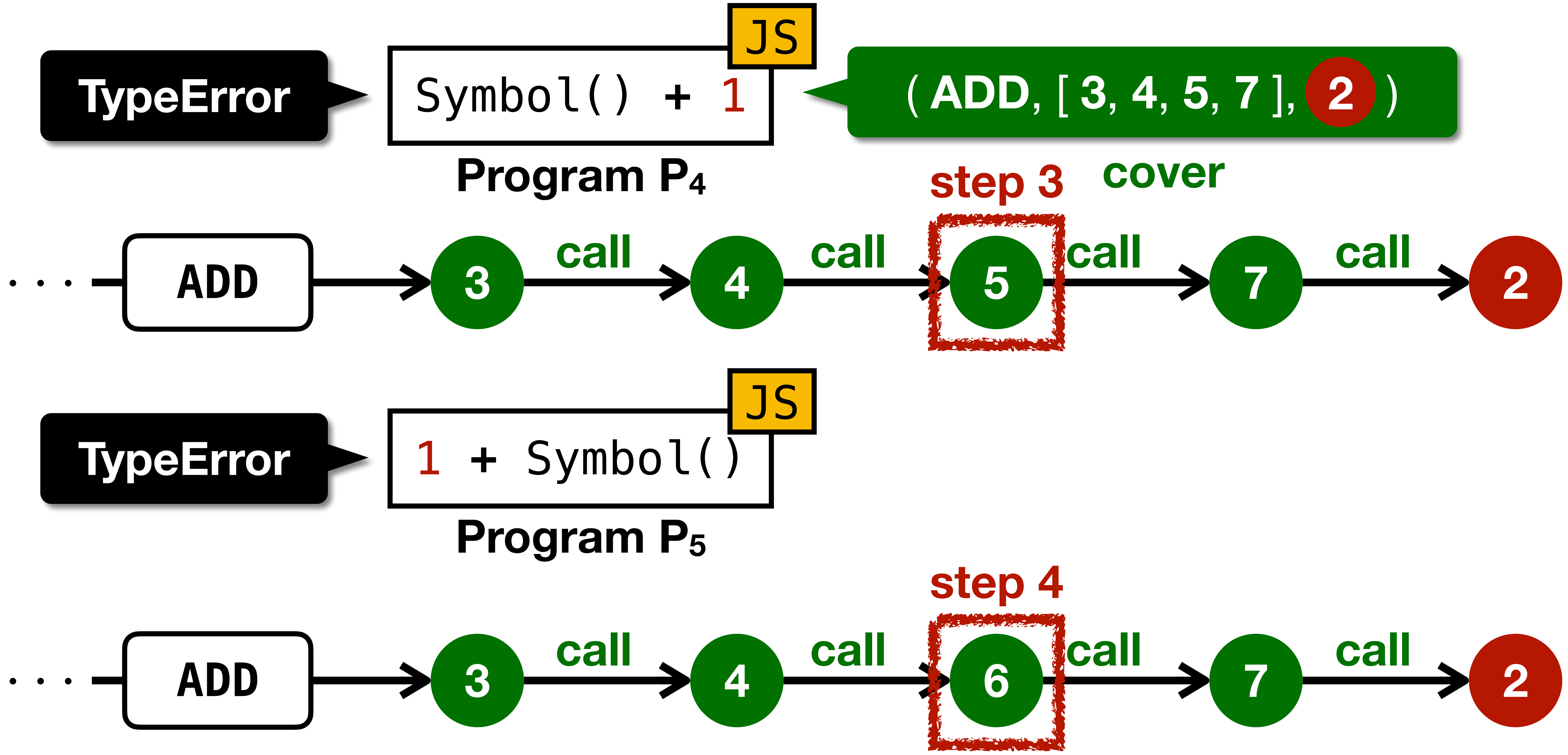
TR = (Feature, Call-Path, Node)

- k-**F**eature-**C**all-**P**ath-**S**ensitive (k-**FCPS**) coverage criterion divides the k-FS TRs with the **call-paths from** the innermost enclosing language feature

k-FCPS Coverage

TR = (Feature$^{\leq k}$, Call-Path, given TR)

k-Feature-Call-Path-Sensitive (k-FCPS) Coverage



TypeError

```
Symbol() + 1
```

Program P_4

(ADD, [3, 4, 5, 7], 2)

step 3 cover

TypeError

```
1 + Symbol()
```

Program P_5

step 4

1-FCPS Node Coverage

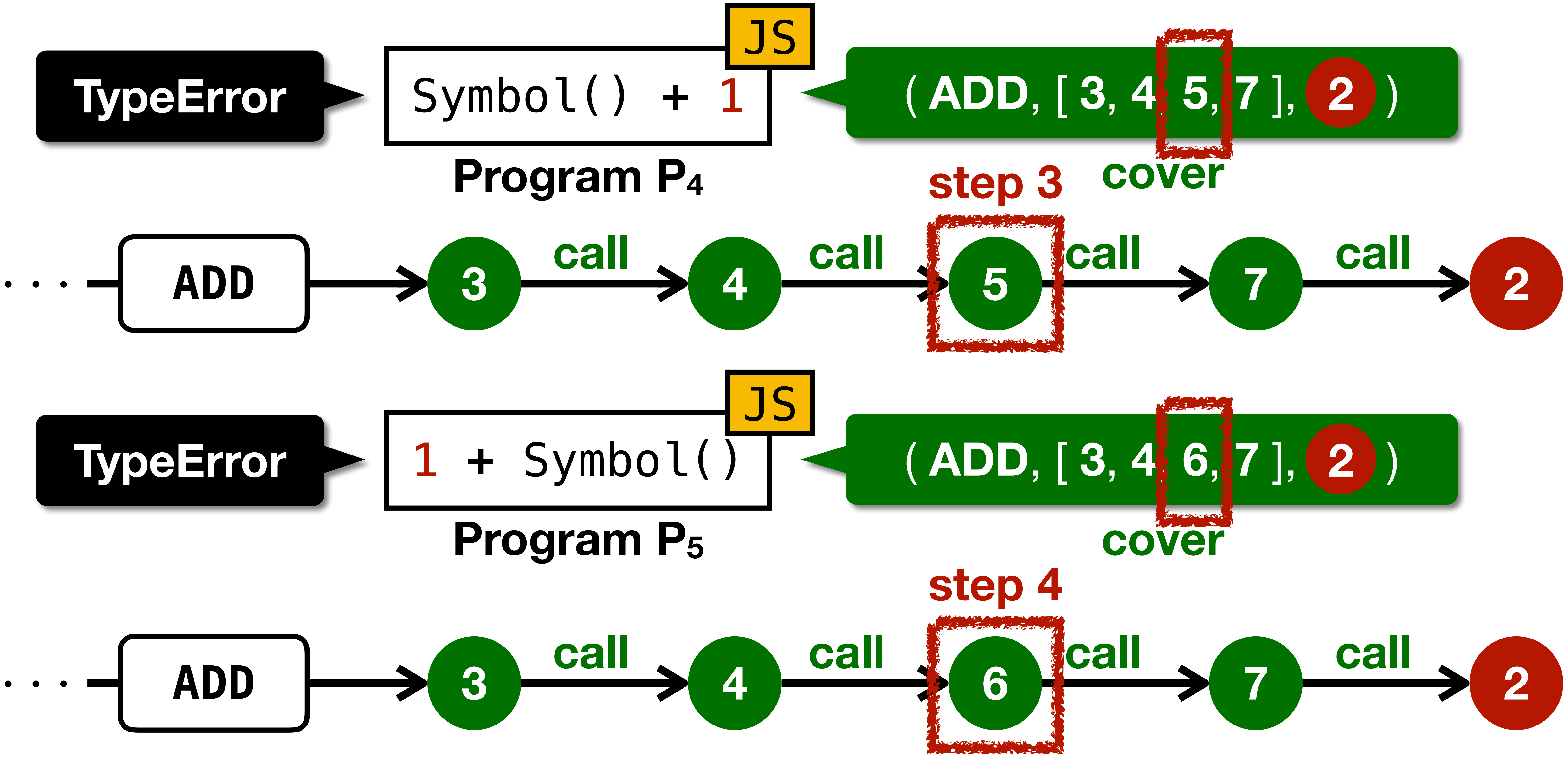
TR = (Feature, Call-Path, Node)

- *k*-**F**eature-**C**all-**P**ath-**S**ensitive (k-**FCPS**) coverage criterion divides the k-FS TRs with the **call-paths from** the innermost enclosing language feature

k-FCPS Coverage

TR = (Feature$^{\leq k}$, Call-Path, given TR)

k-Feature-Call-Path-Sensitive (k-FCPS) Coverage



```
Symbol() + 1
```

```
1 + Symbol()
```

1-FCPS Node Coverage

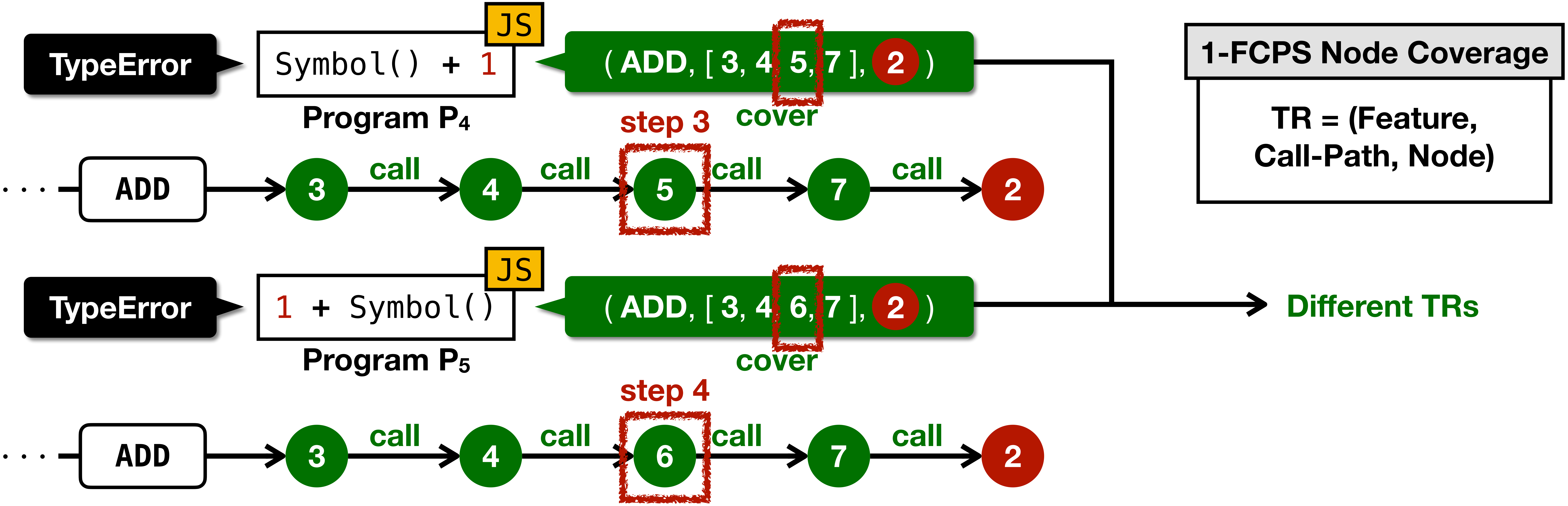
TR = (Feature, Call-Path, Node)

- k-**F**eature-**C**all-**P**ath-**S**ensitive (k-**FCPS**) coverage criterion divides the k-FS TRs with the **call-paths from** the innermost enclosing language feature

k-FCPS Coverage

TR = (Feature$^{\leq k}$, Call-Path, given **TR)**

k-Feature-Call-Path-Sensitive (*k*-FCPS) Coverage



1-FCPS Node Coverage

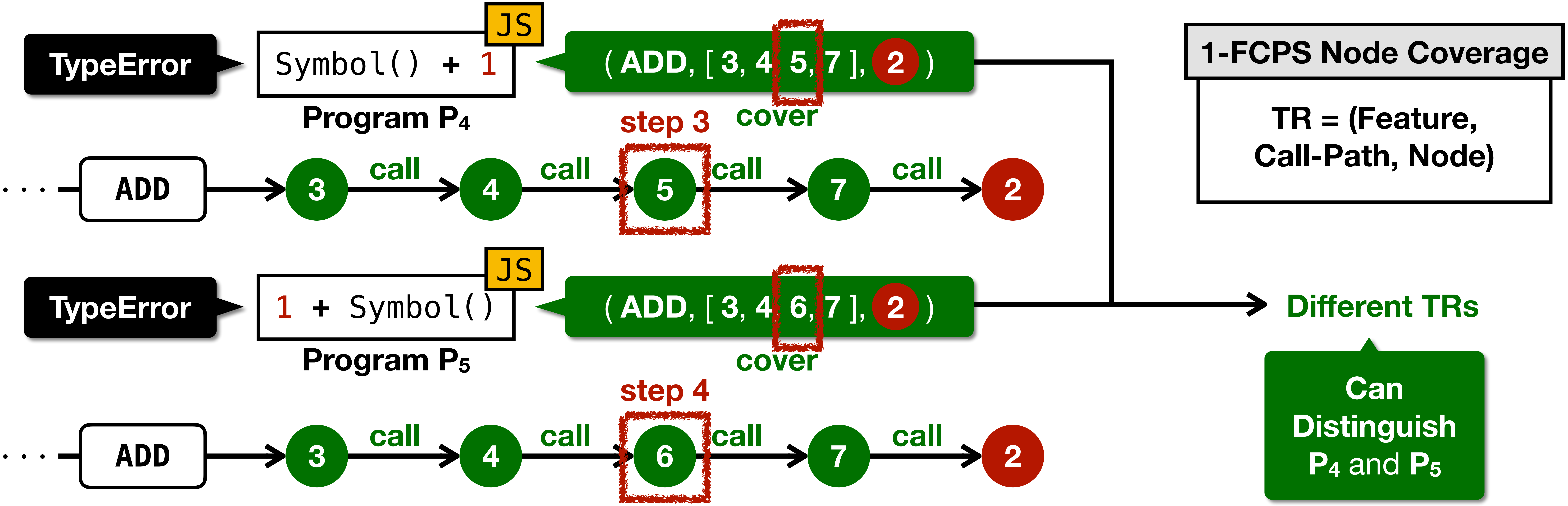
TR = (Feature, Call-Path, Node)

- *k*-**F**eature-**C**all-**P**ath-**S**ensitive (*k*-**FCPS**) coverage criterion divides the *k*-FS TRs with the **call-paths from** the innermost enclosing language feature

***k*-FCPS Coverage**

TR = (Feature$^{\leq k}$, Call-Path, given **TR)**

k-Feature-Call-Path-Sensitive (k-FCPS) Coverage



- *k*-**F**eature-**C**all-**P**ath-**S**ensitive (k-**FCPS**) coverage criterion divides the k-FS TRs with the **call-paths from** the innermost enclosing language feature

k-FCPS Coverage

TR = (Feature$^{\leq k}$, Call-Path, given **TR)**

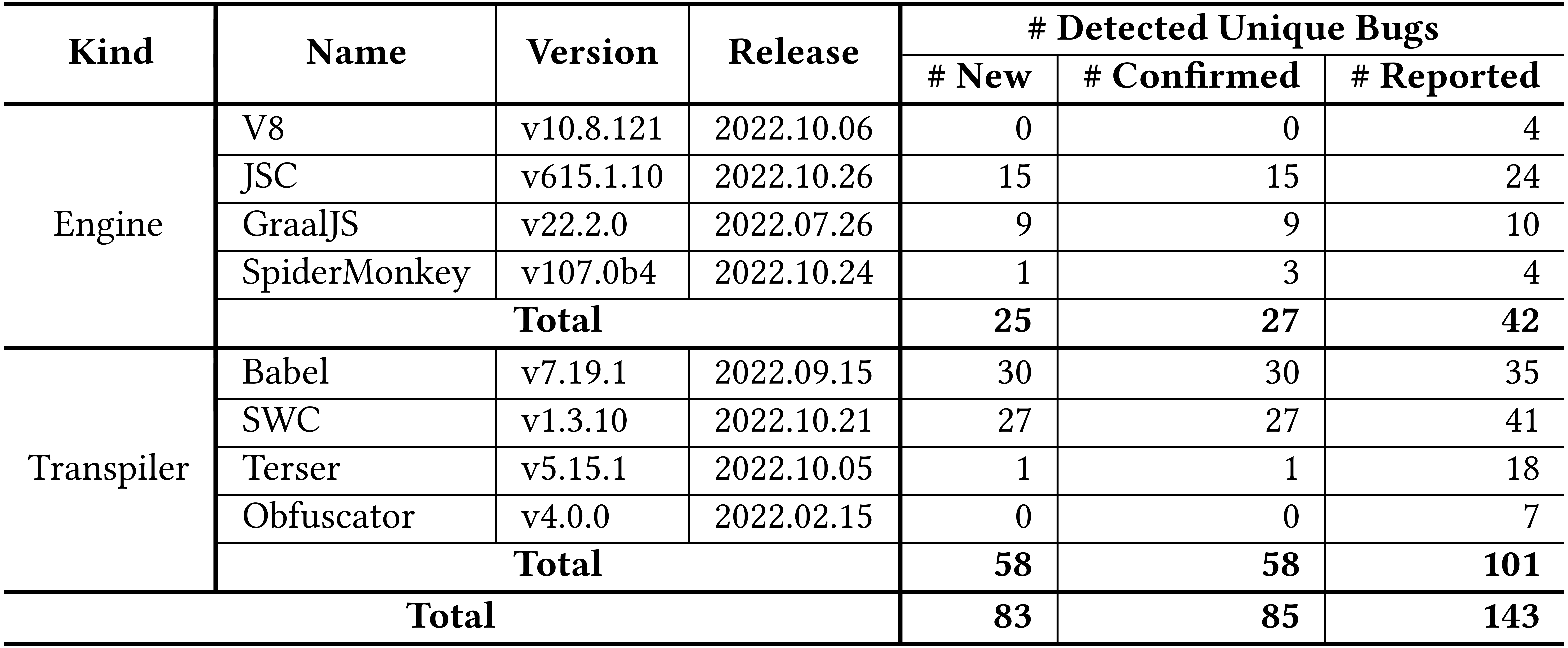
Evaluation

5 different k-FS and k-FCPS coverage criteria

- **Conformance Test Synthesis** in 50 hours with **0-FS** / **1-FS** / **2-FS** / **1-FCPS** / **2-FCPS**
- **JavaScript Specification** — ECMA-262 for **ES13 (2022)**
- **JavaScript Implementations** — **4 Engines** and **4 Transpilers**

Kind	Name	Version	Release
Engine	**V8**	v10.8.121	2022.10.06
	JSC	v615.1.10	2022.10.26
	GraalJS	v22.2.0	2022.07.26
	SpiderMonkey	v107.0b4	2022.10.24
Transpiler	**Babel**	v7.19.1	2022.09.15
	SWC	v1.3.10	2022.10.21
	Terser	v5.15.1	2022.10.05
	Obfuscator	v4.0.0	2022.02.15

RQ1) Conformance Bug Detection



Kind	Name	Version	Release	# Detected Unique Bugs		
				# New	# Confirmed	# Reported
Engine	V8	v10.8.121	2022.10.06	0	0	4
	JSC	v615.1.10	2022.10.26	15	15	24
	GraalJS	v22.2.0	2022.07.26	9	9	10
	SpiderMonkey	v107.0b4	2022.10.24	1	3	4
	Total			**25**	**27**	**42**
Transpiler	Babel	v7.19.1	2022.09.15	30	30	35
	SWC	v1.3.10	2022.10.21	27	27	41
	Terser	v5.15.1	2022.10.05	1	1	18
	Obfuscator	v4.0.0	2022.02.15	0	0	7
	Total			**58**	**58**	**101**
Total				**83**	**85**	**143**

RQ1) Conformance Bug Detection

Kind	Name	Version	Release	# Detected Unique Bugs		
				# New	# Confirmed	# Reported
Engine	V8	v10.8.121	2022.10.06	0	0	4
	JSC	v615.1.10	2022.10.26	15	15	24
	GraalJS	v22.2.0	2022.07.26	9	9	10
	SpiderMonkey	v107.0b4	2022.10.24	1	3	4
	Total			**25**	**27**	**42**
Transpiler	Babel	v7.19.1	2022.09.15	30	30	35
	SWC	v1.3.10	2022.10.21	27	27	41
	Terser	v5.15.1	2022.10.05	1	1	18
	Obfuscator	v4.0.0	2022.02.15	0	0	7
	Total			**58**	**58**	**101**
Total				**83**	**85**	**143**

PLRG

RQ1) Conformance Bug Detection

Kind	Name	Version	Release	# Detected Unique Bugs		
				# New	# Confirmed	# Reported
Engine	V8	v10.8.121	2022.10.06	0	0	4
	JSC	v615.1.10	2022.10.26	15	15	24
	GraalJS	v22.2.0	2022.07.26	9	9	10
	SpiderMonkey	v107.0b4	2022.10.24	1	3	4
	Total			**25**	**27**	**42**
Transpiler	Babel	v7.19.1	2022.09.15	30	30	35
	SWC	v1.3.10	2022.10.21	27	27	41
	Terser	v5.15.1	2022.10.05	1	1	18
	Obfuscator	v4.0.0	2022.02.15	0	0	7
	Total			**58**	**58**	**101**
Total				**83**	**85**	**143**

RQ1) Conformance Bug Detection

Kind	Name	Version	Release	# Detected Unique Bugs		
				# New	# Confirmed	# Reported
Engine	V8	v10.8.121	2022.10.06	0	0	4
	JSC	v615.1.10	2022.10.26	15	15	24
	GraalJS	v22.2.0	2022.07.26	9	9	10
	SpiderMonkey	v107.0b4	2022.10.24	1	3	4
	Total			**25**	**27**	**42**
Transpiler	Babel	v7.19.1	2022.09.15	30	30	35
	SWC	v1.3.10	2022.10.21	27	27	41
	Terser	v5.15.1	2022.10.05	1	1	18
	Obfuscator	v4.0.0	2022.02.15	0	0	7
	Total			**58**	**58**	**101**
Total				**83**	**85**	**143**

RQ2) Effectiveness of k-FS Coverage Criteria

Coverage Criteria $C_{\mathbb{G}}$	# Covered k-F(CP)S-TR (k)			# Syn. Test	# Bug
	# Node	# Branch	# Total		
0-FS node-or-branch (0-fs)	10.0	5.6	15.6	2,111	55
1-FS node-or-branch (1-fs)	79.3	45.7	125.0	6,766	83
1-FCPS node-or-branch (1-fcps)	179.7	97.6	277.3	9,092	87
2-FS node-or-branch (2-fs)	1,199.8	696.3	1,896.1	97,423	102
2-FCPS node-or-branch (2-fcps)	2,323.1	1,297.6	3,620.7	122,589	111

RQ2) Effectiveness of k-FS Coverage Criteria

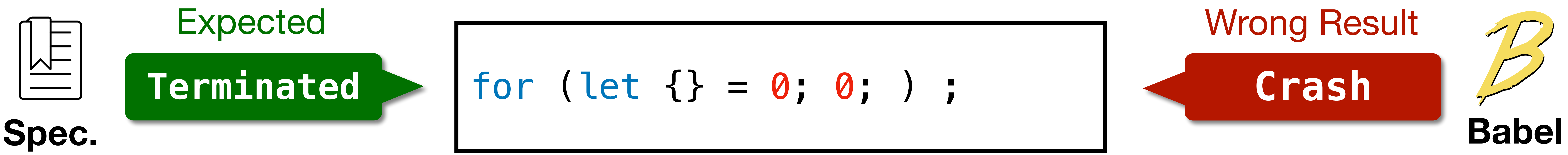
Coverage Criteria $C_{\mathbb{G}}$	# Covered k-F(CP)S-TR (k)			# Syn. Test	# Bug
	# Node	# Branch	# Total		
0-FS node-or-branch (0-fs)	10.0	5.6	15.6	2,111	55
1-FS node-or-branch (1-fs)	79.3	45.7	125.0	6,766	83
1-FCPS node-or-branch (1-fcps)	179.7	97.6	277.3	9,092	87
2-FS node-or-branch (2-fs)	1,199.8	696.3	1,896.1	97,423	102
2-FCPS node-or-branch (2-fcps)	2,323.1	1,297.6	3,620.7	122,589	111

+28

RQ2) Effectiveness of k-FS Coverage Criteria

Coverage Criteria $C_{\mathbb{G}}$	# Covered k-F(CP)S-TR (k)			# Syn. Test	# Bug
	# Node	# Branch	# Total		
0-FS node-or-branch (0-fs)	10.0	5.6	15.6	2,111	55
1-FS node-or-branch (1-fs)	79.3	45.7	125.0	6,766	83
1-FCPS node-or-branch (1-fcps)	179.7	97.6	277.3	9,092	87
2-FS node-or-branch (2-fs)	1,199.8	696.3	1,896.1	97,423	102
2-FCPS node-or-branch (2-fcps)	2,323.1	1,297.6	3,620.7	122,589	111

+28



```
for (let {} = 0; 0; ) ;
```

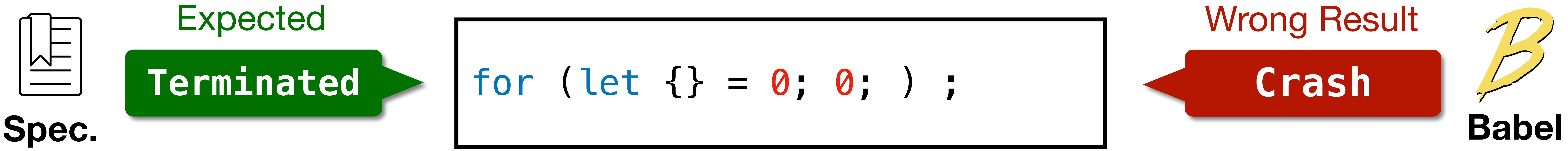
Synthesized with **1-FS** but not with **0-FS**

RQ2) Effectiveness of k-FS Coverage Criteria

Coverage Criteria $C_{\mathbb{G}}$	# Covered k-F(CP)S-TR (k)			# Syn. Test	# Bug
	# Node	# Branch	# Total		
0-FS node-or-branch (0-fs)	10.0	5.6	15.6	2,111	55
1-FS node-or-branch (1-fs)	79.3	45.7	125.0	6,766	83
1-FCPS node-or-branch (1-fcps)	179.7	97.6	277.3	9,092	87
2-FS node-or-branch (2-fs)	1,199.8	696.3	1,896.1	97,423	102
2-FCPS node-or-branch (2-fcps)	2,323.1	1,297.6	3,620.7	122,589	111

+28

+19



```
for (let {} = 0; 0; ) ;
```

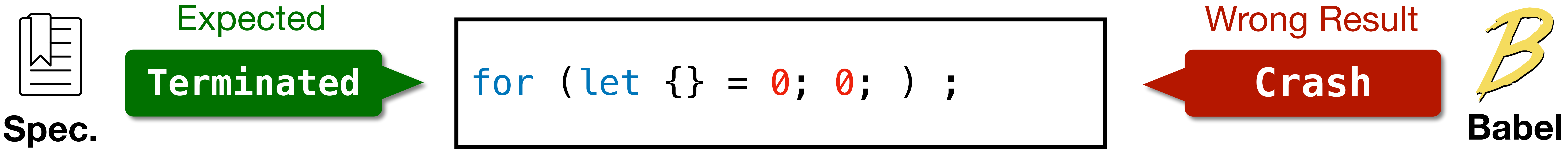
Synthesized with **1-FS** but not with **0-FS**

RQ2) Effectiveness of k-FS Coverage Criteria

Coverage Criteria $C_{\mathbb{G}}$	# Covered k-F(CP)S-TR (k)			# Syn. Test	# Bug
	# Node	# Branch	# Total		
0-FS node-or-branch (0-fs)	10.0	5.6	15.6	2,111	55
1-FS node-or-branch (1-fs)	79.3	45.7	125.0	6,766	83
1-FCPS node-or-branch (1-fcps)	179.7	97.6	277.3	9,092	87
2-FS node-or-branch (2-fs)	1,199.8	696.3	1,896.1	97,423	102
2-FCPS node-or-branch (2-fcps)	2,323.1	1,297.6	3,620.7	122,589	111

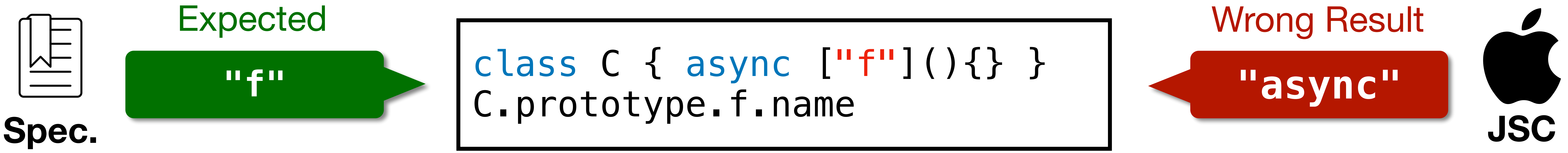
+28

+19



```
for (let {} = 0; 0; ) ;
```

Synthesized with **1-FS** but not with **0-FS**



```
class C { async ["f"](){} }
C.prototype.f.name
```

Synthesized with **2-FS** but not with **1-FS**

RQ3) Effectiveness of k-FCPS Coverage Criteria

Coverage Criteria $C_{\mathbb{G}}$	# Covered k-F(CP)S-TR (k)			# Syn. Test	# Bug
	# Node	# Branch	# Total		
0-FS node-or-branch (0-fs)	10.0	5.6	15.6	2,111	55
1-FS node-or-branch (1-fs)	79.3	45.7	125.0	6,766	83
1-FCPS node-or-branch (1-fcps)	179.7	97.6	277.3	9,092	87
2-FS node-or-branch (2-fs)	1,199.8	696.3	1,896.1	97,423	102
2-FCPS node-or-branch (2-fcps)	2,323.1	1,297.6	3,620.7	122,589	111

RQ3) Effectiveness of k-FCPS Coverage Criteria

Coverage Criteria $C_{\mathbb{G}}$	# Covered k-F(CP)S-TR (k)			# Syn. Test	# Bug
	# Node	# Branch	# Total		
0-FS node-or-branch (0-fs)	10.0	5.6	15.6	2,111	55
1-FS node-or-branch (1-fs)	79.3	45.7	125.0	6,766	83
1-FCPS node-or-branch (1-fcps)	179.7	97.6	277.3	9,092	87
2-FS node-or-branch (2-fs)	1,199.8	696.3	1,896.1	97,423	102
2-FCPS node-or-branch (2-fcps)	2,323.1	1,297.6	3,620.7	122,589	111

+4

RQ3) Effectiveness of k-FCPS Coverage Criteria

Coverage Criteria $C_{\mathbb{G}}$	# Covered k-F(CP)S-TR (k)			# Syn. Test	# Bug
	# Node	# Branch	# Total		
0-FS node-or-branch (0-fs)	10.0	5.6	15.6	2,111	55
1-FS node-or-branch (1-fs)	79.3	45.7	125.0	6,766	83
1-FCPS node-or-branch (1-fcps)	179.7	97.6	277.3	9,092	87
2-FS node-or-branch (2-fs)	1,199.8	696.3	1,896.1	97,423	102
2-FCPS node-or-branch (2-fcps)	2,323.1	1,297.6	3,620.7	122,589	111

+4

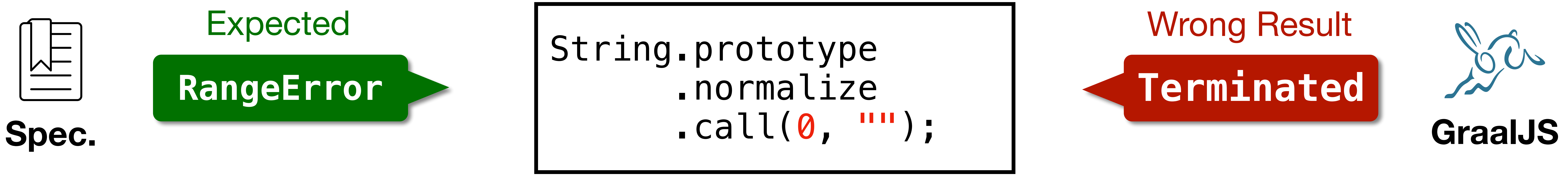
+9

RQ3) Effectiveness of k-FCPS Coverage Criteria

Coverage Criteria $C_{\mathbb{G}}$	# Covered k-F(CP)S-TR (k)			# Syn. Test	# Bug
	# Node	# Branch	# Total		
0-FS node-or-branch (0-fs)	10.0	5.6	15.6	2,111	55
1-FS node-or-branch (1-fs)	79.3	45.7	125.0	6,766	83
1-FCPS node-or-branch (1-fcps)	179.7	97.6	277.3	9,092	87
2-FS node-or-branch (2-fs)	1,199.8	696.3	1,896.1	97,423	102
2-FCPS node-or-branch (2-fcps)	2,323.1	1,297.6	3,620.7	122,589	111

+4

+9



```
String.prototype
        .normalize
        .call(0, "");
```

Synthesized with **1-FCPS** or **2-FCPS** but not with **1-FS** or **2-FS**

PLDI 2023

Feature-Sens. Coverage

ASE 2020

ICSE 2021

ECMA-262 (JS Spec.)

Mech. Spec. Extraction

Mechanized Spec.

Conform. Test Synthesis

Conformance Tests

JavaScript Engines

PLDI 2023

Ongoing

Feature-Sens. Coverage

Selective FS-Coverage

ASE 2020

ICSE 2021

ECMA-262 (JS Spec.)

Mech. Spec. Extraction

Mechanized Spec.

Conform. Test Synthesis

Conformance Tests

JavaScript Engines

ASE 2021
PLDI 2023
Ongoing
Type-related Spec. Bug
Spec. Type Analysis
Feature-Sens. Coverage
Selective FS-Coverage
ASE 2020
ICSE 2021
ECMA-262 (JS Spec.)
Mech. Spec. Extraction
Mechanized Spec.
Conform. Test Synthesis
Conformance Tests
JavaScript Engines

ASE 2021
Ongoing
PLDI 2023
Ongoing
Type-related Spec. Bug
Spec. Type Analysis
Type Guard Inference
Feature-Sens. Coverage
Selective FS-Coverage
ASE 2020
ICSE 2021
ECMA-262 (JS Spec.)
Mech. Spec. Extraction
Mechanized Spec.
Conform. Test Synthesis
Conformance Tests
JavaScript Engines

Future Work
Spec. Repair Tool
Type-related Spec. Bug
ASE 2021
Spec. Type Analysis
Ongoing
Type Guard Inference
PLDI 2023
Feature-Sens. Coverage
Ongoing
Selective FS-Coverage
ECMA-262 (JS Spec.)
ASE 2020
Mech. Spec. Extraction
Mechanized Spec.
ICSE 2021
Conform. Test Synthesis
Conformance Tests
JavaScript Engines

Future Work
Spec. Repair Tool
Type-related Spec. Bug
ASE 2021
Spec. Type Analysis
Ongoing
Type Guard Inference
PLDI 2023
Feature-Sens. Coverage
Ongoing
Selective FS-Coverage
ASE 2020
ICSE 2021
ECMA-262 (JS Spec.)
Mech. Spec. Extraction
Mechanized Spec.
Conform. Test Synthesis
Conformance Tests
JavaScript Engines
FSE 2022
Static Analyzer
Static Analyzer Derivation

Future Work

Spec. Repair Tool

Type-related Spec. Bug

ASE 2021

Spec. Type Analysis

Ongoing

Type Guard Inference

PLDI 2023

Feature-Sens. Coverage

Ongoing

Selective FS-Coverage

ECMA-262 (JS Spec.)

ASE 2020

Mech. Spec. Extraction

Mechanized Spec.

ICSE 2021

Conform. Test Synthesis

Conformance Tests

JavaScript Engines

Static Analyzer

FSE 2022

Static Analyzer Derivation

Future Work

Dataflow Facts Derivation

Dataflow Facts

Future Work

Spec. Repair Tool

Type-related Spec. Bug

ASE 2021

Spec. Type Analysis

Ongoing

Type Guard Inference

PLDI 2023

Feature-Sens. Coverage

Ongoing

Selective FS-Coverage

ECMA-262 (JS Spec.)

ASE 2020

Mech. Spec. Extraction

Mechanized Spec.

ICSE 2021

Conform. Test Synthesis

Conformance Tests

JavaScript Engines

Static Analyzer

FSE 2022

Static Analyzer Derivation

Future Work

Dataflow Facts Derivation

Dataflow Facts

Ongoing

Spec. Synthesis

PLRG