Language Design and Implementation using JavaScript Mechanized Specification

Jihyeok Park and Sukyoung Ryu

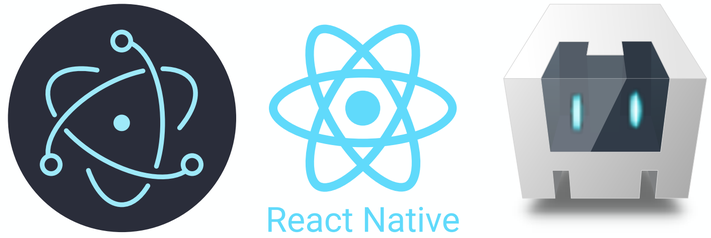
2024.11.20 @ UNIST

PLRG @ Korea Univ.

- **Programming Language Research Group (PLRG)**
- **Members:** 2 Master Students / 6 Undergraduate Students
- **Research Areas:** Programming Languages (PL) and Software Engineering (SE)
  - Program Analysis
  - Mechanized Language Specification
  - Automated Testing
  - Program Synthesis
- **Publications:** PL and SE
  - **PL**: PLDI (2023 / 2024)
  - **SE**: ICSE (2021) / FSE (2021, 2022), ASE (2020, 2021)

JavaScript is Everywhere

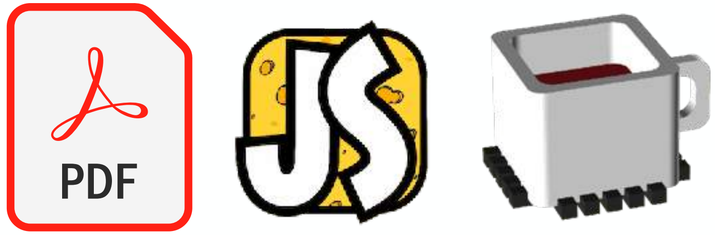
Client-Side Programming



Mobile/Desktop Applications

JS

Sever-Side Programming

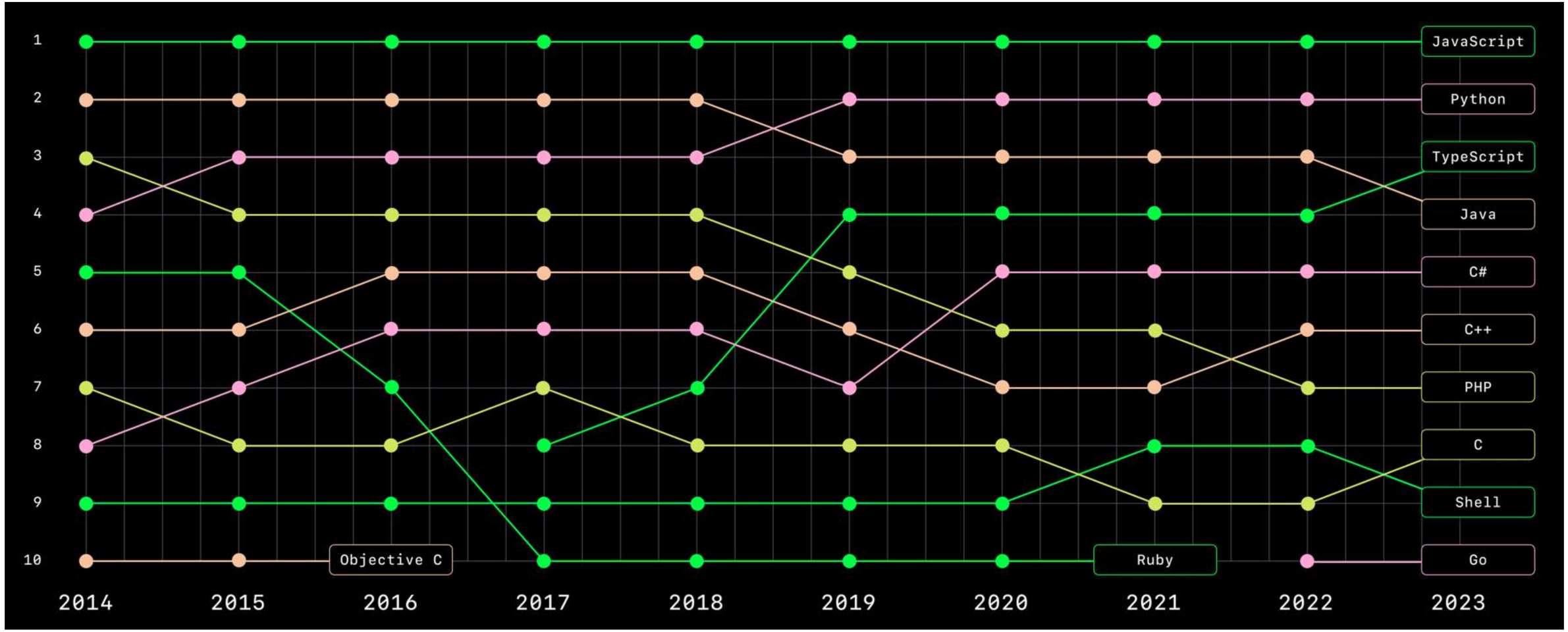


Others (PDF, IoT, Microcontrollers, etc.)

PLRG

JavaScript is Everywhere

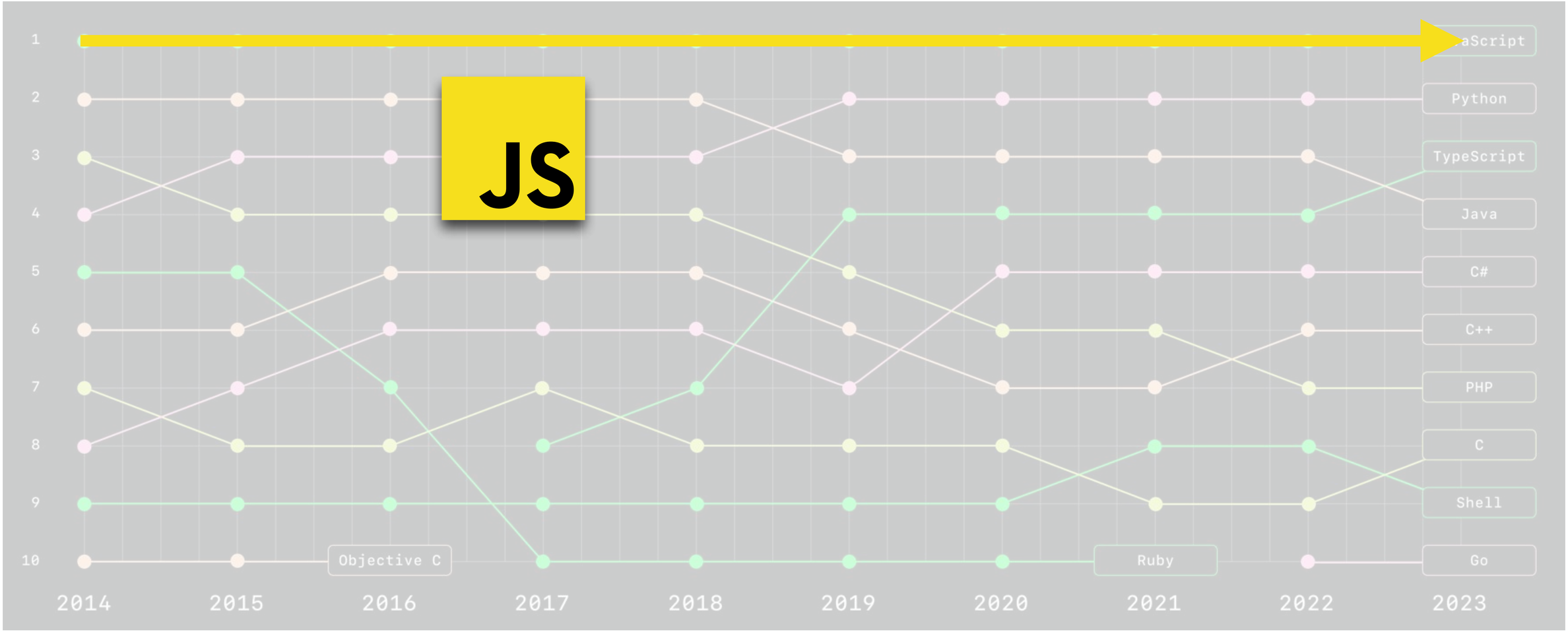
GitHub



https://octoverse.github.com/

JavaScript is Everywhere

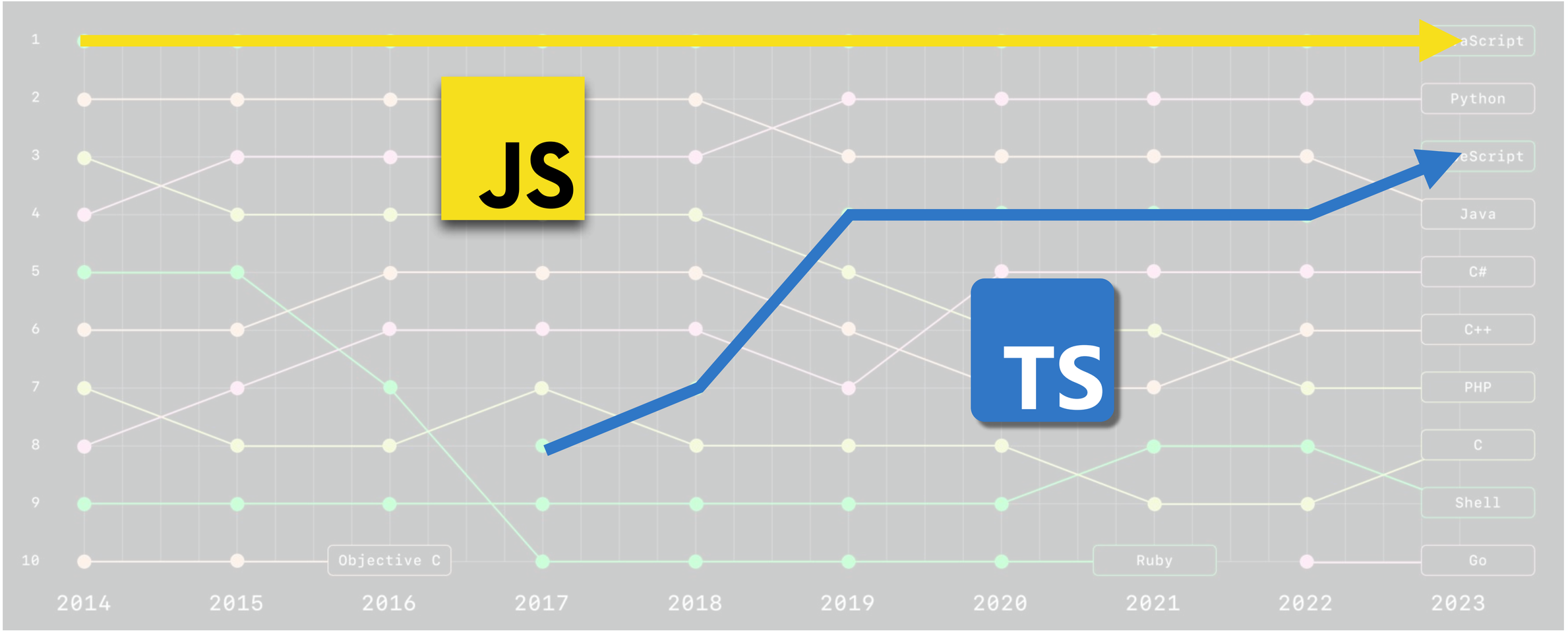
GitHub



https://octoverse.github.com/

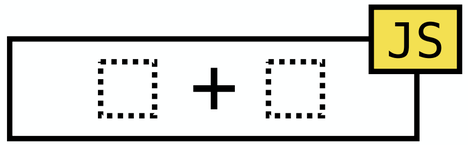
JavaScript is Everywhere

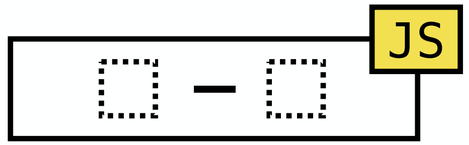
GitHub



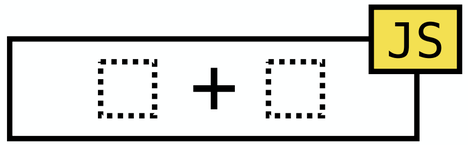
https://octoverse.github.com/

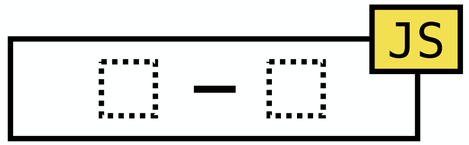
But, **JavaScript** is Complicated

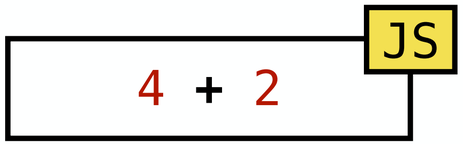




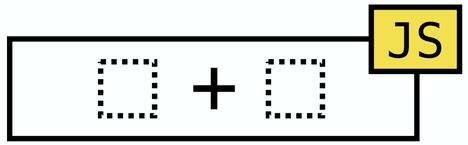
But, **JavaScript** is Complicated

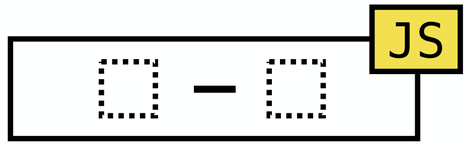


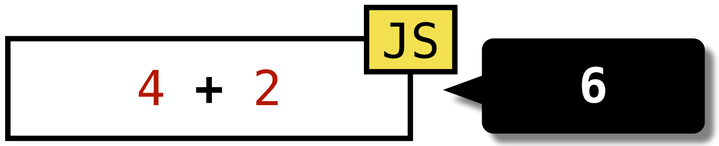




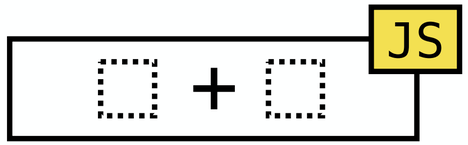
But, **JavaScript** is Complicated

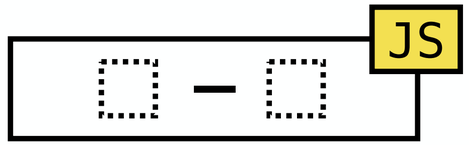


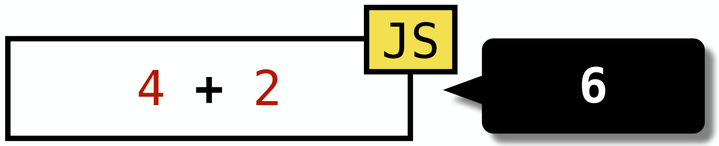


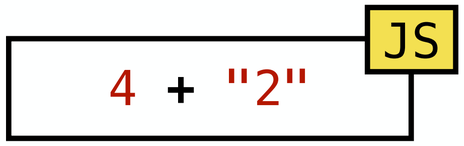


But, **JavaScript** is Complicated

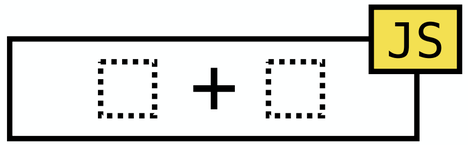


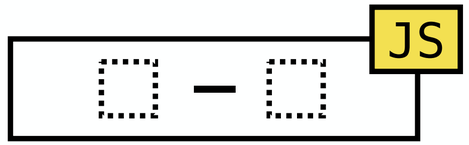


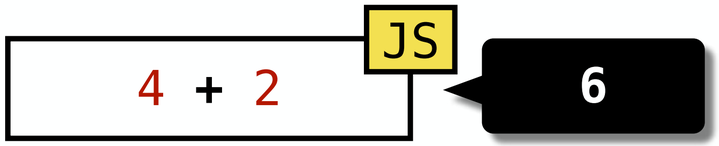


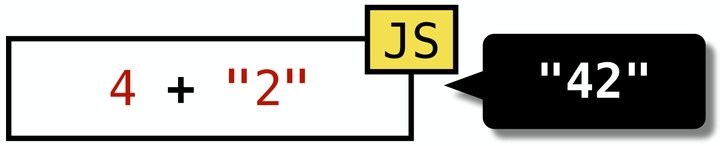


But, **JavaScript** is Complicated

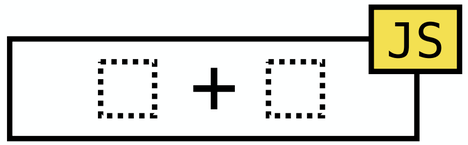


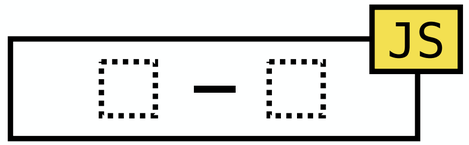


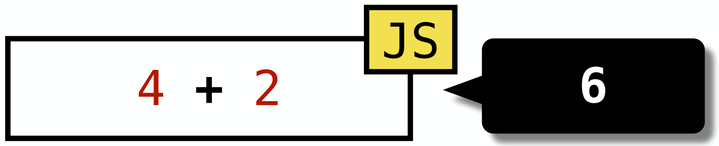


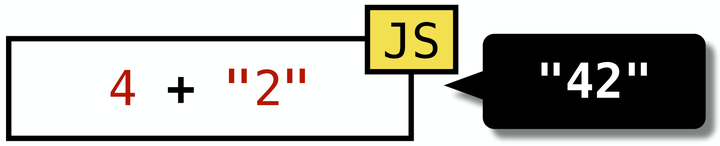


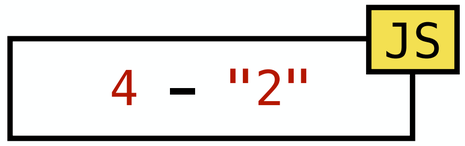
But, **JavaScript** is Complicated



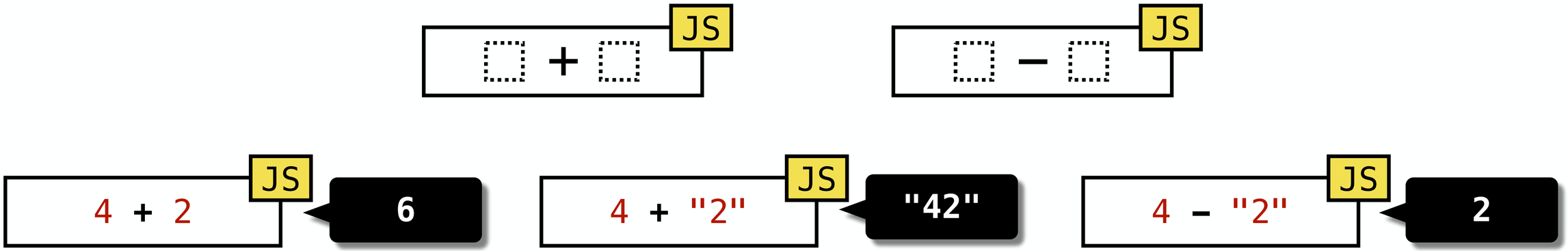




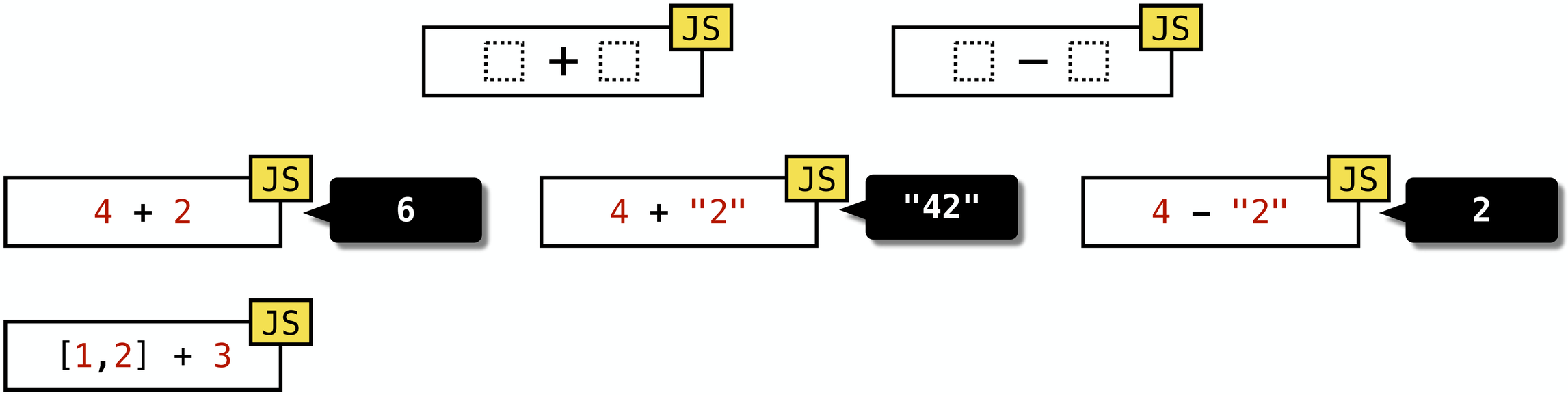




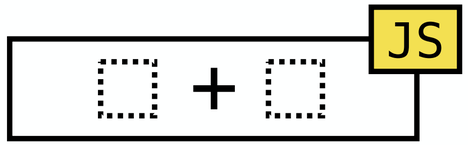
But, **JavaScript** is Complicated

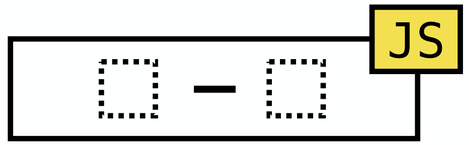


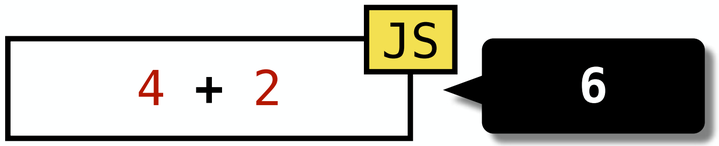
But, **JavaScript** is Complicated

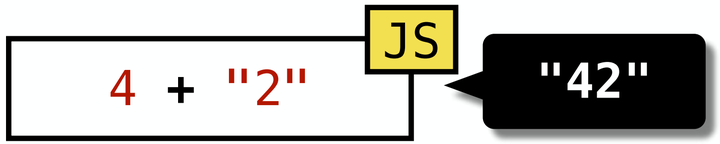


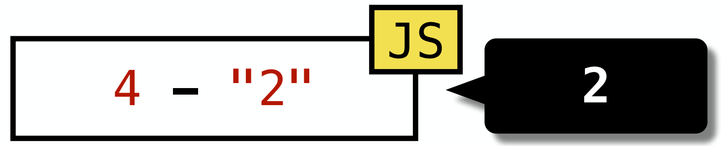
But, **JavaScript** is Complicated

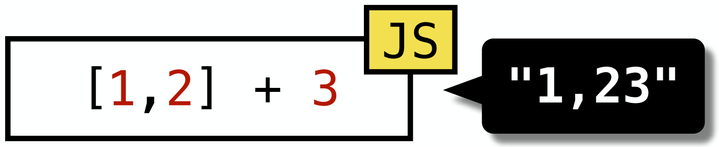




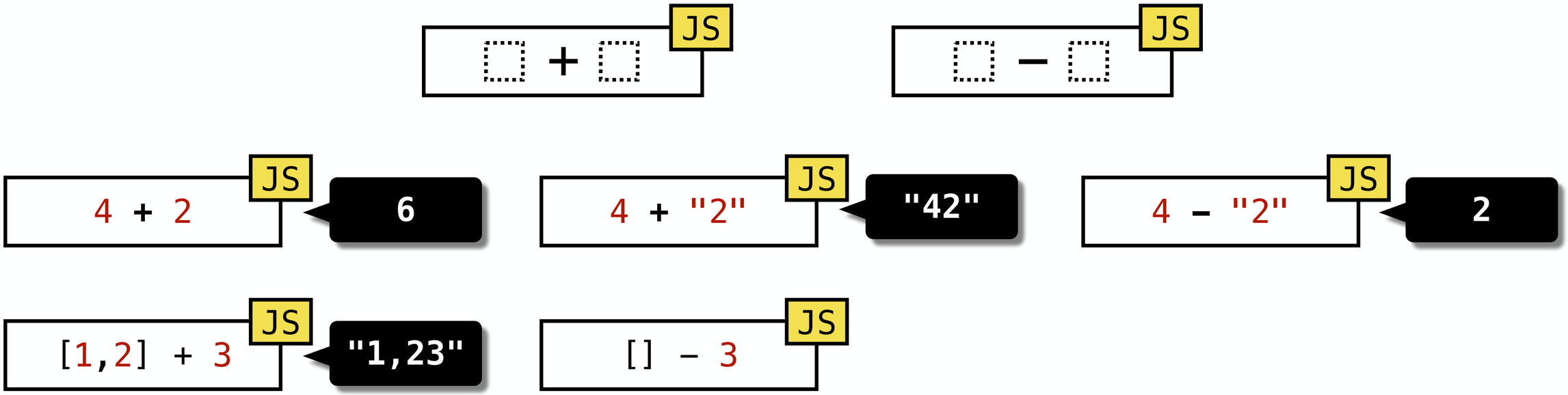




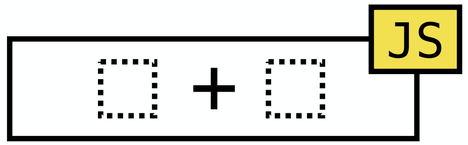


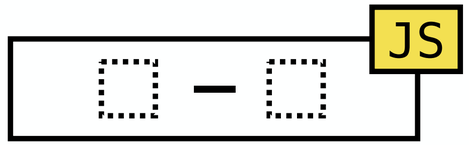


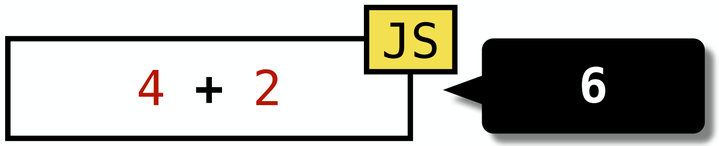
But, **JavaScript** is Complicated

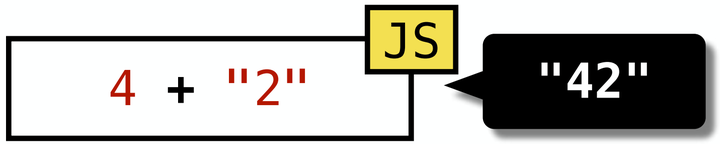


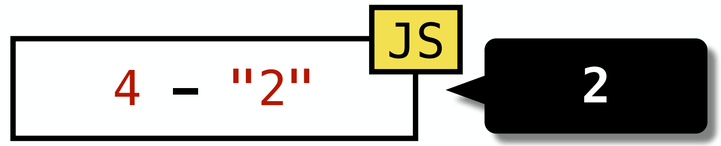
But, **JavaScript** is Complicated

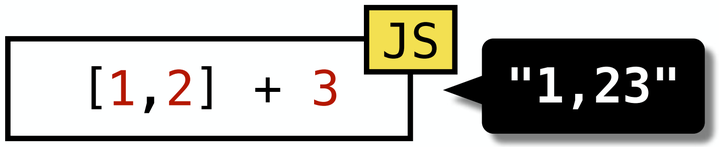


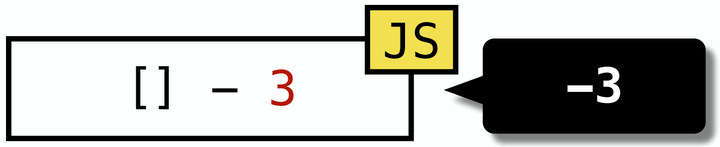




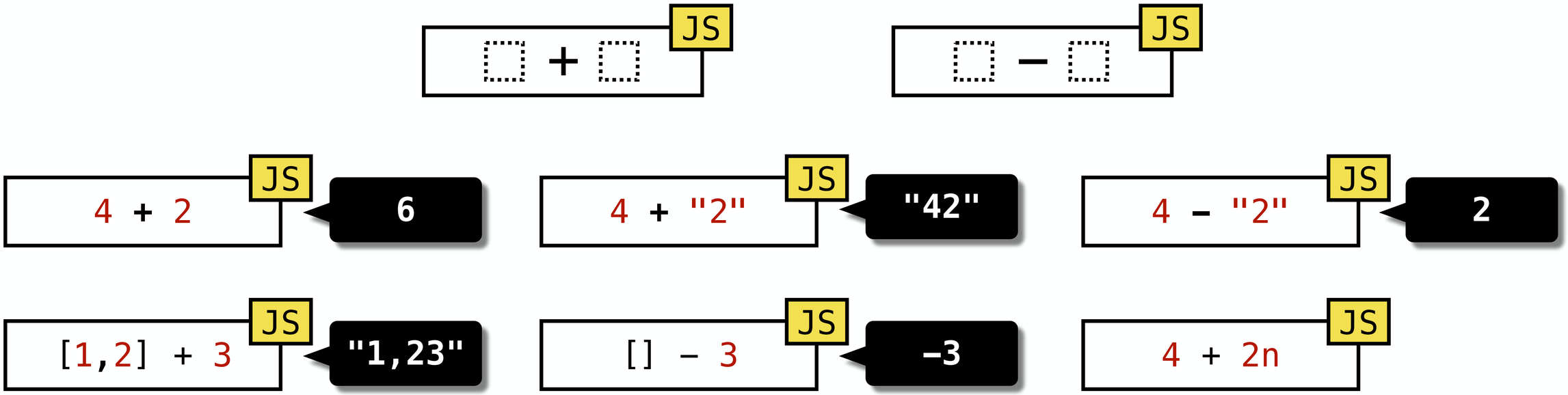




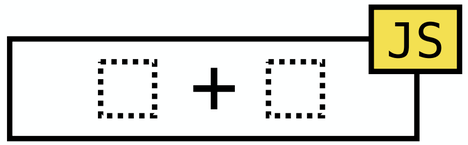


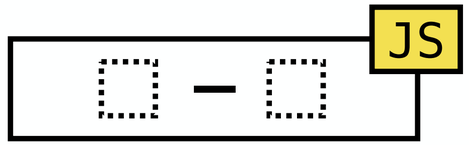


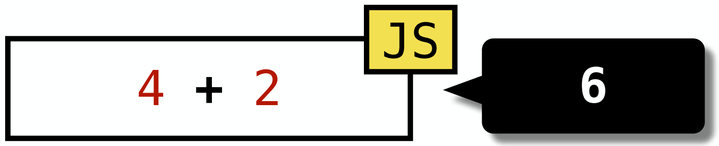
But, **JavaScript** is Complicated

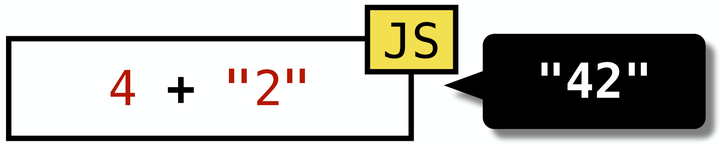


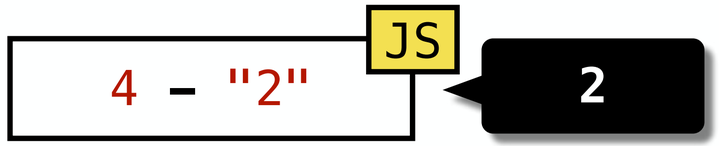
But, **JavaScript** is Complicated

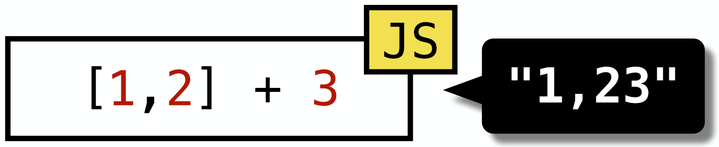


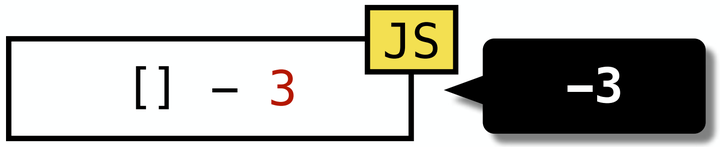


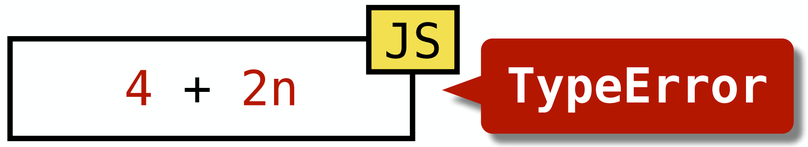




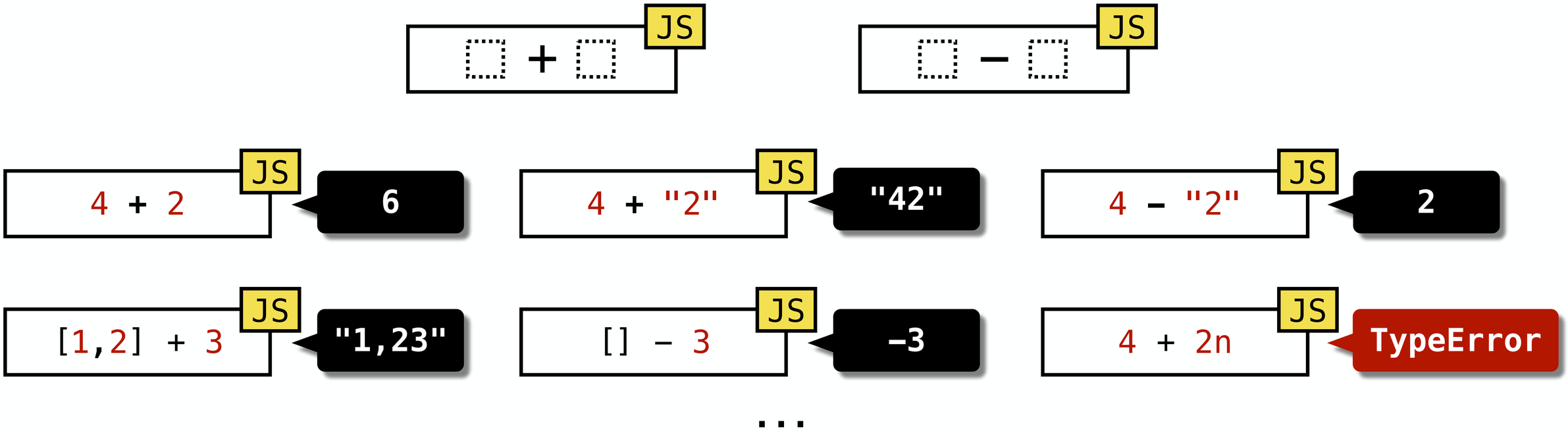




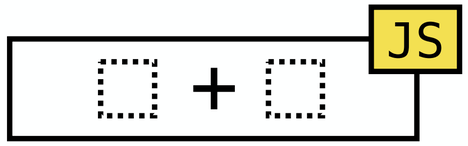


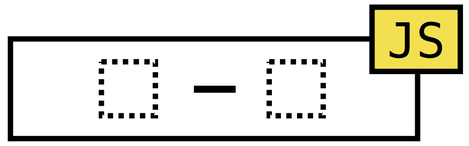


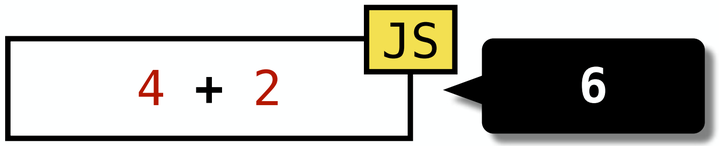
But, **JavaScript** is Complicated

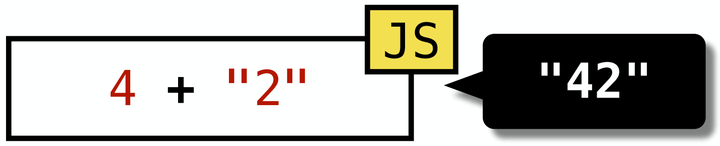


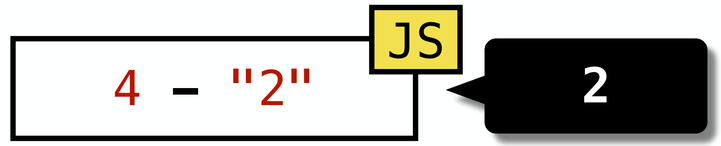
But, **JavaScript** is Complicated

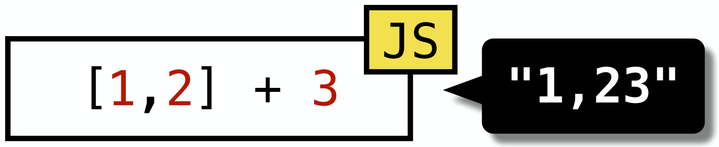


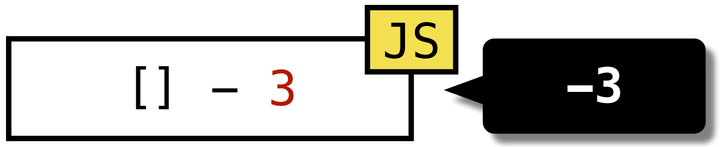


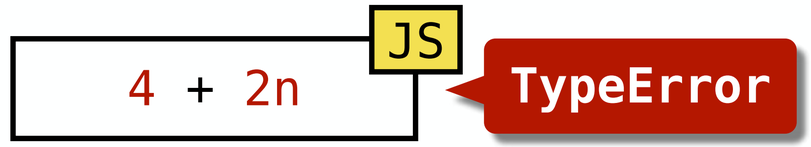




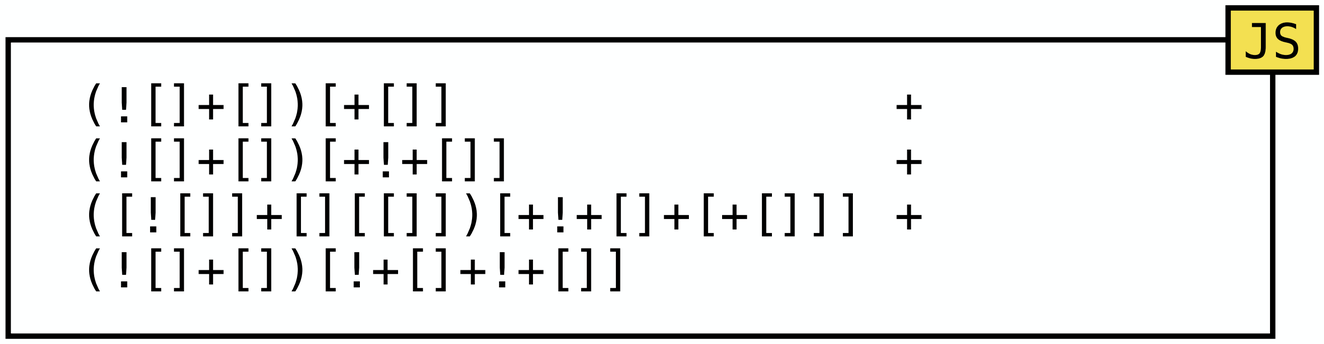




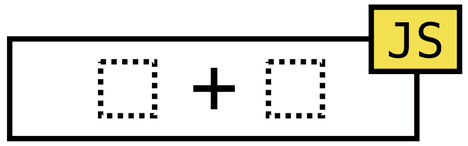


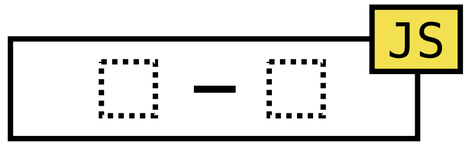


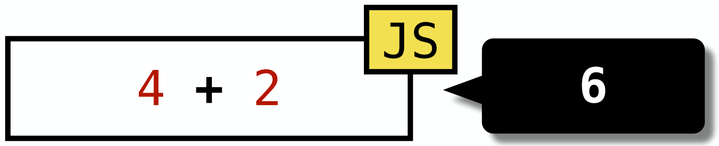
...

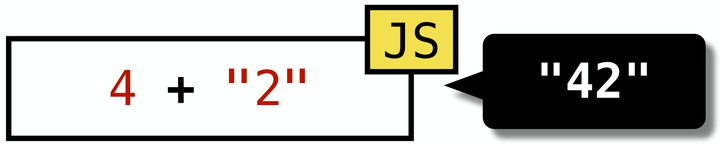


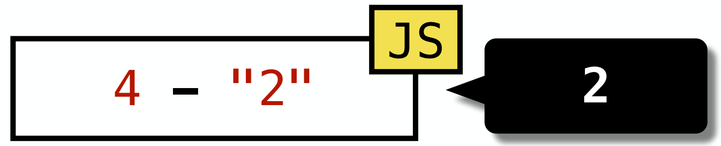
But, **JavaScript** is Complicated

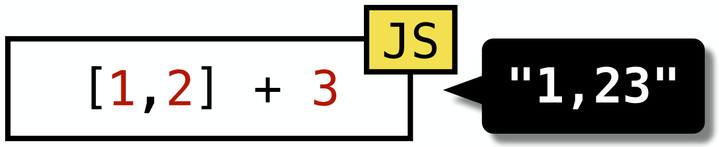


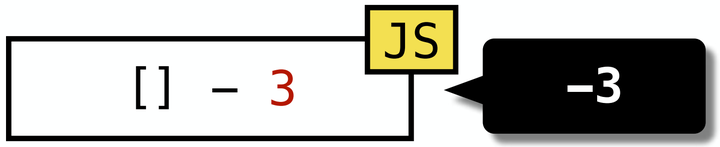


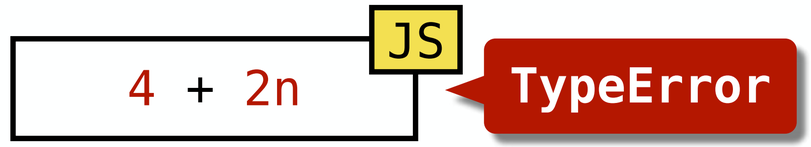




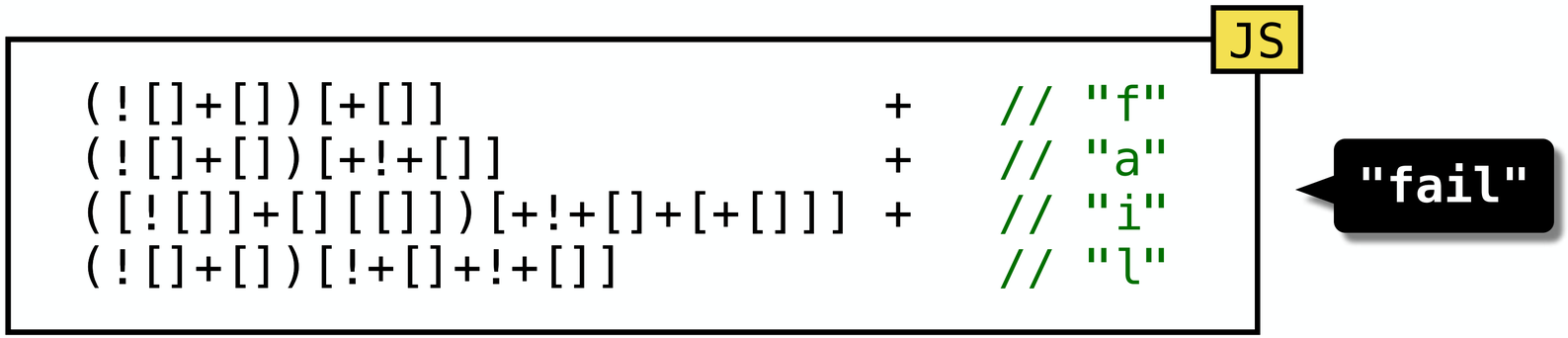




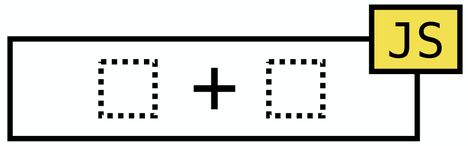


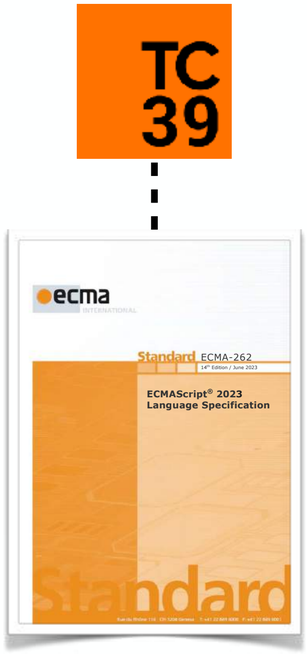


...



Language Specification (ECMA-262) of JavaScript





ECMA-262
(JavaScript Spec.)

Syntax

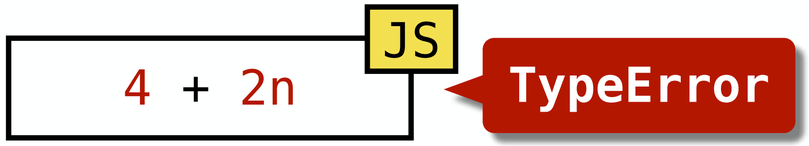
AdditiveExpression`[Yield, Await]` :
MultiplicativeExpression`[?Yield, ?Await]`
AdditiveExpression`[?Yield, ?Await]` **+** *MultiplicativeExpression*`[?Yield, ?Await]`
AdditiveExpression`[?Yield, ?Await]` **−** *MultiplicativeExpression*`[?Yield, ?Await]`

Semantics

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

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

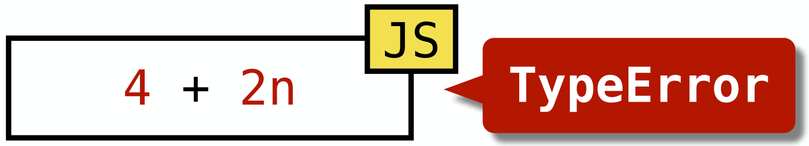
Language Specification (ECMA-262) of JavaScript



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

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

Language Specification (ECMA-262) of JavaScript



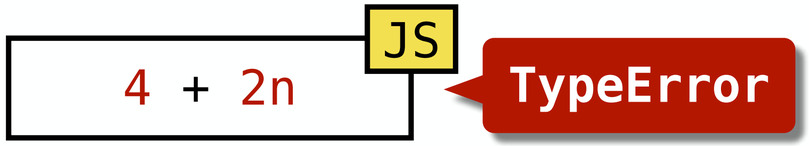
AdditiveExpression : *AdditiveExpression* + *MultiplicativeExpression*

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*).

Language Specification (ECMA-262) of JavaScript



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

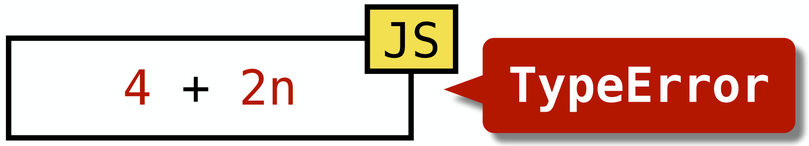
1. Return ? EvaluateStringOrNumericBinaryExpression(*AdditiveExpression*, **+**, *MultiplicativeExpression*).

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*).

Language Specification (ECMA-262) of JavaScript



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

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

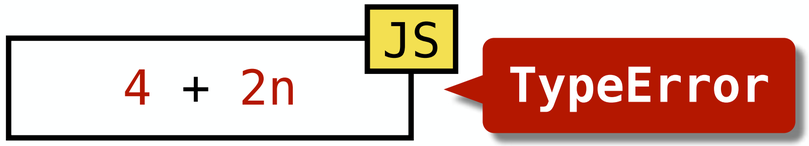
Expr **4** | **+** | Expr **2n**

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



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*).

Evaluate Left

3. Let *rref* be ? Evaluation of *rightOperand*.
4. Let *rval* be ? GetValue(*rref*).

Evaluate Right

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

Language Specification (ECMA-262) of JavaScript

```
4 + 2n
```
JS

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*).

Evaluate Left

3. Let *rref* be ? Evaluation of *rightOperand*.
4. Let *rval* be ? GetValue(*rref*).

Evaluate Right

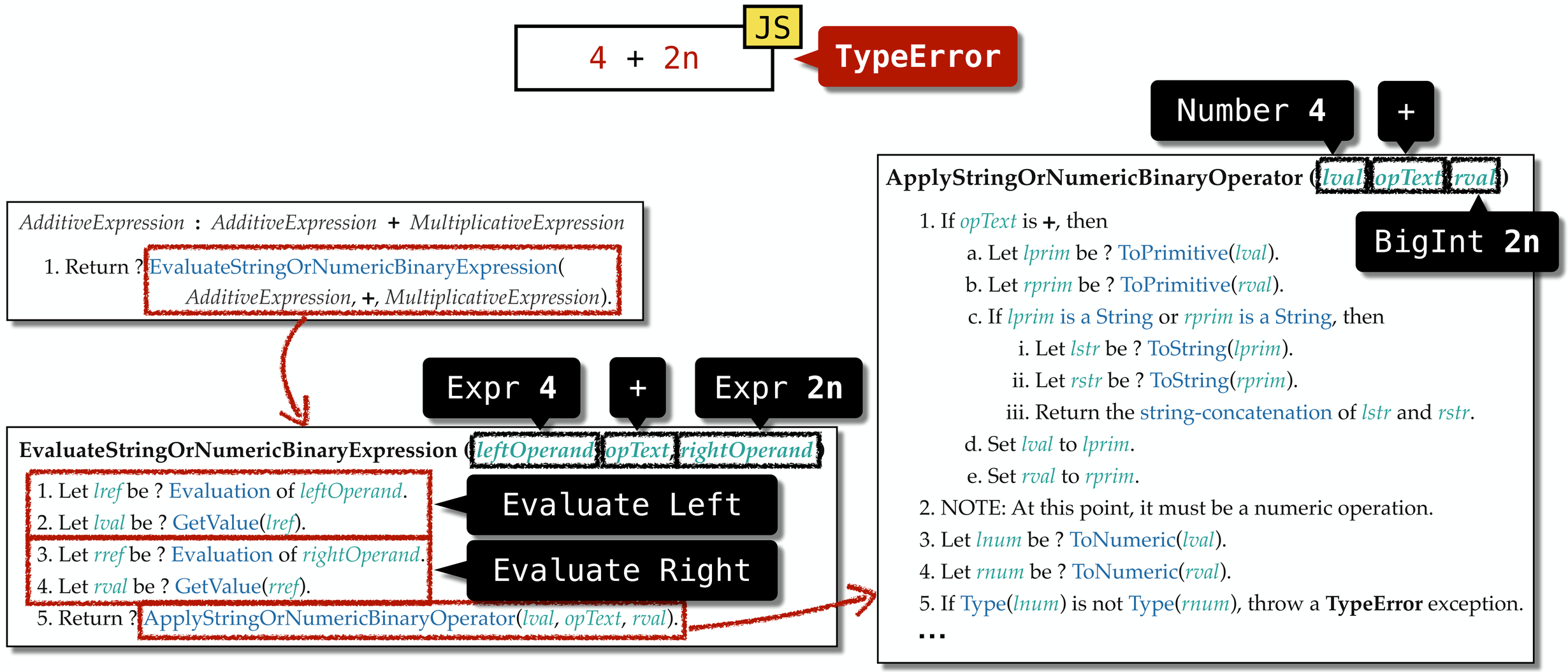
5. Return ? ApplyStringOrNumericBinaryOperator(*lval*, *opText*, *rval*).

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

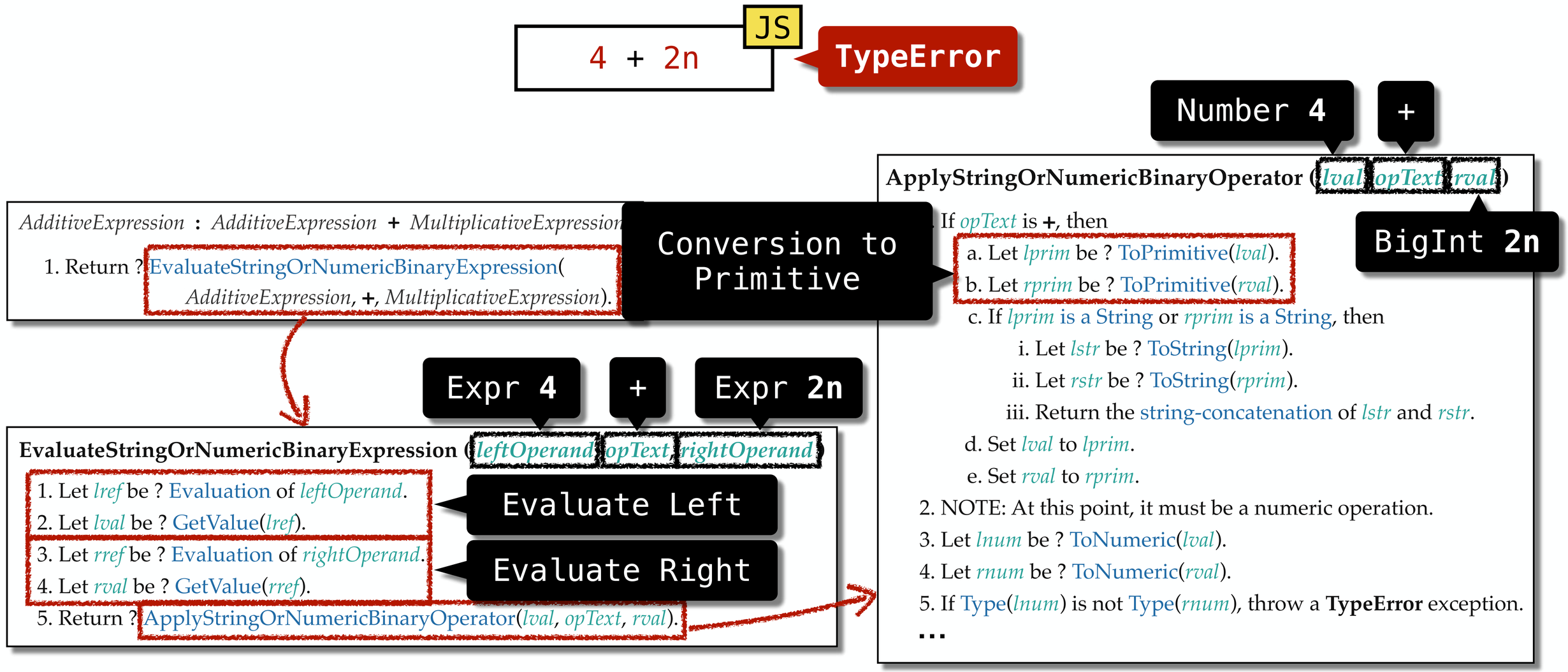
1. If *opText* is **+**, then
    a. Let *lprim* be ? ToPrimitive(*lval*).
    b. Let *rprim* be ? ToPrimitive(*rval*).
    c. If *lprim* is a String or *rprim* is a String, then
        i. Let *lstr* be ? ToString(*lprim*).
        ii. Let *rstr* be ? ToString(*rprim*).
        iii. Return the string-concatenation 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.

...

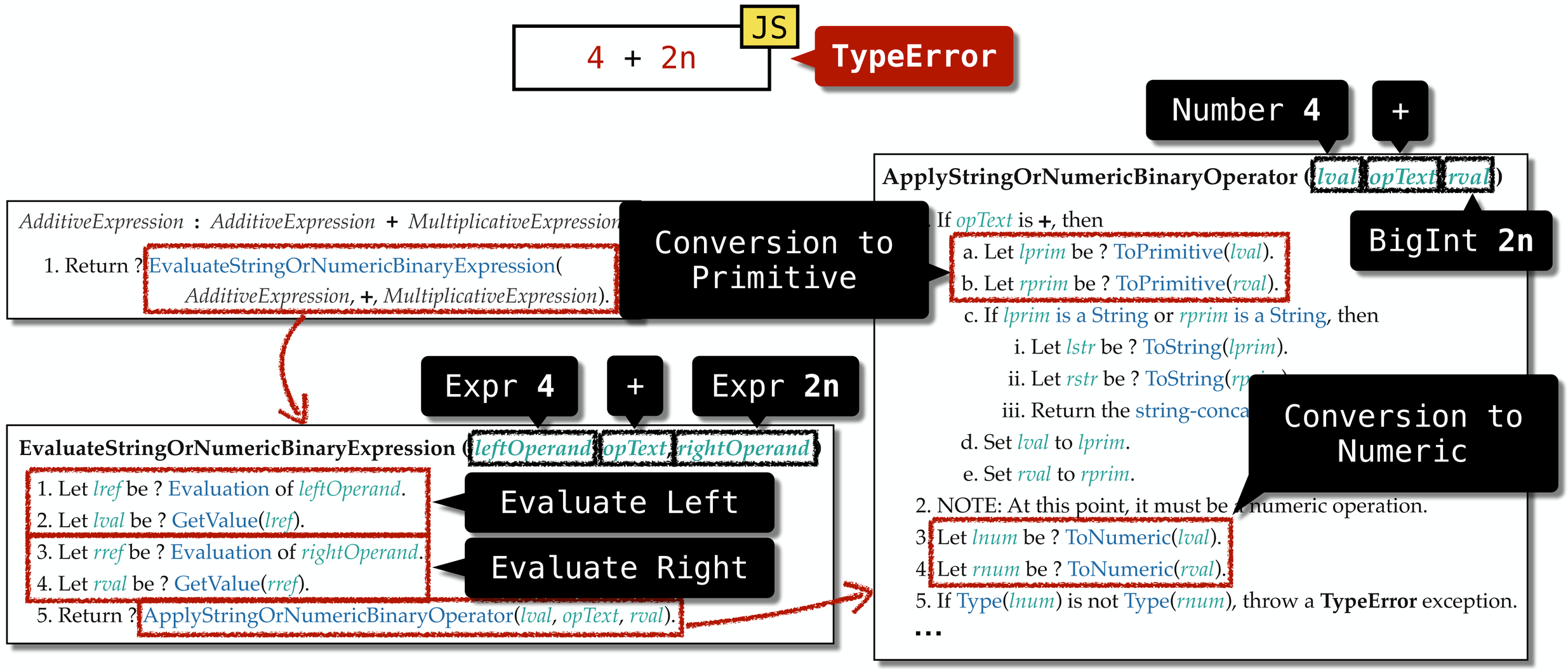
Language Specification (ECMA-262) of JavaScript



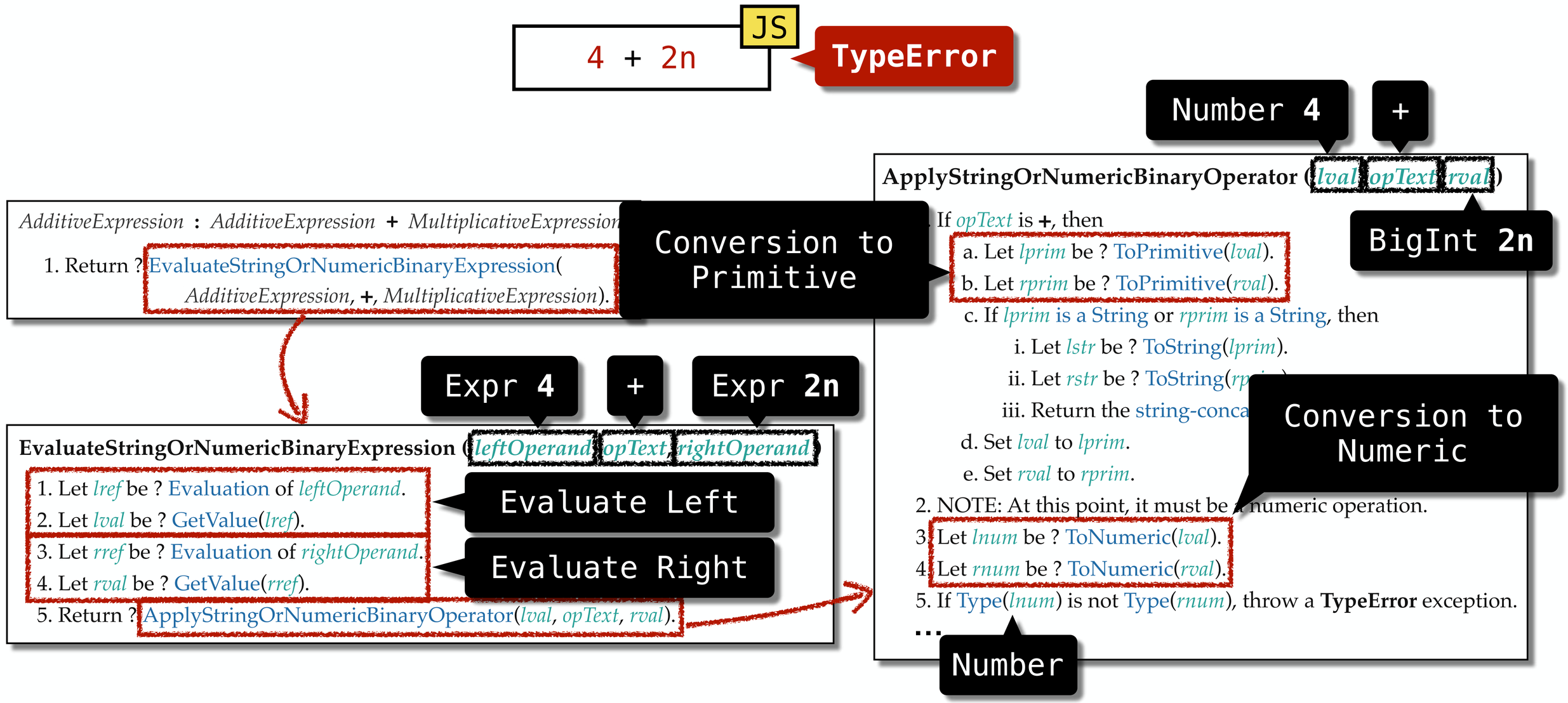
Language Specification (ECMA-262) of JavaScript



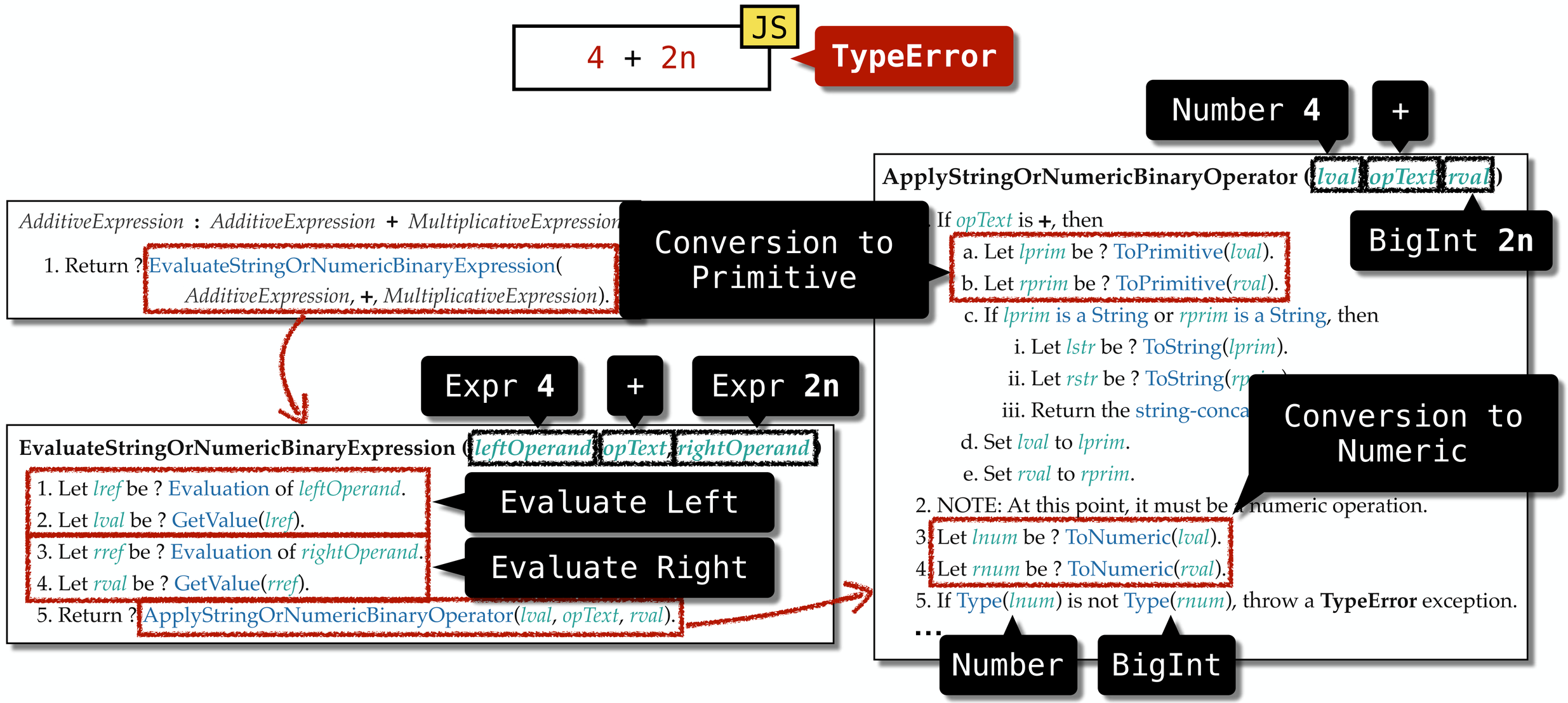
Language Specification (ECMA-262) of JavaScript



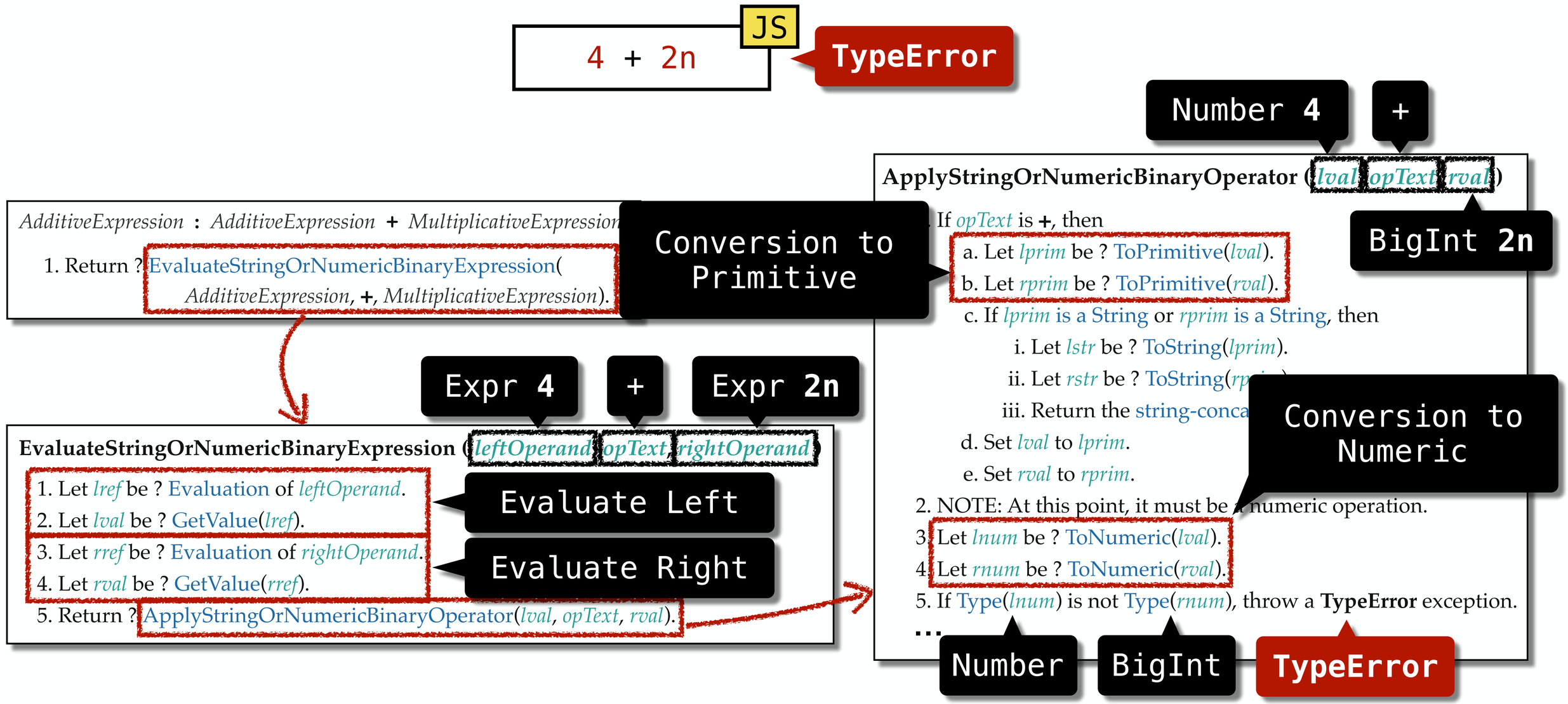
Language Specification (ECMA-262) of JavaScript



Language Specification (ECMA-262) of JavaScript

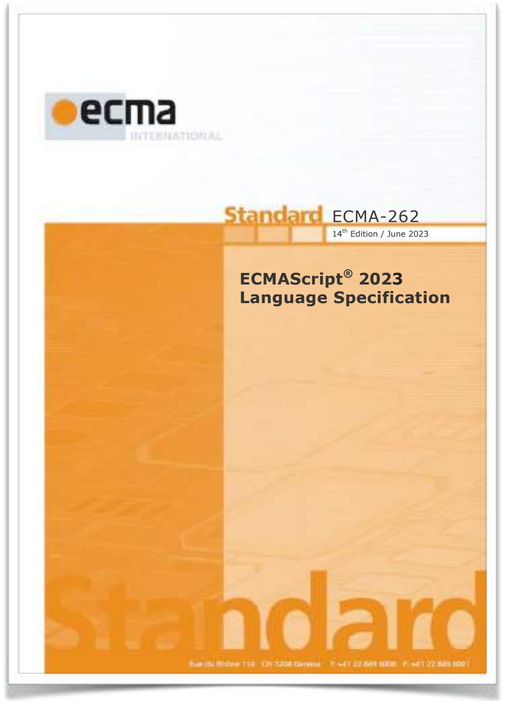


Language Specification (ECMA-262) of JavaScript



Design and Implementation of JavaScript

Update



ECMA-262
(JavaScript Spec.)

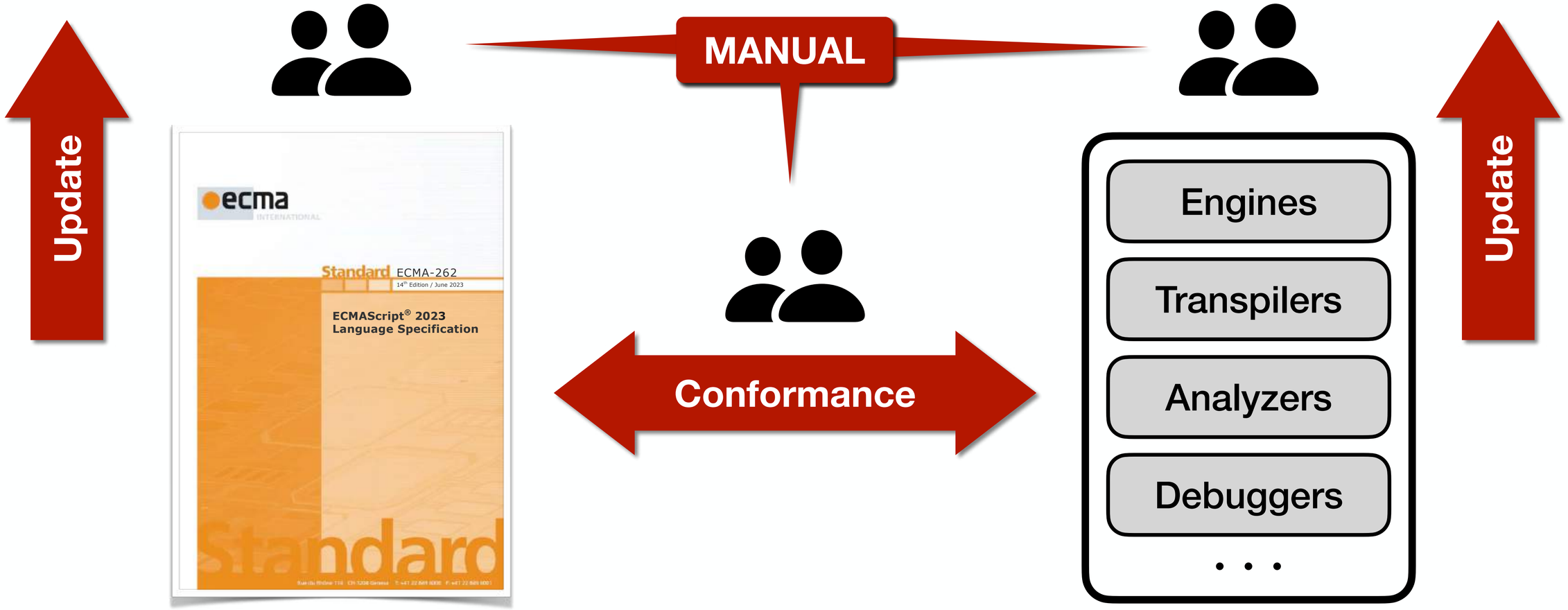
Conformance

Engines
Transpilers
Analyzers
Debuggers
• • •

JavaScript Implementations

Update

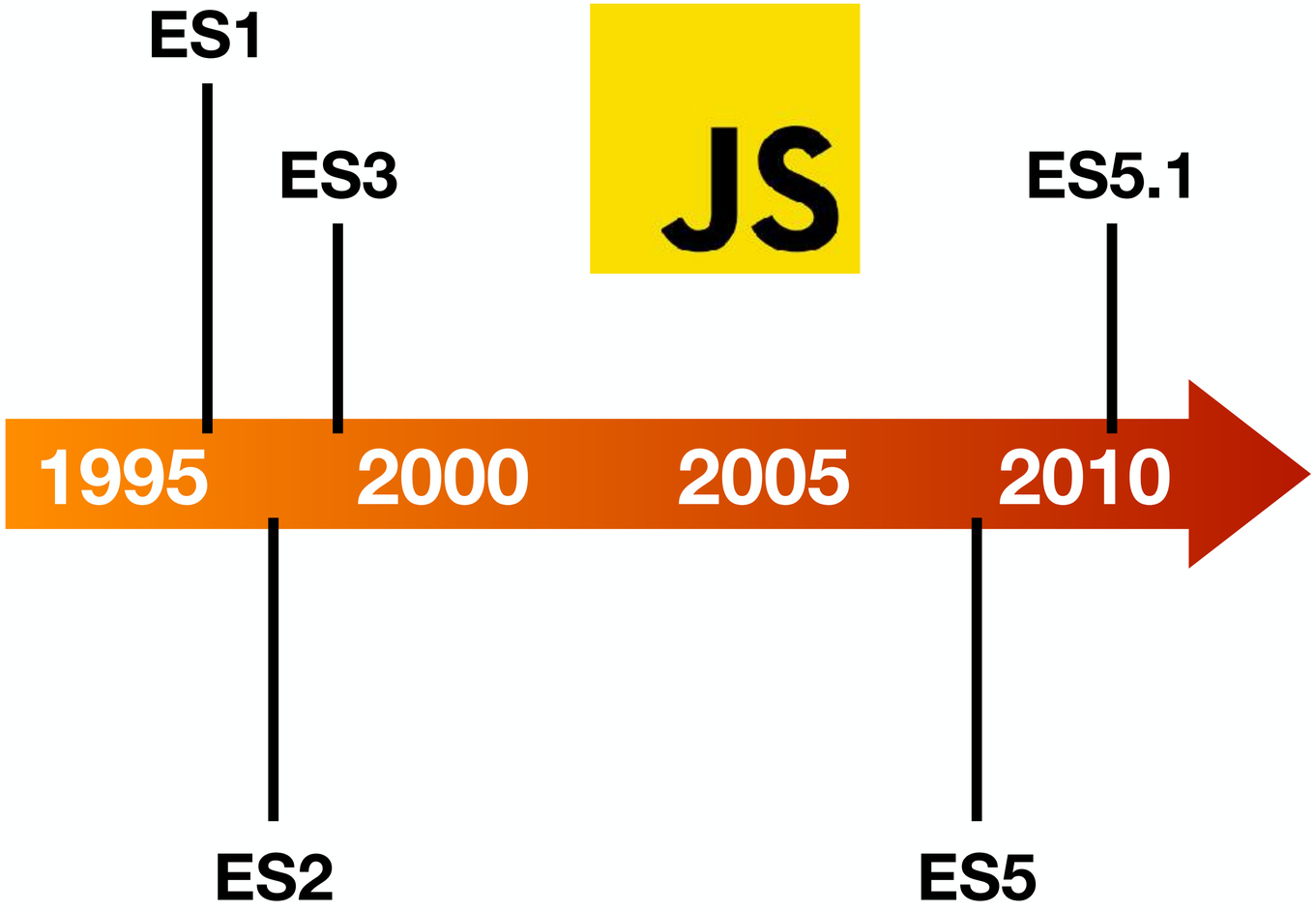
Design and Implementation of JavaScript



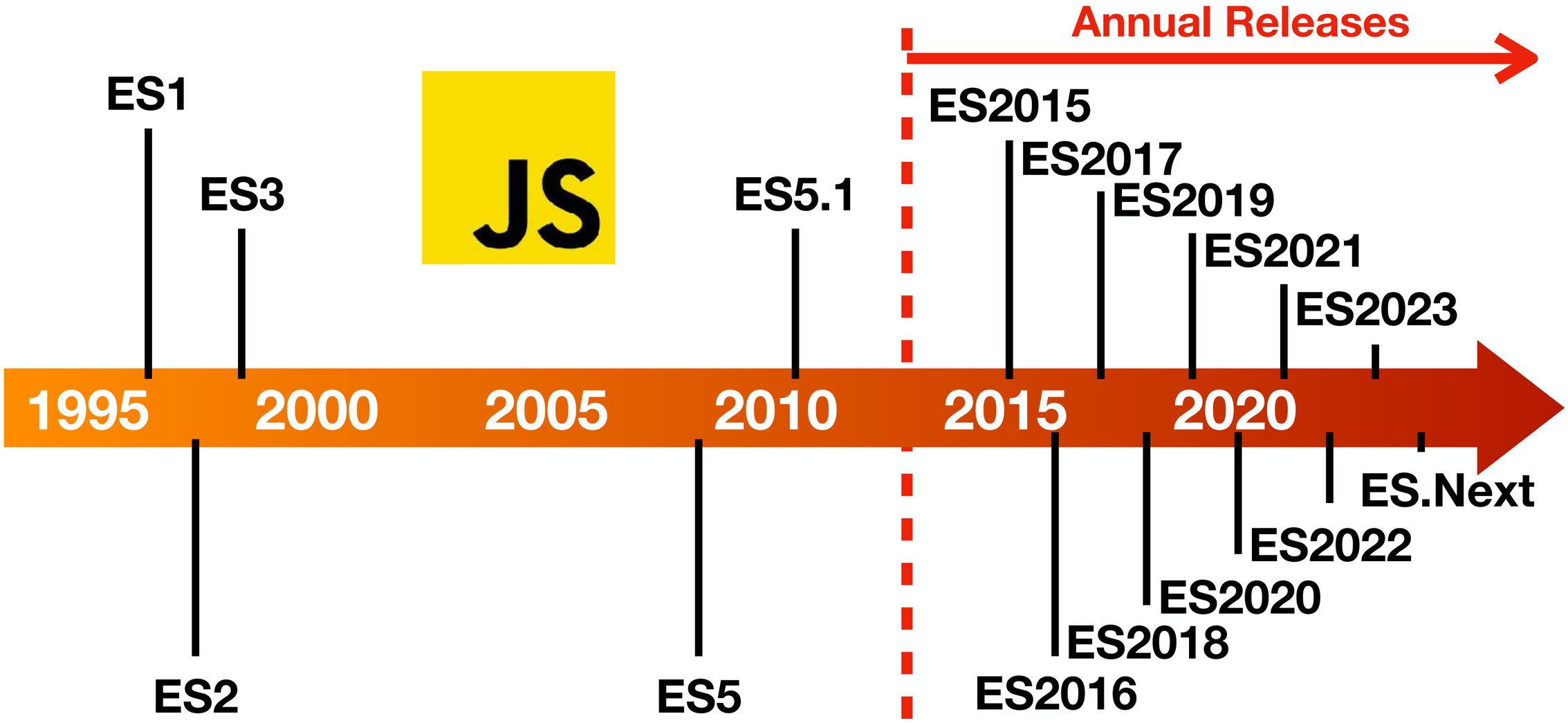
ECMA-262
(JavaScript Spec.)

JavaScript Implementations

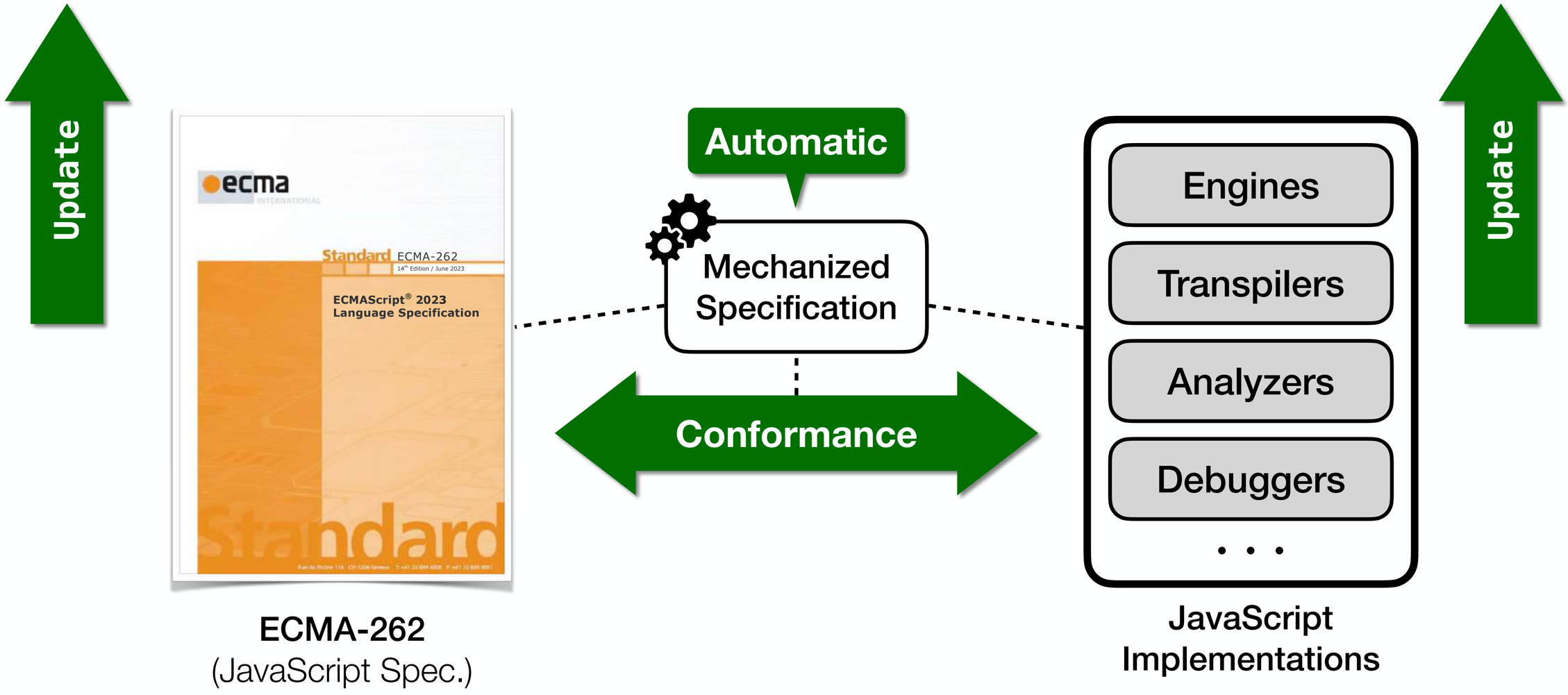
Problem - Fast Evolving JavaScript

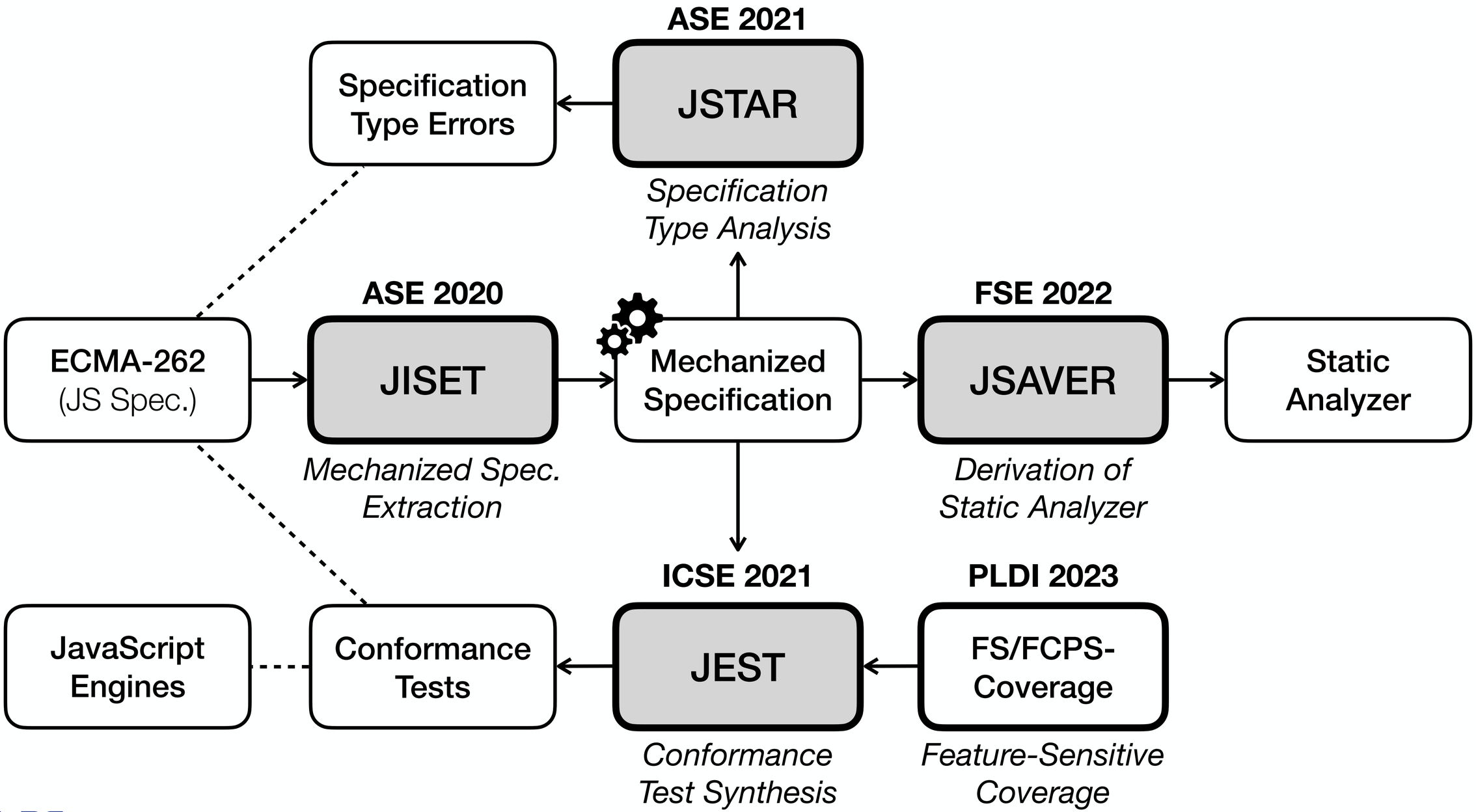


Problem - Fast Evolving JavaScript

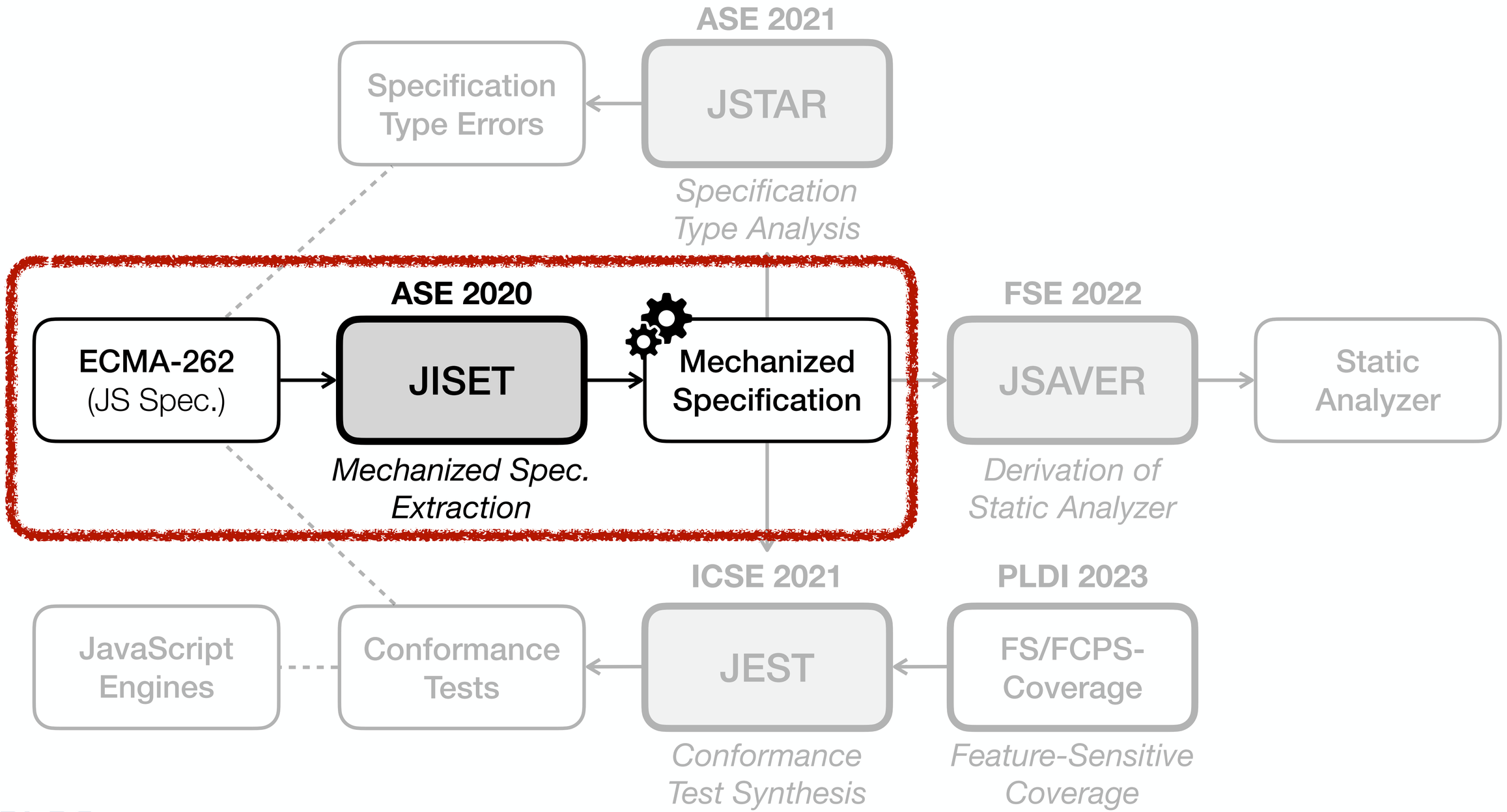


Solution - Mechanized Language Specification

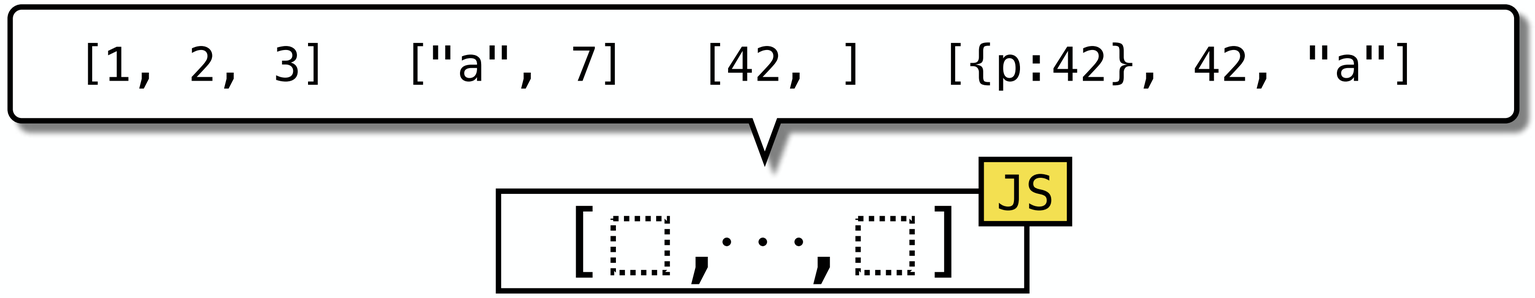



ASE 2021
Specification Type Errors
JSTAR
Specification Type Analysis
ASE 2020
FSE 2022
ECMA-262 (JS Spec.)
JISET
Mechanized Specification
JSAVER
Static Analyzer
Mechanized Spec. Extraction
Derivation of Static Analyzer
ICSE 2021
PLDI 2023
JavaScript Engines
Conformance Tests
JEST
FS/FCPS-Coverage
Conformance Test Synthesis
Feature-Sensitive Coverage

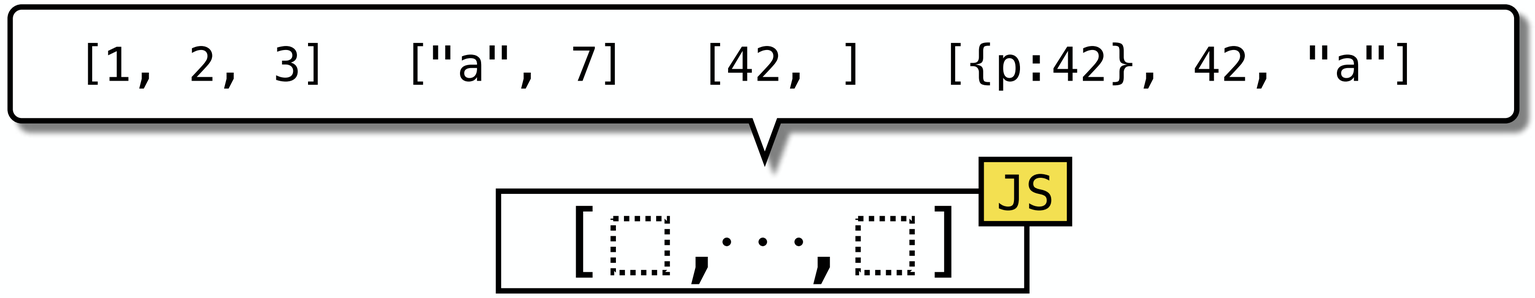
[ASE 2020] J. Park et al., “JISET: JavaScript IR-based Semantics Extraction Toolchain”



Language Specification (ECMA-262) of JavaScript



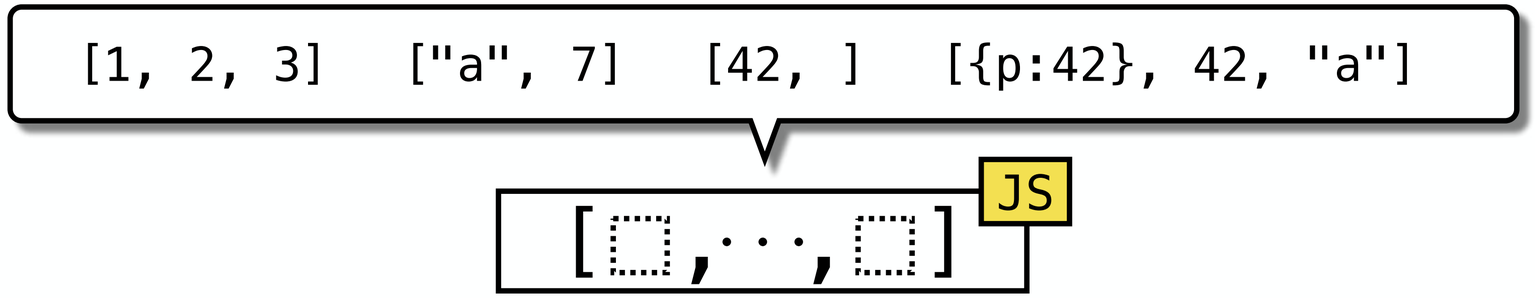
Language Specification (ECMA-262) of JavaScript



Syntax

ArrayLiteral[Yield, Await] :
- [*Elision*opt]
- [*ElementList*[?Yield, ?Await]]
- [*ElementList*[?Yield, ?Await] , *Elision*opt]

Language Specification (ECMA-262) of JavaScript



Syntax

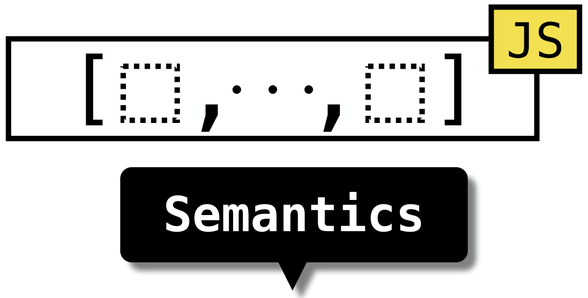
ArrayLiteral[Yield, Await] :
- [*Elision*opt]
- [*ElementList*[?Yield, ?Await]]
- [*ElementList*[?Yield, ?Await] , *Elision*opt]

Semantics

ArrayLiteral : [*ElementList* , *Elision*opt]

1. Let *array* be ! ArrayCreate(0).
2. Let *nextIndex* be ? ArrayAccumulation of *ElementList* with arguments *array* and 0.
3. If *Elision* is present, then
    a. Perform ? ArrayAccumulation of *Elision* with arguments *array* and *nextIndex*.
4. Return *array*.

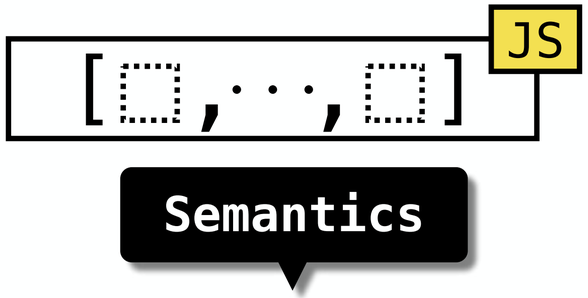
JISET - Patterns in Abstract Algorithms



ArrayLiteral : `[` *ElementList* `,` *Elision*$_{opt}$ `]`

1. Let *array* be ! ArrayCreate(0).
2. Let *nextIndex* be ? ArrayAccumulation of *ElementList* with arguments *array* and 0.
3. If *Elision* is present, then
    a. Perform ? ArrayAccumulation of *Elision* with arguments *array* and *nextIndex*.
4. Return *array*.

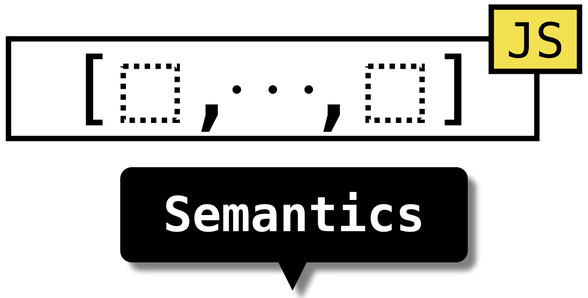
JISET - Patterns in Abstract Algorithms



ArrayLiteral : `[` *ElementList* `,` *Elision*$_{opt}$ `]`

1. Let *array* be ! ArrayCreate(0).
2. Let *nextIndex* be ? ArrayAccumulation of *ElementList* with arguments *array* and 0.
3. If *Elision* is present, then
    a. Perform ? ArrayAccumulation of *Elision* with arguments *array* and *nextIndex*.
4. Return *array*.

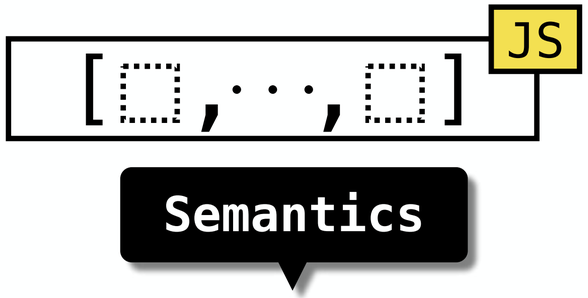
JISET - Patterns in Abstract Algorithms



ArrayLiteral : `[` *ElementList* `,` *Elision*$_{opt}$ `]`

1. Let *array* be ! ArrayCreate(0).
2. Let *nextIndex* be ? ArrayAccumulation of *ElementList* with arguments *array* and 0.
3. If *Elision* is present, then
    a. Perform ? ArrayAccumulation of *Elision* with arguments *array* and *nextIndex*.
4. Return *array*.

JISET - Patterns in Abstract Algorithms



ArrayLiteral : `[` *ElementList* `,` *Elision*$_{opt}$ `]`

1. Let *array* be ! ArrayCreate(0).
2. Let *nextIndex* be ? ArrayAccumulation of *ElementList* with arguments *array* and 0.
3. If *Elision* is present, then
    a. Perform ? ArrayAccumulation of *Elision* with arguments *array* and *nextIndex*.
4. Return *array*.

JISET - Metalanguage for ECMA-262

(**IR$_{ES}$ - Intermediate Representation** for ECMA-262)

$$
\begin{array}{lll}
\text{Programs} & \mathfrak{P} \ni P & ::= f^* \\
\text{Functions} & \mathcal{F} \ni f & ::= \texttt{syntax}^? \ \texttt{def x(x}^*\texttt{) \{}[\ell : i]^*\texttt{\}} \\
\text{Variables} & \mathcal{X} \ni \texttt{x} & \\
\text{Labels} & \mathcal{L} \ni \ell & \\
\text{Instructions} & \mathcal{I} \ni i & ::= r := e \mid \texttt{x} := \texttt{\{\}} \mid \texttt{x} := e(e^*) \\
 & & \quad \mid \ \texttt{if}\ e\ \ell\ \ell \mid \texttt{return}\ e \\
\text{Expressions} & \mathcal{E} \ni e & ::= v^{\text{p}} \mid \texttt{op}(e^*) \mid r \\
\text{References} & \mathcal{R} \ni r & ::= \texttt{x} \mid e\texttt{[}e\texttt{]} \mid e\texttt{[}e\texttt{]}_{\text{js}} \\
 & & \quad \vdots \\
\text{Values} & v \in \mathbb{V} & = \mathbb{A} \uplus \mathbb{V}^{\text{p}} \uplus \mathbb{T} \uplus \mathcal{F} \\
\text{Primitive Values} & v^{\text{p}} \in \mathbb{V}^{\text{p}} & = \mathbb{V}_{\text{bool}} \uplus \mathbb{V}_{\text{int}} \uplus \mathbb{V}_{\text{str}} \uplus \cdots \\
\text{JS ASTs} & t \in \mathbb{T} &
\end{array}
$$

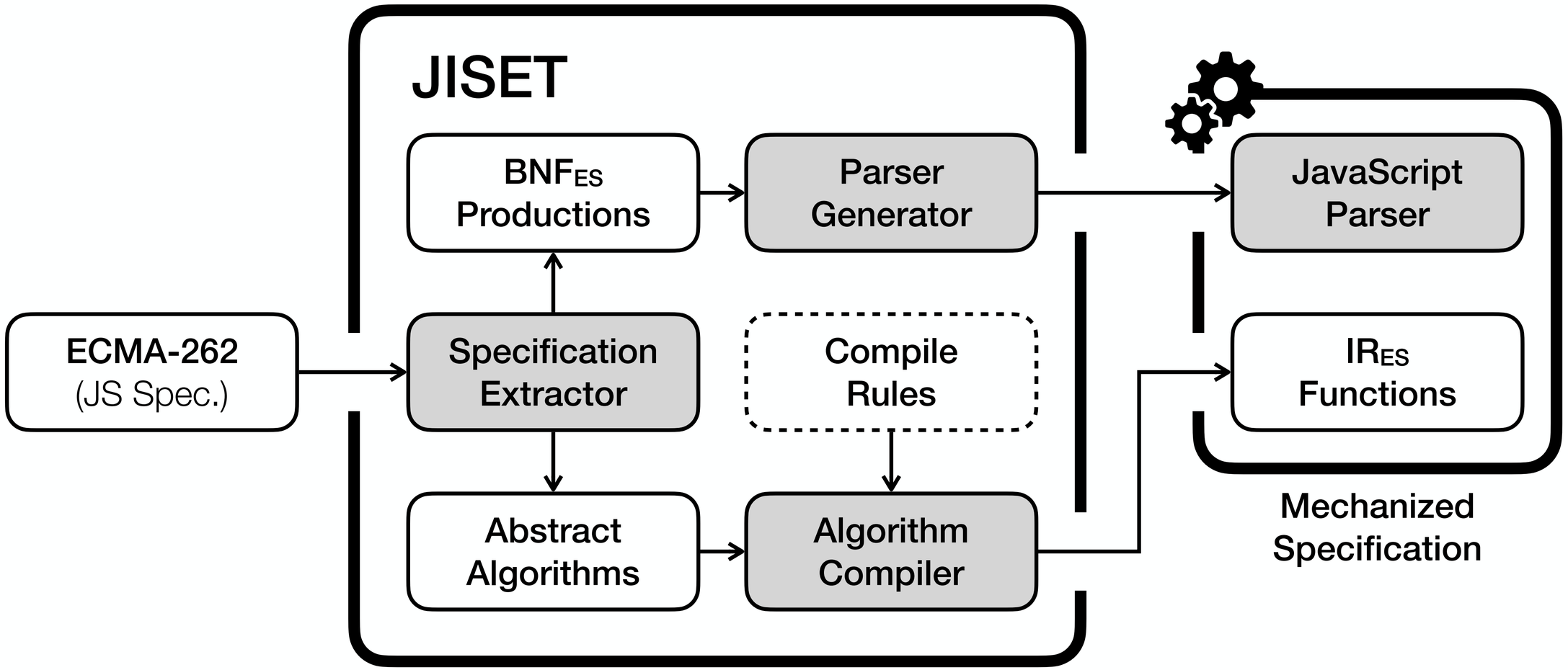
JISET - Metalanguage for ECMA-262

(**IR$_{ES}$ - Intermediate Representation** for ECMA-262)

$$
\begin{array}{lll}
\text{Programs} & \mathfrak{P} \ni P & ::= f^* \\
\text{Functions} & \mathcal{F} \ni f & ::= \texttt{syntax}^? \ \texttt{def}\ \texttt{x}(\texttt{x}^*)\ \{[\ell : i]^*\} \\
\text{Variables} & \mathcal{X} \ni \texttt{x} & \\
\text{Labels} & \mathcal{L} \ni \ell & \\
\text{Instructions} & \mathcal{I} \ni i & ::= r := e \mid \texttt{x} := \{\} \mid \texttt{x} := e(e^*) \\
 & & \quad \mid \ \texttt{if}\ e\ \ell\ \ell \mid \texttt{return}\ e \\
\text{Expressions} & \mathcal{E} \ni e & ::= v^{\text{p}} \mid \texttt{op}(e^*) \mid r \\
\text{References} & \mathcal{R} \ni r & ::= \texttt{x} \mid e[e] \mid e[e]_{\text{js}} \\
 & & \vdots \\
\text{Values} & v \in \mathbb{V} & = \mathbb{A} \uplus \mathbb{V}^{\text{p}} \uplus \mathbb{T} \uplus \mathcal{F} \\
\text{Primitive Values} & v^{\text{p}} \in \mathbb{V}^{\text{p}} & = \mathbb{V}_{\text{bool}} \uplus \mathbb{V}_{\text{int}} \uplus \mathbb{V}_{\text{str}} \uplus \cdots \\
\text{JS ASTs} & t \in \mathbb{T} &
\end{array}
$$

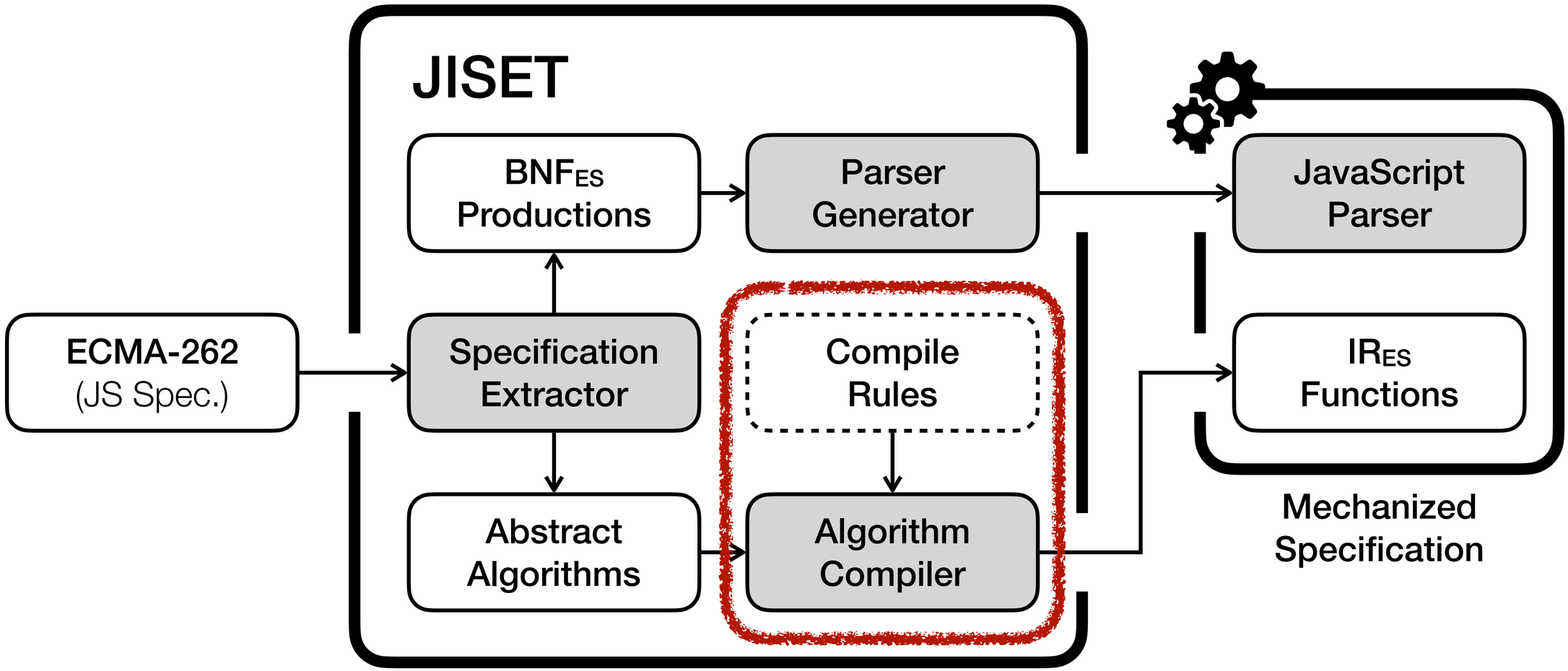
JISET

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



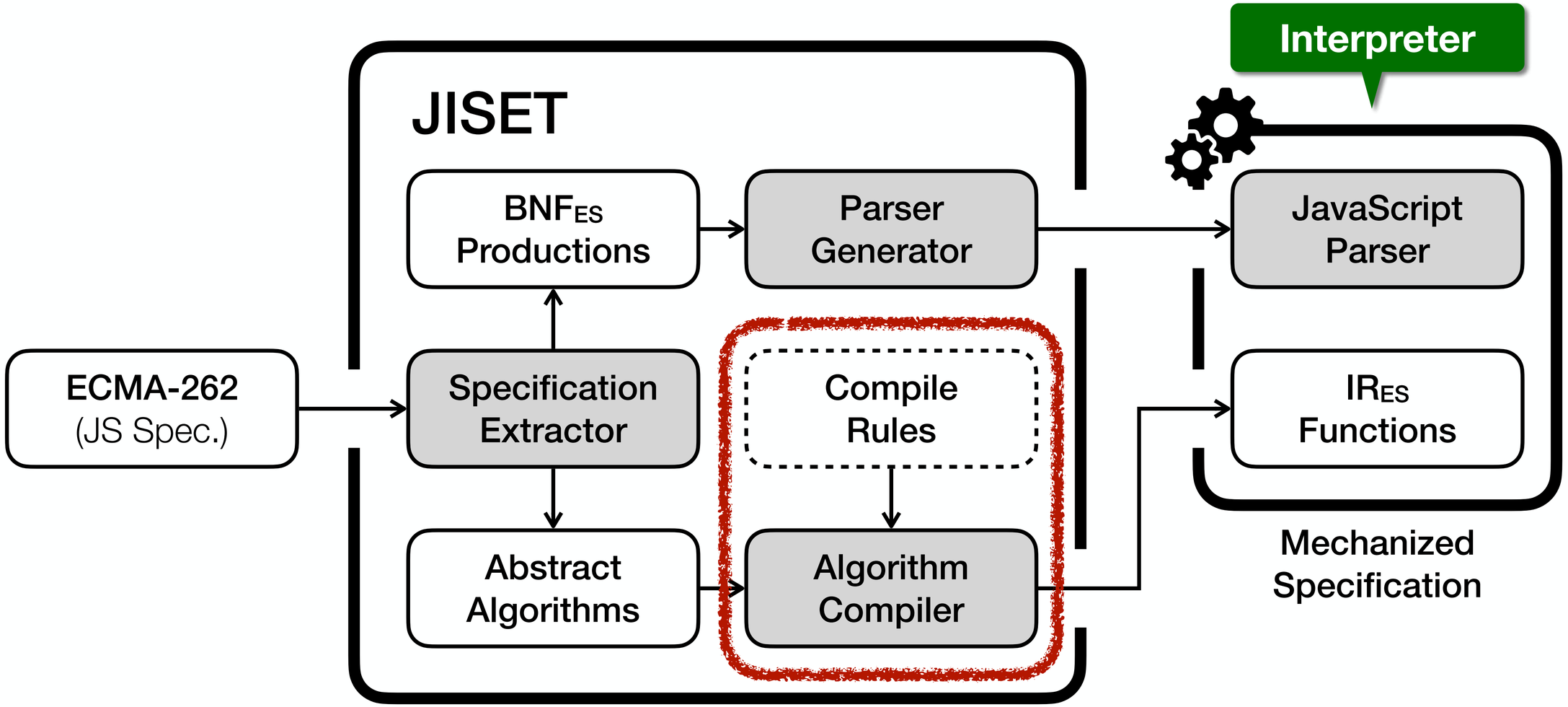
JISET

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



JISET

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

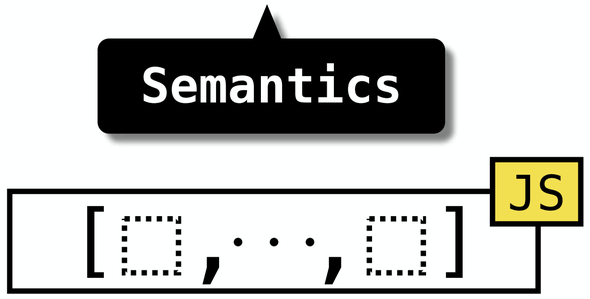


JISET - Algorithm Compiler

Abstract algorithm for *ArrayLiteral* in ES13

ArrayLiteral : `[` *ElementList* `,` *Elision*$_{opt}$ `]`

1. Let *array* be ! ArrayCreate(0).
2. Let *nextIndex* be ? ArrayAccumulation of *ElementList* with arguments *array* and 0.
3. If *Elision* is present, then
    a. Perform ? ArrayAccumulation of *Elision* with arguments *array* and *nextIndex*.
4. Return *array*.



JISET - Algorithm Compiler

Abstract algorithm for *ArrayLiteral* in ES13

ArrayLiteral : [*ElementList* , *Elision*$_{opt}$]

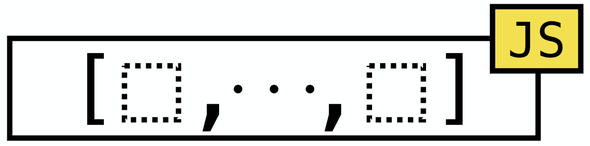
1. Let *array* be ! ArrayCreate(0).
2. Let *nextIndex* be ? ArrayAccumulation of *ElementList* with arguments *array* and 0.
3. If *Elision* is present, then
    a. Perform ? ArrayAccum
        with arguments *array*
4. Return *array*.

118 compile rules for
steps in abstract algorithms

```
syntax def ArrayLiteral[2].Evaluation(
  this, ElementList, Elision
){
  let array = [! (ArrayCreate 0)]
  let nextIndex =
    [? (ElementList.ArrayAccumulation array 0)]
  if (! (= Elision absent))
    [? (Elision.ArrayAccumulation array nextIndex)]
  return array
}
```

IR$_{ES}$ function for *ArrayLiteral* in ES13

Semantics



Parsing rules **Conversion Rules**

```
S = // statements
  Let ~ V ~ be ~ E ~ .   ^^  ILet

E = // expressions
  ! E                    ^^  EAbruptCheck |
  str ~ ( ~ E ~ )        ^^  ECall        |
  num                    ^^ _.toDoub le
```

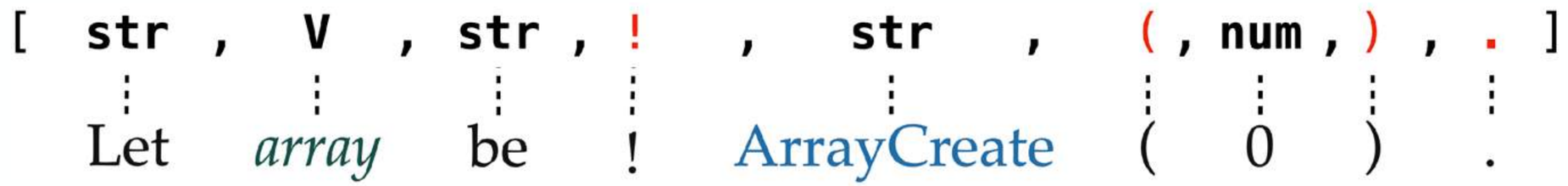
Simplified compile rules

Let *array* be ! ArrayCreate (0) .

Parsing rules **Conversion Rules**

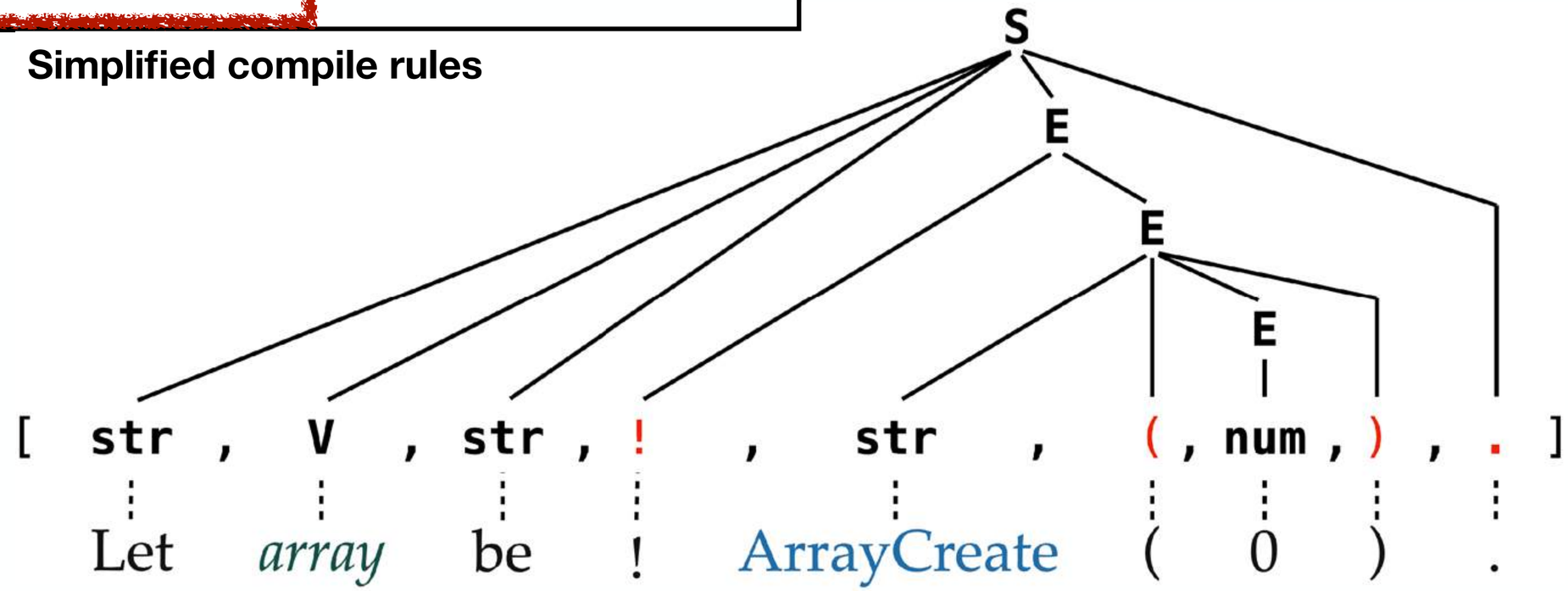
```
S = // statements
  Let ~ V ~ be ~ E ~ .   ^^  ILet

E = // expressions
  ! E                    ^^  EAbruptCheck |
  str ~ ( ~ E ~ )        ^^  ECall        |
  num                    ^^ _.toDoub le
```

Simplified compile rules



Parsing rules	Conversion Rules
`S = // statements`	
`  Let ~ V ~ be ~ E ~ .`	`^^  ILet`
`E = // expressions`	
`  ! E`	`^^  EAbruptCheck |`
`  str ~ ( ~ E ~ )`	`^^  ECall |`
`  num`	`^^ _.toDoub le`

Simplified compile rules



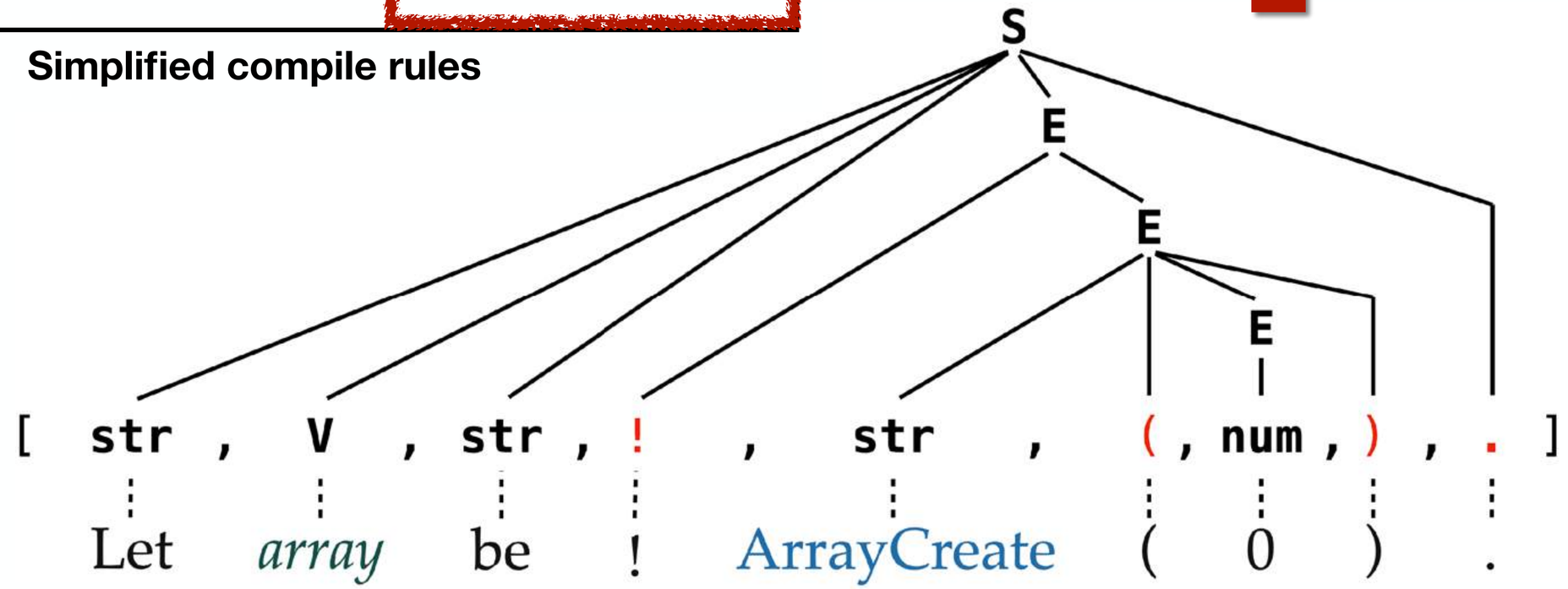
Parsing rules | **Conversion Rules**

```
S = // statements
  Let ~ V ~ be ~ E ~ .   ^^  ILet

E = // expressions
  ! E                    ^^  EAbruptCheck |
  str ~ ( ~ E ~ )        ^^  ECall        |
  num                    ^^  _.toDouble
```

```
let array = [! (ArrayCreate 0)]
```

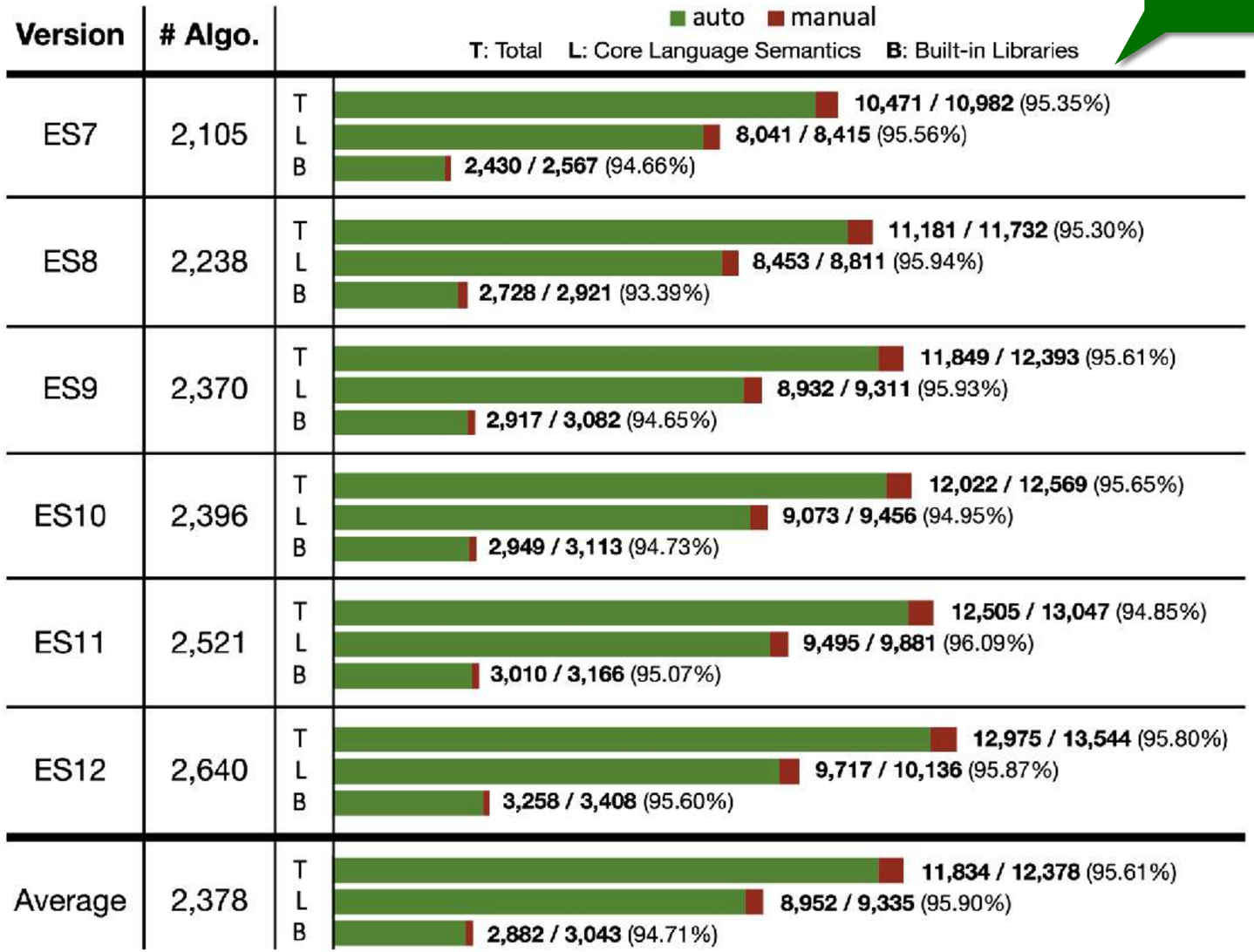
```
ILet(array, EAbruptCheck(
  ECall("ArrayCreate", 0)))
```

Simplified compile rules



JISET - Evaluation

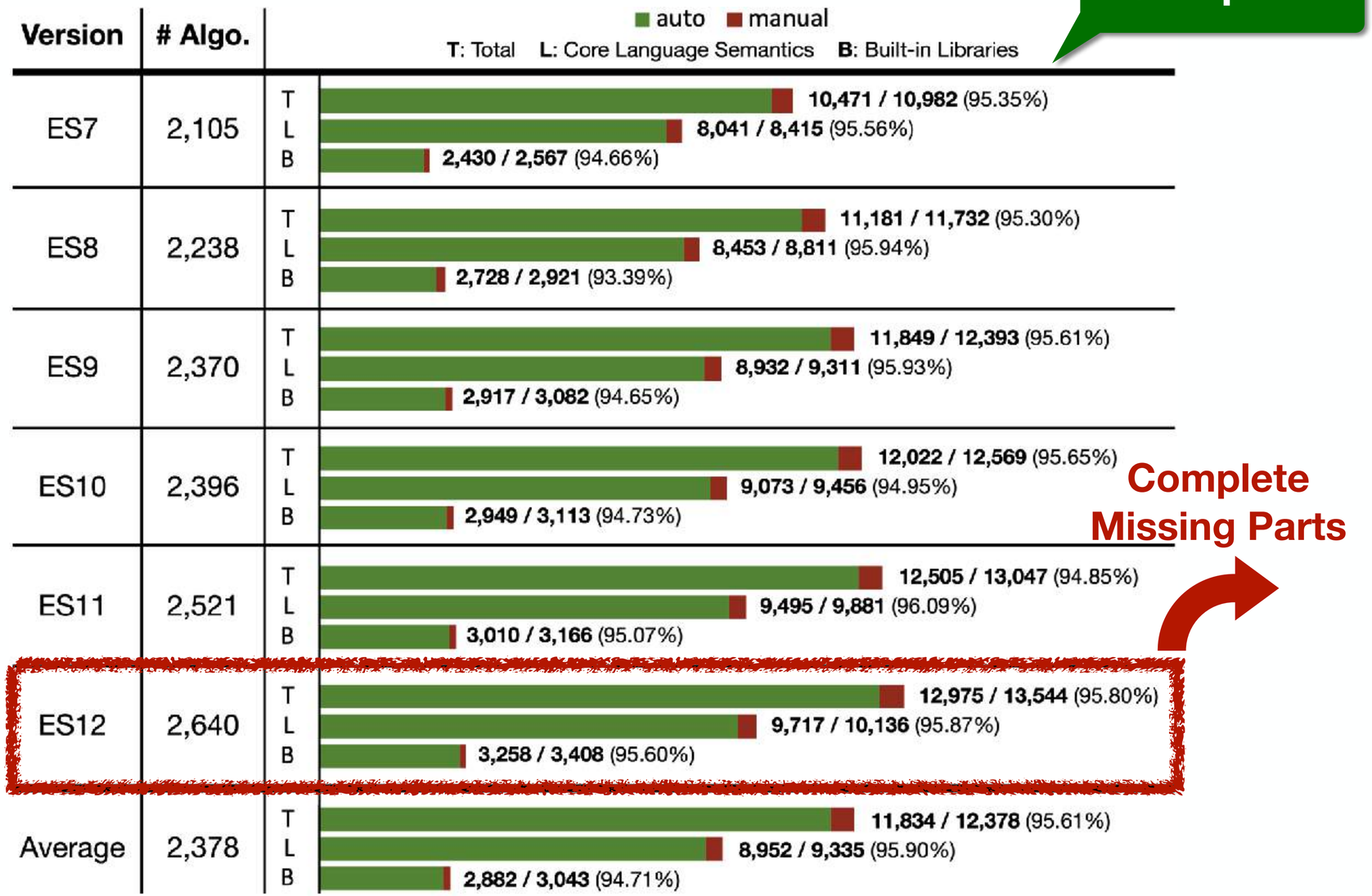
≈ 96% Compiled



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%)

JISET - Evaluation

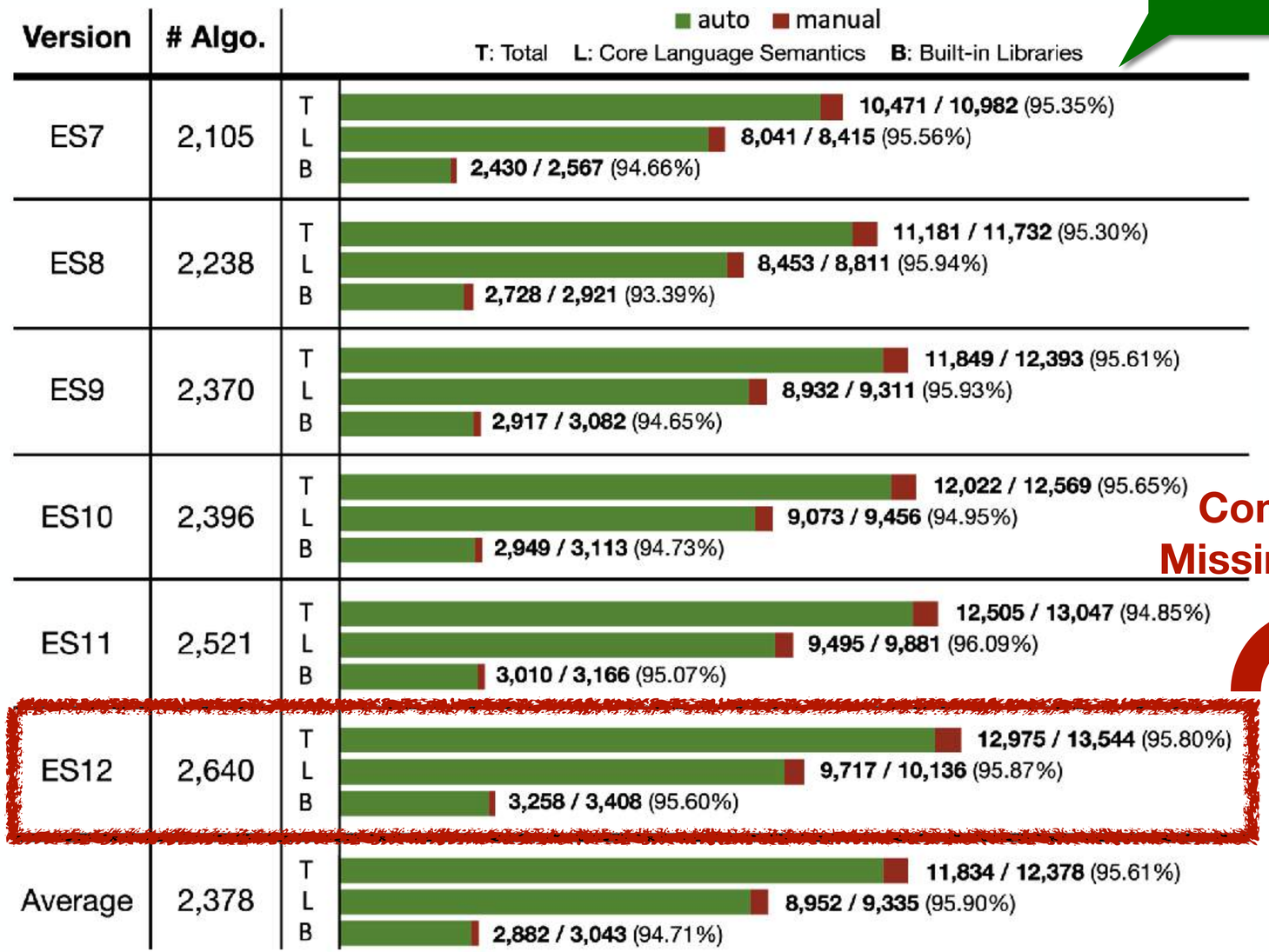
≈ 96% Compiled



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%)

Complete Missing Parts

JISET - Evaluation



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%)

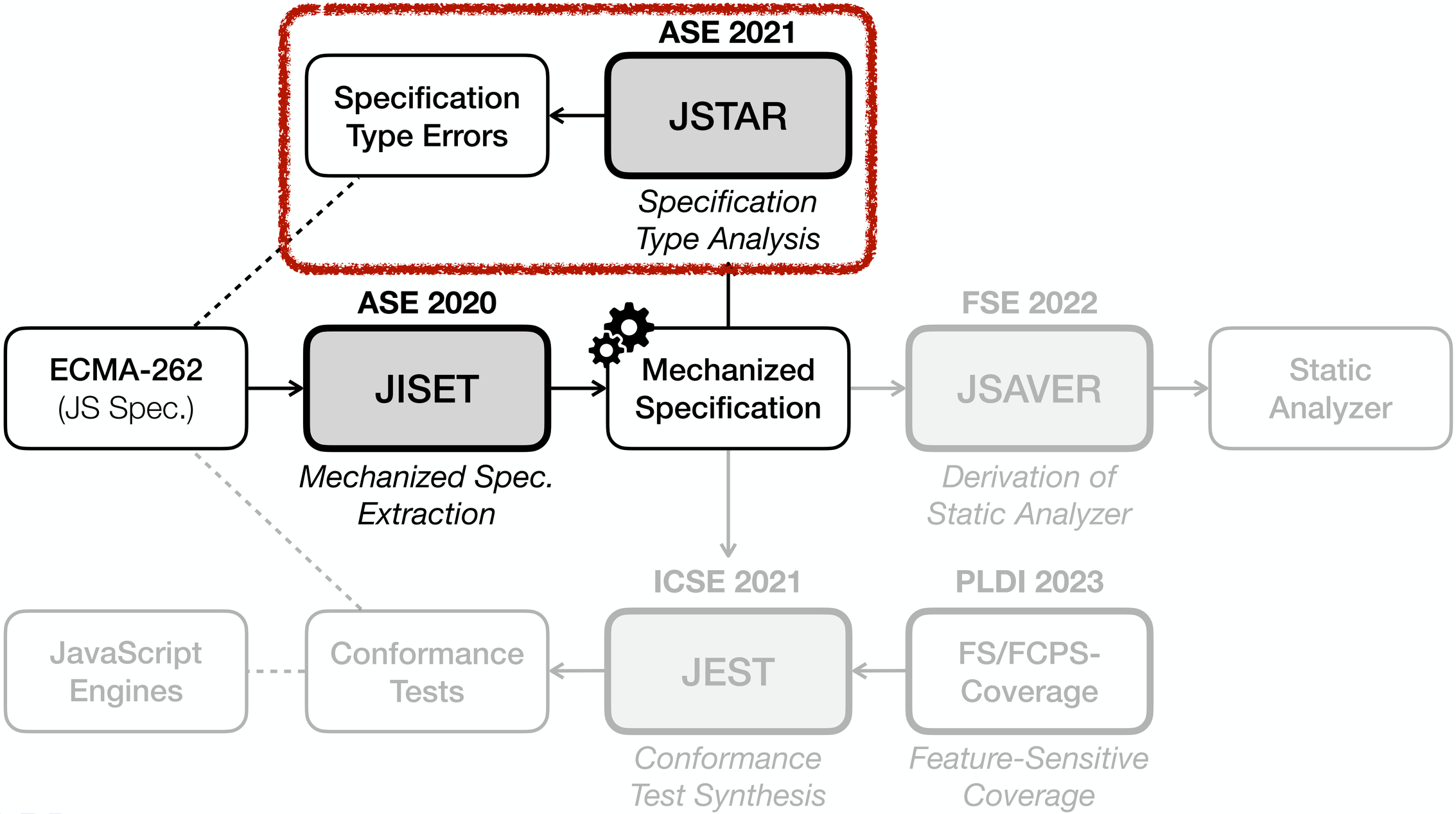
≈ 96% Compiled

Passed All Tests

Complete Missing Parts

- **Test262**
  (Official Conformance Tests)
  - 18,556 applicable tests
- **Parsing tests**
  - Passed all 18,556 tests
- **Evaluation Tests**
  - Passed all 18,556 tests

[ASE 2021] J. Park et al., “JSTAR: JavaScript Specification Type Analyzer using Refinement”



JSTAR - Specification Type Analysis

20.3.2.28 Math.round (*x*)

1. Let *n* be ? ToNumber(*x*).
2. If *n* is an integral Number, return *n*.
3. If $x < 0.5$ and $x > 0$, return **+0**.
4. If $x < 0$ and $x \geq -0.5$, return **-0**.

• • •

https://github.com/tc39/ecma262/tree/575149cfd77aebcf3a129e165bd89e14caafc31c

JSTAR - Specification Type Analysis

`String | Boolean | Number | Object | ...`

20.3.2.28 Math.round (*x*)

1. Let *n* be ? ToNumber(*x*).
2. If *n* is an integral Number, return *n*.
3. If $x < 0.5$ and $x > 0$, return **+0**.
4. If $x < 0$ and $x \geq -0.5$, return **-0**.

. . .

https://github.com/tc39/ecma262/tree/575149cfd77aebcf3a129e165bd89e14caafc31c

JSTAR - Specification Type Analysis

`String | Boolean | Number | Object | ...`

20.3.2.28 Math.round (*x*)

`Number | Exception`

1. Let *n* be ? ToNumber(*x*).
2. If *n* is an integral Number, return *n*.
3. If $x < 0.5$ and $x > 0$, return **+0**.
4. If $x < 0$ and $x \geq -0.5$, return **-0**.

. . .

https://github.com/tc39/ecma262/tree/575149cfd77aebcf3a129e165bd89e14caafc31c

JSTAR - Specification Type Analysis

`String | Boolean | Number | Object | ...`

20.3.2.28 Math.round (*x*)

`Number | Exception`

1. Let *n* be ? ToNumber(*x*).
2. If *n* is an integral Number, return *n*.
3. If $x < 0.5$ and $x > 0$, return **+0**.
4. If $x < 0$ and $x \geq -0.5$, return **-0**.

. . .

https://github.com/tc39/ecma262/tree/575149cfd77aebcf3a129e165bd89e14caafc31c

JSTAR - Specification Type Analysis

String | Boolean | Number | Object | ...

20.3.2.28 Math.round (*x*)

Number

Number | Exception

1. Let *n* be ? ToNumber(*x*).
2. If *n* is an integral Number, return *n*.
3. If $x < 0.5$ and $x > 0$, return **+0**.
4. If $x < 0$ and $x \geq -0.5$, return **-0**.

. . .

https://github.com/tc39/ecma262/tree/575149cfd77aebcf3a129e165bd89e14caafc31c

JSTAR - Specification Type Analysis

`String | Boolean | Number | Object | ...`

20.3.2.28 Math.round (*x*)

`Number` `Number | Exception`

1. Let *n* be ? ToNumber(*x*).
2. If *n* is an integral Number, return *n*.
3. If $x < 0.5$ and $x > 0$, return **+0**.
4. If $x < 0$ and $x \geq -0.5$, return **-0**.

. . .

`Type Error: \`<\`, \`>\`, and \`>=\` are numeric operators`

https://github.com/tc39/ecma262/tree/575149cfd77aebcf3a129e165bd89e14caafc31c

JSTAR - Specification Type Analysis

`String | Boolean | Number | Object | ...`

20.3.2.28 Math.round (*x*)

`Number` `Number | Exception`

1. Let *n* be ? ToNumber(*x*).
2. If *n* is an integral Number, return *n*.
3. If $x < 0.5$ and $x > 0$ return **+0**.
4. If $x < 0$ and $x \geq -0.5$ return **-0**.

. . .

```
Type Error:
`<`, `>`, and `>=`
are numeric operators
```

```
Math.round(true)  = ???
Math.round(false) = ???
```

https://github.com/tc39/ecma262/tree/575149cfd77aebcf3a129e165bd89e14caafc31c

JSTAR - Specification Type Analysis

String | Boolean | Number | Object | ...

20.3.2.28 Math.round (*x*)

Number | Number | Exception

1. Let *n* be ? ToNumber(*x*).
2. If *n* is an integral Number, return *n*.
3. If $x < 0.5$ and $x > 0$ return **+0**.
4. If $x < 0$ and $x \geq -0.5$ return **-0**.

. . .

Type Error:
`<`, `>`, and `>=`
are numeric operators

```
Math.round(true)  = ???
Math.round(false) = ???
```

3. If $n < 0.5$ and $n > 0$, return **+0**.
4. If $n < 0$ and $n \geq -0.5$, return **-0**.

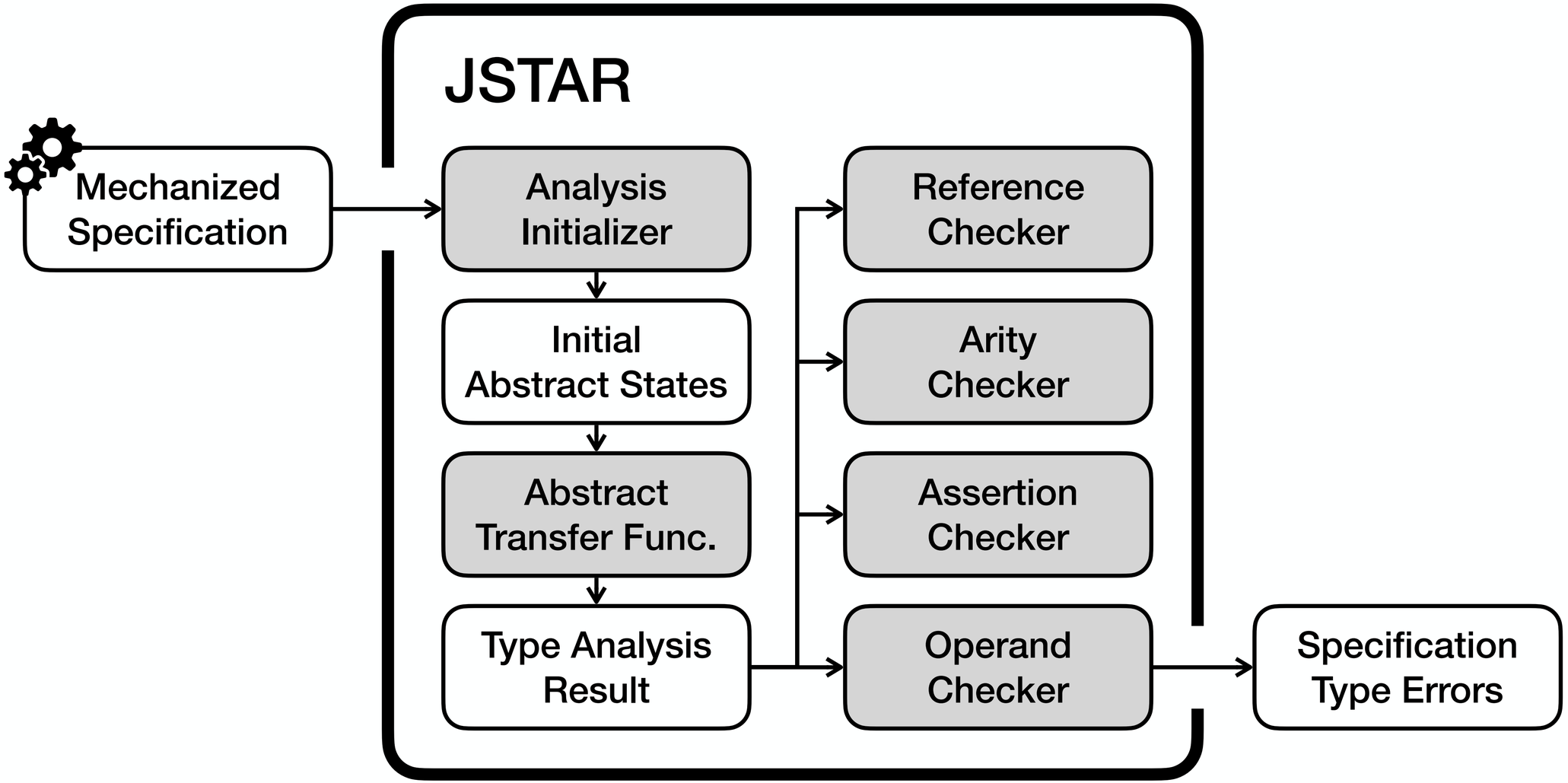
Fixed

```
Math.round(true)  =  0
Math.round(false) =  1
```

https://github.com/tc39/ecma262/tree/575149cfd77aebcf3a129e165bd89e14caafc31c

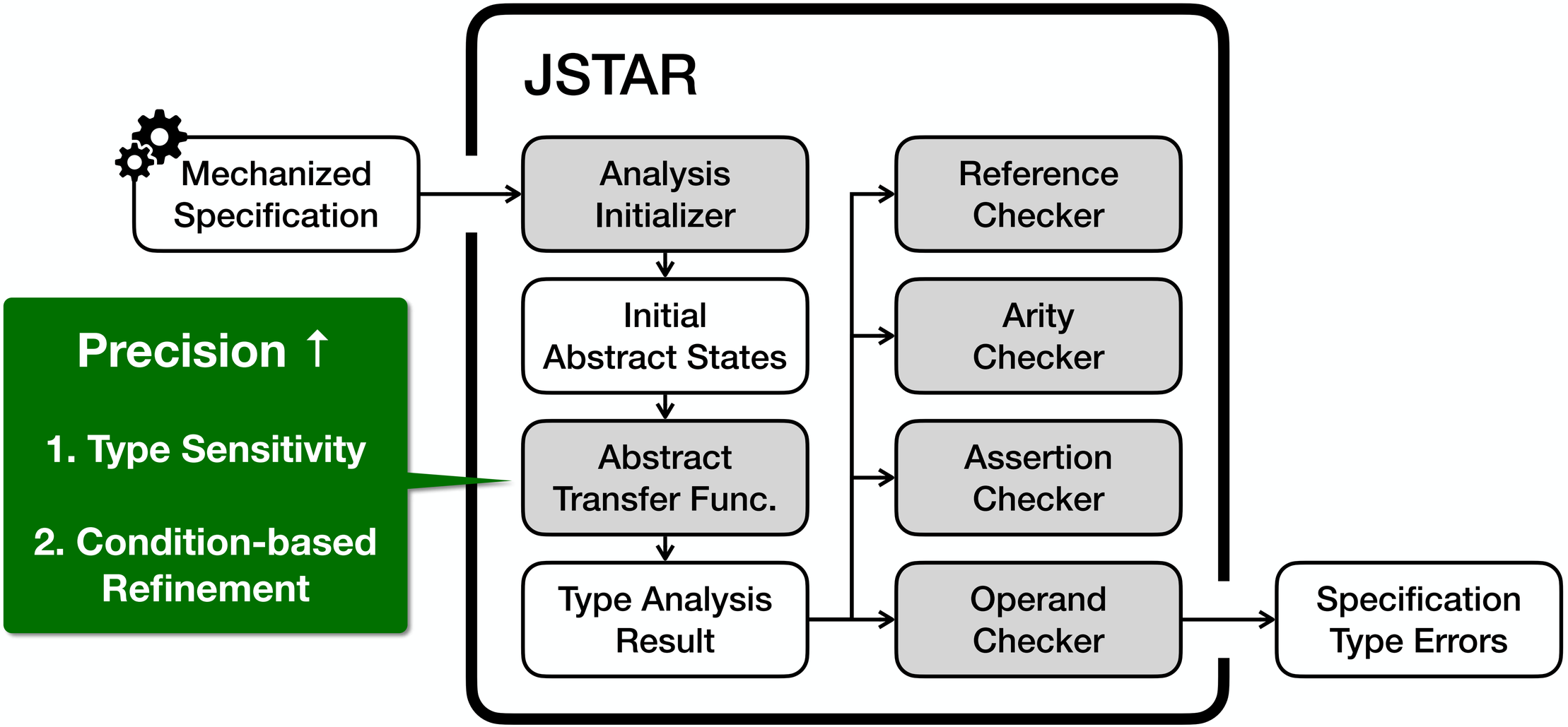
JSTAR

(**J**avaScript **S**pecification **T**ype **A**nalyzer using **R**efinement)

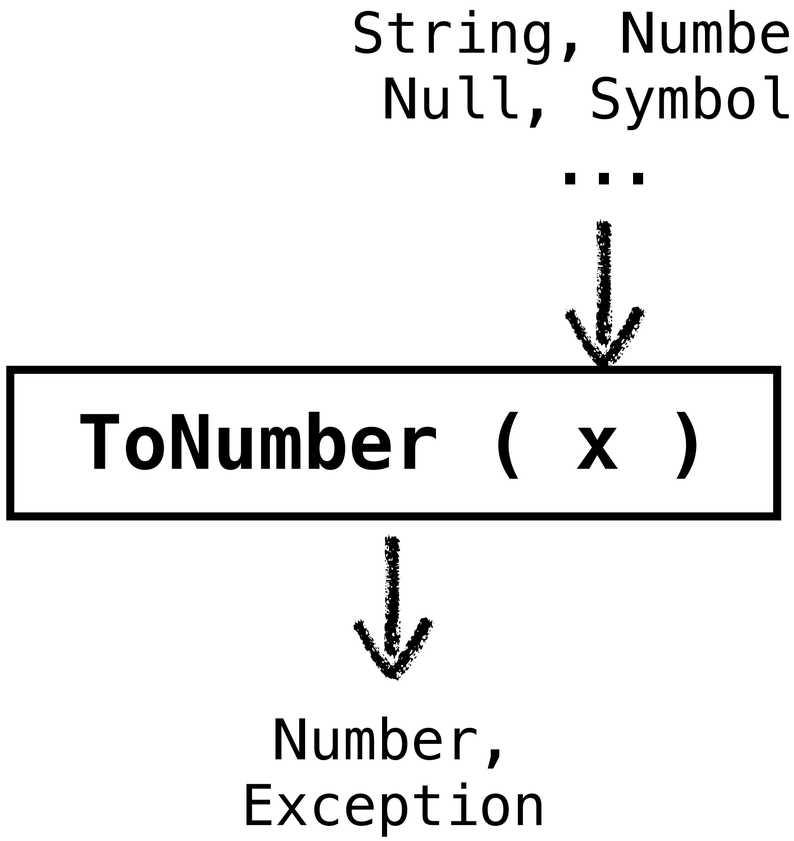


JSTAR

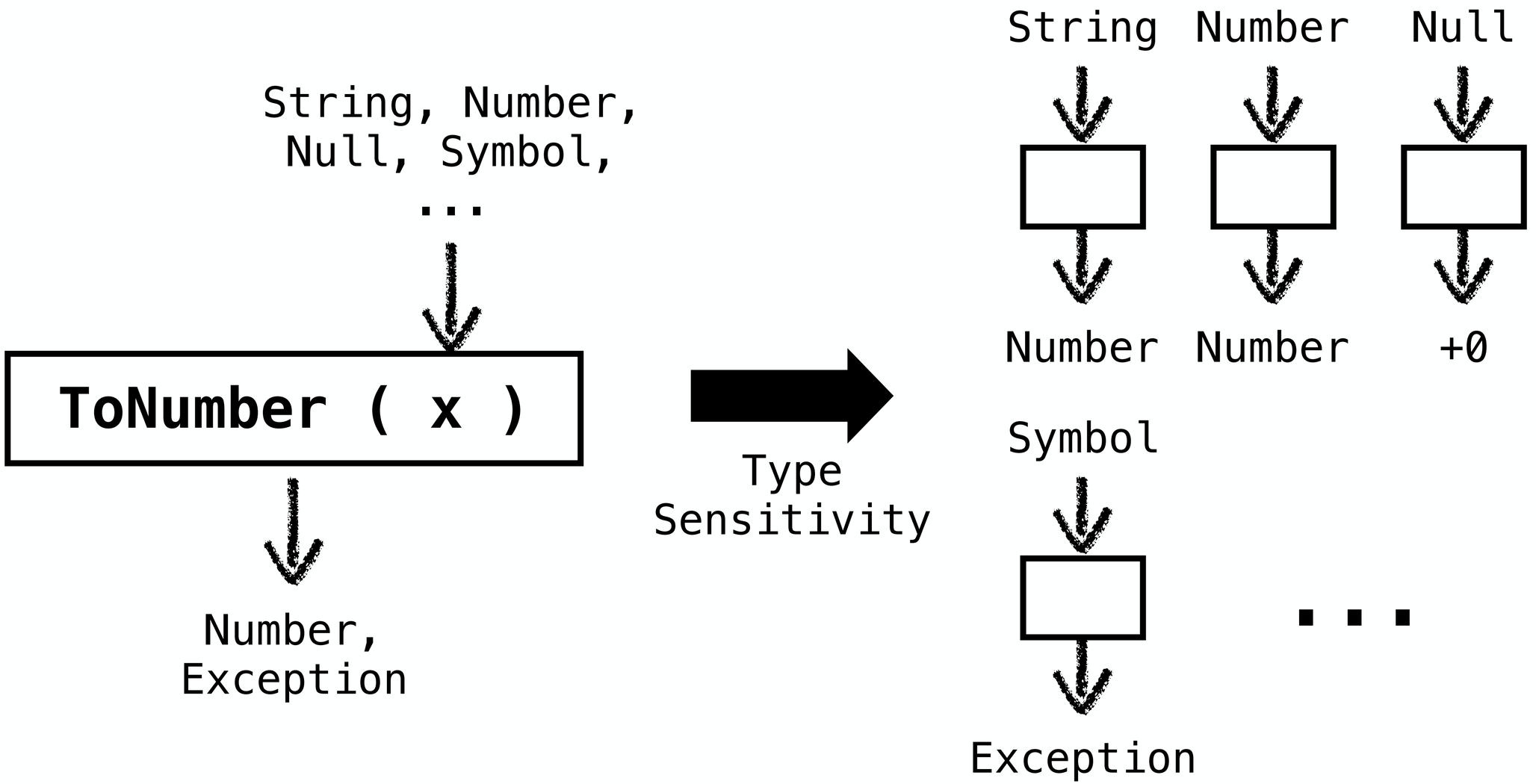
(**J**avaScript **S**pecification **T**ype **A**nalyzer using **R**efinement)



JSTAR - Type Sensitivity



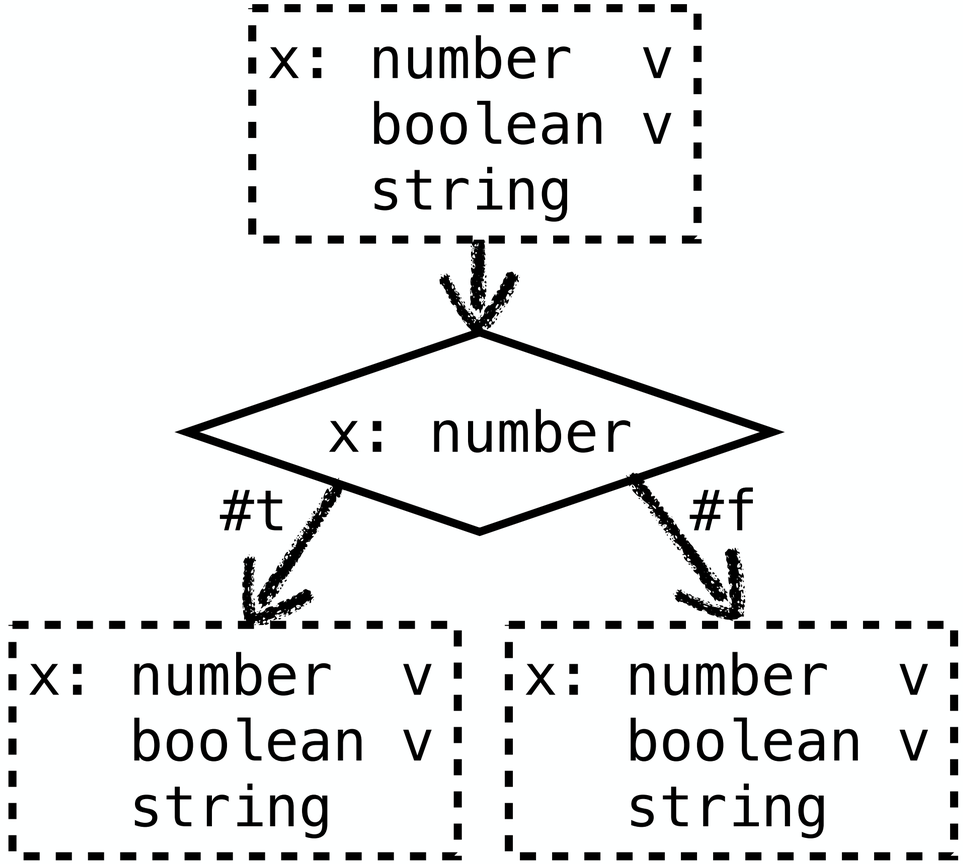
JSTAR - Type Sensitivity



JSTAR - Condition-based Refinement

$$
\begin{aligned}
\texttt{refine}(!e, b)(\sigma^\sharp) &= \texttt{refine}(e, \neg b)(\sigma^\sharp) \\
\texttt{refine}(e_0 \texttt{ || } e_1, b)(\sigma^\sharp) &= \begin{cases} \sigma_0^\sharp \sqcup \sigma_1^\sharp & \text{if } b \\ \sigma_0^\sharp \sqcap \sigma_1^\sharp & \text{if } \neg b \end{cases} \\
\texttt{refine}(e_0 \texttt{ \&\& } e_1, b)(\sigma^\sharp) &= \begin{cases} \sigma_0^\sharp \sqcap \sigma_1^\sharp & \text{if } b \\ \sigma_0^\sharp \sqcup \sigma_1^\sharp & \text{if } \neg b \end{cases} \\
\texttt{refine}(\texttt{x.Type} == c_{\texttt{normal}}, \texttt{\#t})(\sigma^\sharp) &= \sigma^\sharp[\texttt{x} \mapsto \tau_{\texttt{x}}^\sharp \sqcap \texttt{normal}(\mathbb{T})] \\
\texttt{refine}(\texttt{x.Type} == c_{\texttt{normal}}, \texttt{\#f})(\sigma^\sharp) &= \sigma^\sharp[\texttt{x} \mapsto \tau_{\texttt{x}}^\sharp \sqcap \{\texttt{abrupt}\}] \\
\texttt{refine}(\texttt{x} == e, \texttt{\#t})(\sigma^\sharp) &= \sigma^\sharp[\texttt{x} \mapsto \tau_{\texttt{x}}^\sharp \sqcap \tau_e^\sharp] \\
\texttt{refine}(\texttt{x} == e, \texttt{\#f})(\sigma^\sharp) &= \sigma^\sharp[\texttt{x} \mapsto \tau_{\texttt{x}}^\sharp \setminus \lfloor \tau_e^\sharp \rfloor] \\
\texttt{refine}(\texttt{x} : \tau, \texttt{\#t})(\sigma^\sharp) &= \sigma^\sharp[\texttt{x} \mapsto \tau_{\texttt{x}}^\sharp \sqcap \{\tau\}] \\
\texttt{refine}(\texttt{x} : \tau, \texttt{\#f})(\sigma^\sharp) &= \sigma^\sharp[\texttt{x} \mapsto \tau_{\texttt{x}}^\sharp \setminus \{\tau' \mid \tau' <: \tau\}] \\
\texttt{refine}(e, b)(\sigma^\sharp) &= \sigma^\sharp
\end{aligned}
$$

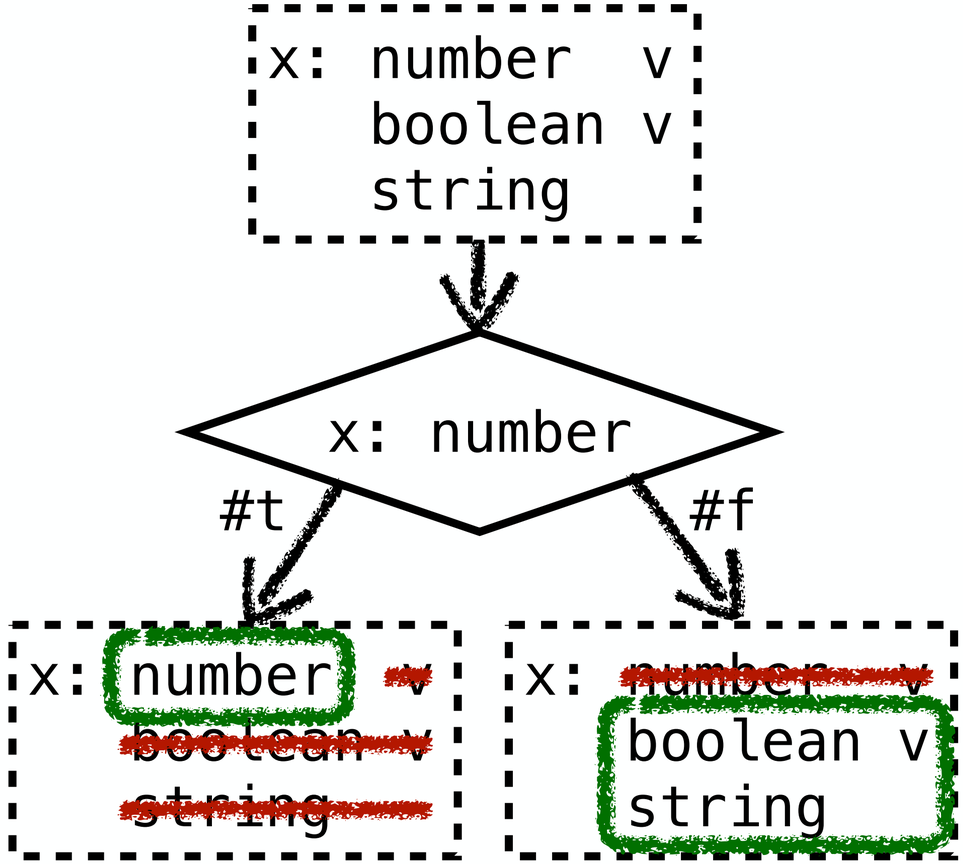
where $\sigma_j^\sharp = \texttt{refine}(e_j, b)(\sigma^\sharp)$ for $j = 0, 1$, $\tau_e^\sharp = [\![e]\!]_e^\sharp(\sigma^\sharp)$, and $\lfloor \tau^\sharp \rfloor$ returns $\{\tau\}$ if $\tau^\sharp$ denotes a singleton type τ, or returns $\varnothing$, otherwise.



JSTAR - Condition-based Refinement

$$
\begin{aligned}
\texttt{refine}(!e, b)(\sigma^\sharp) &= \texttt{refine}(e, \neg b)(\sigma^\sharp) \\
\texttt{refine}(e_0 \texttt{ || } e_1, b)(\sigma^\sharp) &= \begin{cases} \sigma_0^\sharp \sqcup \sigma_1^\sharp & \text{if } b \\ \sigma_0^\sharp \sqcap \sigma_1^\sharp & \text{if } \neg b \end{cases} \\
\texttt{refine}(e_0 \texttt{ \&\& } e_1, b)(\sigma^\sharp) &= \begin{cases} \sigma_0^\sharp \sqcap \sigma_1^\sharp & \text{if } b \\ \sigma_0^\sharp \sqcup \sigma_1^\sharp & \text{if } \neg b \end{cases} \\
\texttt{refine}(\texttt{x.Type} == c_{\texttt{normal}}, \texttt{\#t})(\sigma^\sharp) &= \sigma^\sharp[\texttt{x} \mapsto \tau_{\texttt{x}}^\sharp \sqcap \texttt{normal}(\mathbb{T})] \\
\texttt{refine}(\texttt{x.Type} == c_{\texttt{normal}}, \texttt{\#f})(\sigma^\sharp) &= \sigma^\sharp[\texttt{x} \mapsto \tau_{\texttt{x}}^\sharp \sqcap \{\texttt{abrupt}\}] \\
\texttt{refine}(\texttt{x} == e, \texttt{\#t})(\sigma^\sharp) &= \sigma^\sharp[\texttt{x} \mapsto \tau_{\texttt{x}}^\sharp \sqcap \tau_e^\sharp] \\
\texttt{refine}(\texttt{x} == e, \texttt{\#f})(\sigma^\sharp) &= \sigma^\sharp[\texttt{x} \mapsto \tau_{\texttt{x}}^\sharp \setminus \lfloor \tau_e^\sharp \rfloor] \\
\texttt{refine}(\texttt{x} : \tau, \texttt{\#t})(\sigma^\sharp) &= \sigma^\sharp[\texttt{x} \mapsto \tau_{\texttt{x}}^\sharp \sqcap \{\tau\}] \\
\texttt{refine}(\texttt{x} : \tau, \texttt{\#f})(\sigma^\sharp) &= \sigma^\sharp[\texttt{x} \mapsto \tau_{\texttt{x}}^\sharp \setminus \{\tau' \mid \tau' <: \tau\}] \\
\texttt{refine}(e, b)(\sigma^\sharp) &= \sigma^\sharp
\end{aligned}
$$

where $\sigma_j^\sharp = \texttt{refine}(e_j, b)(\sigma^\sharp)$ for $j = 0, 1$, $\tau_e^\sharp = [\![e]\!]_e^\sharp(\sigma^\sharp)$, and $\lfloor \tau^\sharp \rfloor$ returns $\{\tau\}$ if $\tau^\sharp$ denotes a singleton type τ, or returns $\varnothing$, otherwise.



JSTAR - Evaluation

- Type analysis on **864 versions** of ECMA-262 in 3 years

59.2% Precision

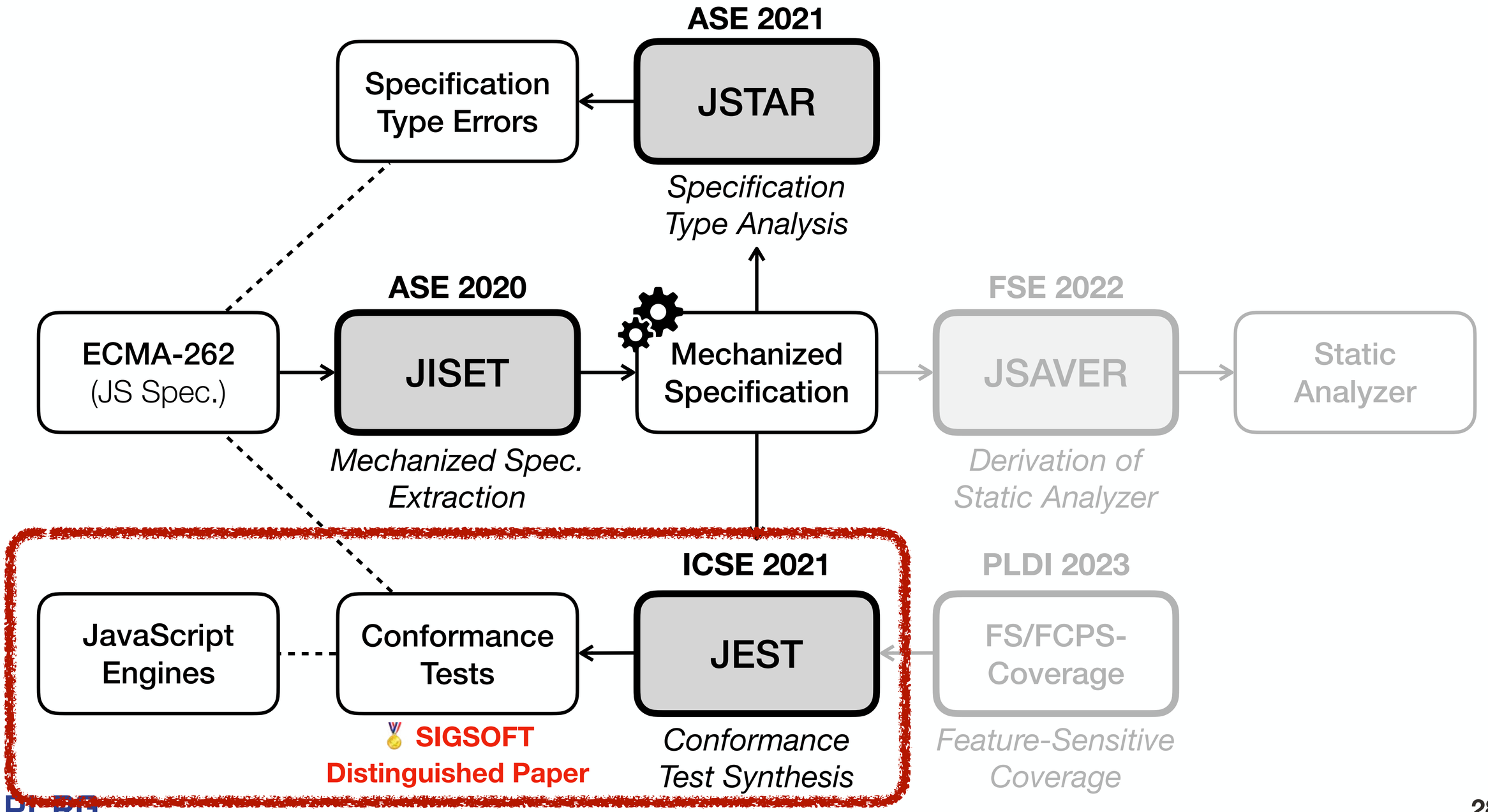
93 Errors Detected

Checker	Bug Kind	Precision = (# True Bugs) / (# Detected Bugs)					
		no-refine		refine		Δ	
Reference	UnknownVar	62 / 106	17 / 60	63 / 78	17 / 31	+1 / -28	/ -29
	DuplicatedVar		45 / 46		46 / 47		+1 / +1
Arity	MissingParam	4 / 4	4 / 4	4 / 4	4 / 4	/	/
Assertion	Assertion	4 / 56	4 / 56	4 / 31	4 / 31	/ -25	/ -25
Operand	NoNumber	22 / 113	2 / 65	22 / 44	2 / 6	/ -69	/ -59
	Abrupt		20 / 48		20 / 38		/ -10
Total		92 / 279 (33.0%)		93 / 157 (59.2%)		+1 / -122 (+26.3%)	

Name	Feature	#	Checker	Created	Life Span
ES12-1	Switch	3	Reference	2015-09-22	1,996 days
ES12-2	Try	3	Reference	2015-09-22	1,996 days
ES12-3	Arguments	1	Reference	2015-09-22	1,996 days
ES12-4	Array	2	Reference	2015-09-22	1,996 days
ES12-5	Async	1	Reference	2015-09-22	1,996 days
ES12-6	Class	1	Reference	2015-09-22	1,996 days
ES12-7	Branch	1	Reference	2015-09-22	1,996 days
ES12-8	Arguments	2	Operand	2015-12-16	1,910 days

14 New Bugs In ES2021

[ICSE 2021] J. Park et al., “JEST: N +1-version Differential Testing of Both JavaScript Engines and Specification”



Conformance of JavaScript Engines

ECMA-262
(JavaScript Spec.)

Conformance

GraalVM

QuickJS

moddable

JavaScript Engines

Conformance of JavaScript Engines

ECMA-262
(JavaScript Spec.)

How?

Conformance

GraalVM

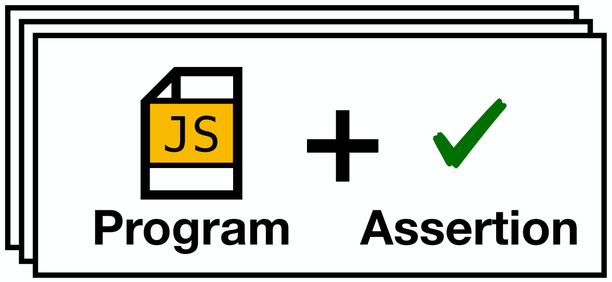
QuickJS

moddable

JavaScript Engines

Conformance of JavaScript Engines

ECMA-262
(JavaScript Spec.)



Conformance Tests

Conformance

Test262
(Official Test Suite)

GraalVM

QuickJS

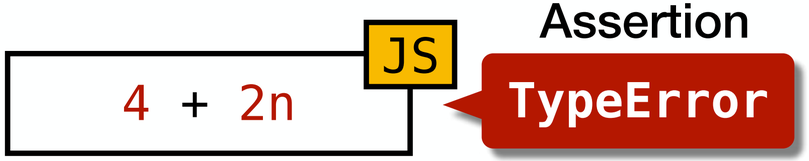
moddable

JavaScript Engines

Conformance of JavaScript Engines

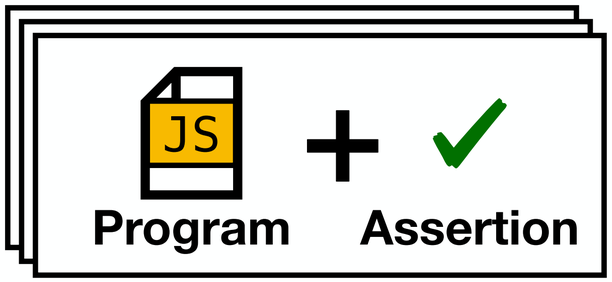
ECMA-262
(JavaScript Spec.)

Example:



Assertion

TypeError



Conformance Tests

Conformance

Test262
(Official Test Suite)

GraalVM

QuickJS

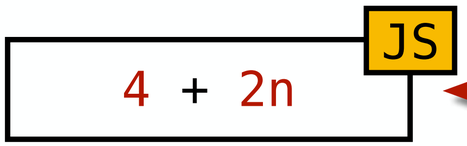
moddable

JavaScript Engines

Problem - Manual Approach

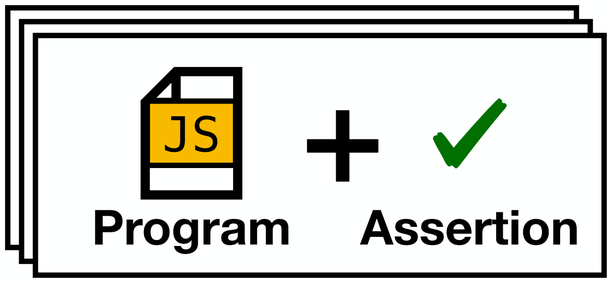
ECMA-262
(JavaScript Spec.)

Example:



Assertion

`TypeError`



Conformance Tests

Conformance

Test262
(Official Test Suite)

`Manual`

GraalVM

`QuickJS`

moddable

JavaScript Engines

N+1-version Differential Testing

ECMA-262
(JavaScript Spec.)

Synthesize

`Test`

test
test
test
test

GraalVM

`QuickJS`

moddable

JavaScript Engines

N+1-version Differential Testing

ECMA-262
(JavaScript Spec.)

Synthesize

`Test`

test
test
test
test

JavaScript Engines

N+1-version Differential Testing

ECMA-262
(JavaScript Spec.)

Synthesize → `Test` → **test** **test** **test** **test**

JavaScript Engines

An engine bug in

N+1-version Differential Testing

ECMA-262
(JavaScript Spec.)

Synthesize

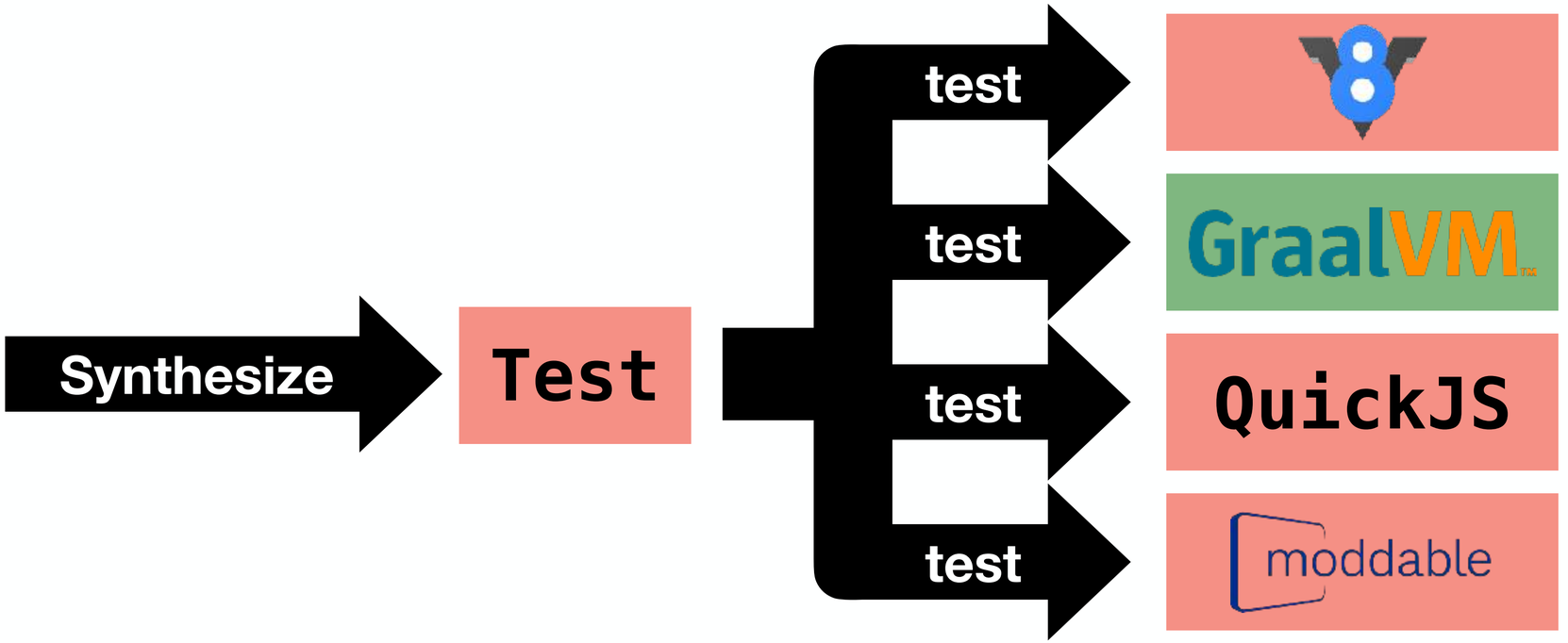
`Test`

test
test
test
test

JavaScript Engines

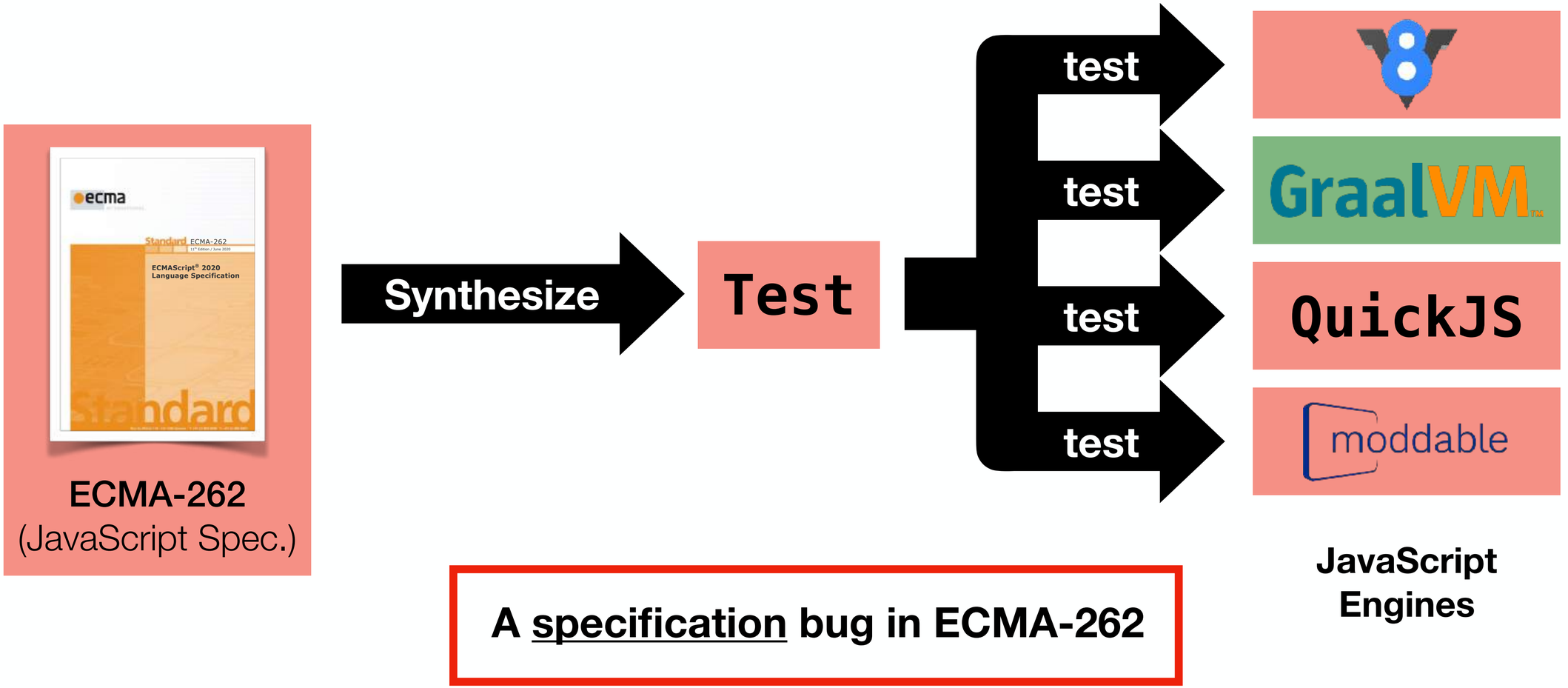
N+1-version Differential Testing

ECMA-262
(JavaScript Spec.)

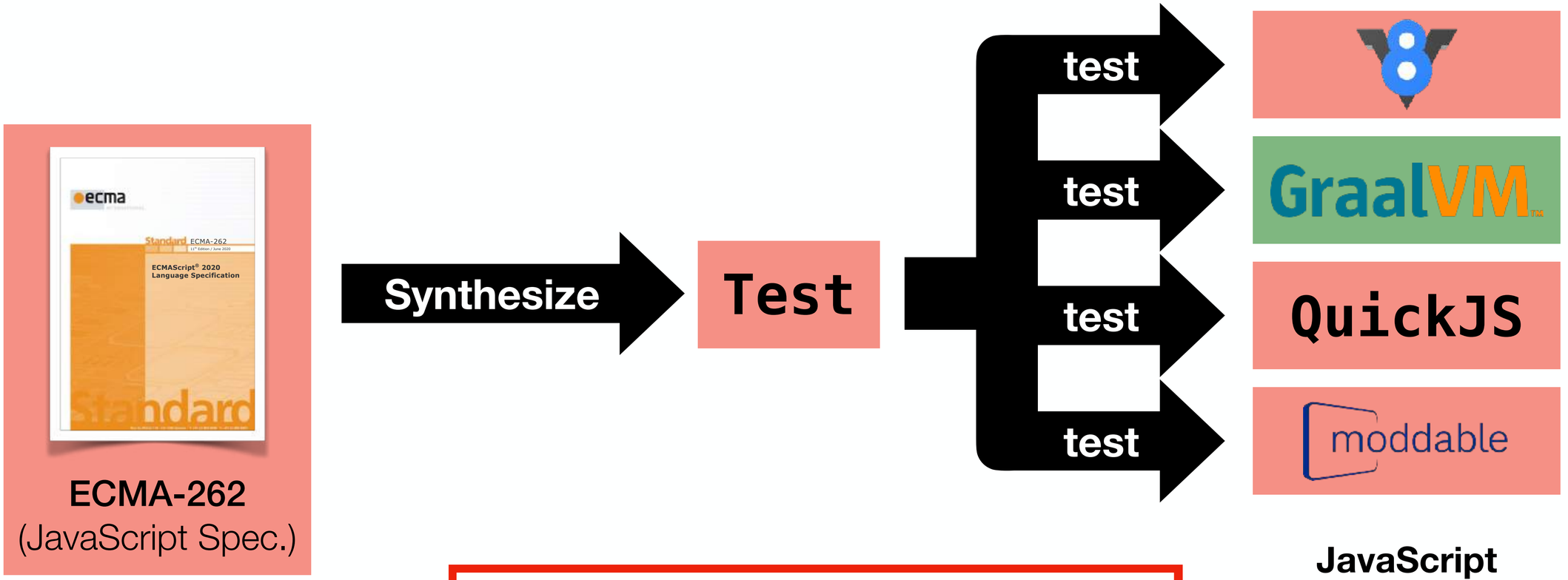


JavaScript Engines

N+1-version Differential Testing



N+1-version Differential Testing

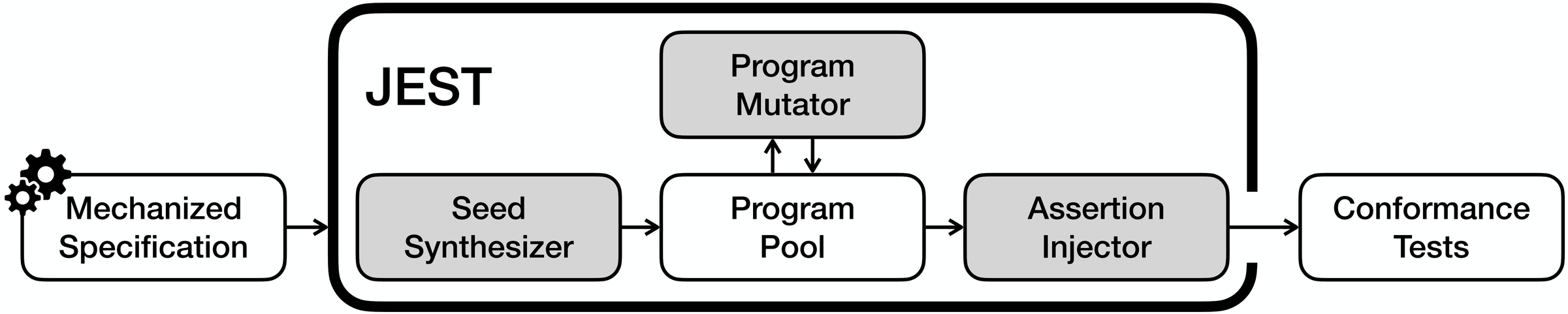


A specification bug in ECMA-262

An engine bug in GraalVM

JEST

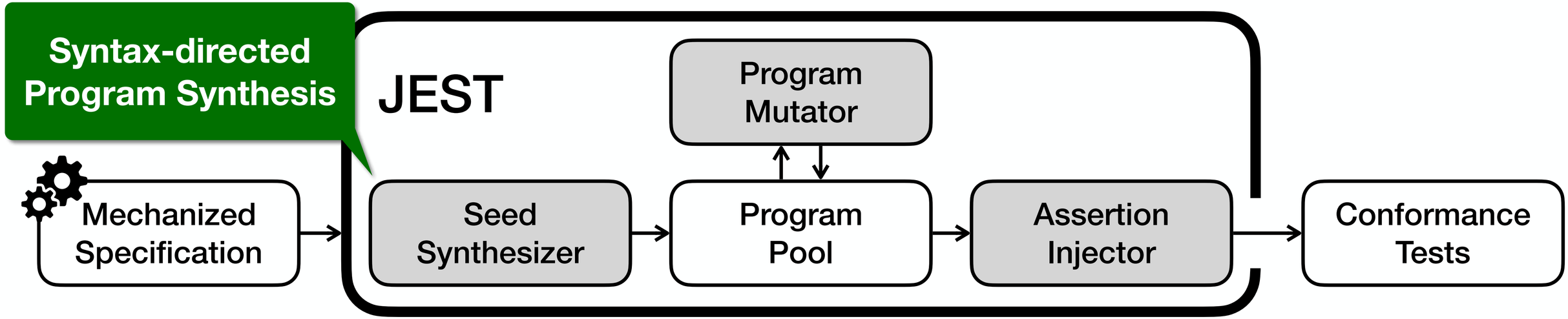
(**J**avaScript **E**ngines and **S**pecification **T**ester)



Program Pool

JEST

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

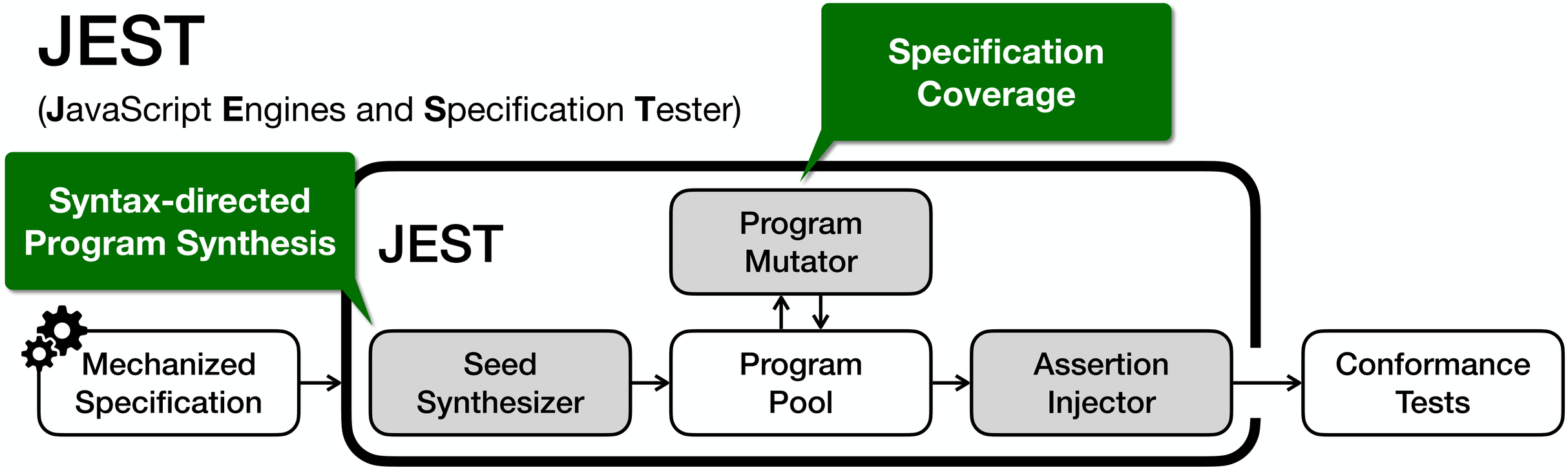


Program Pool

```
let x = 42;
```

JEST

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



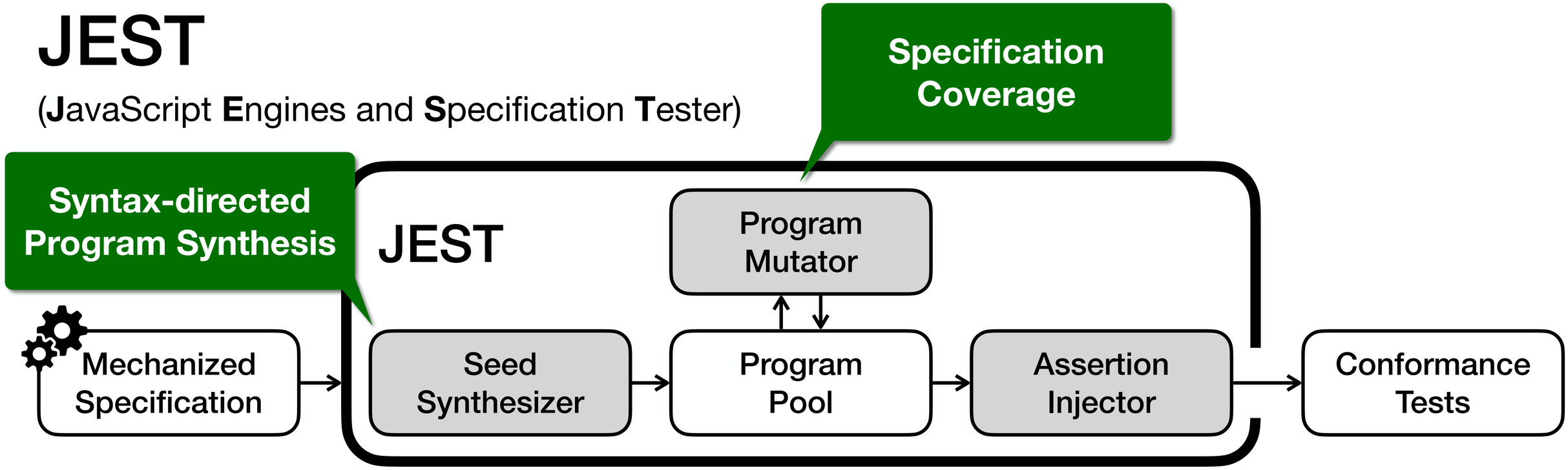
Program Pool

```
let x = 1 + 2;
```

```
let x = 42;
```

JEST

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



Program Pool

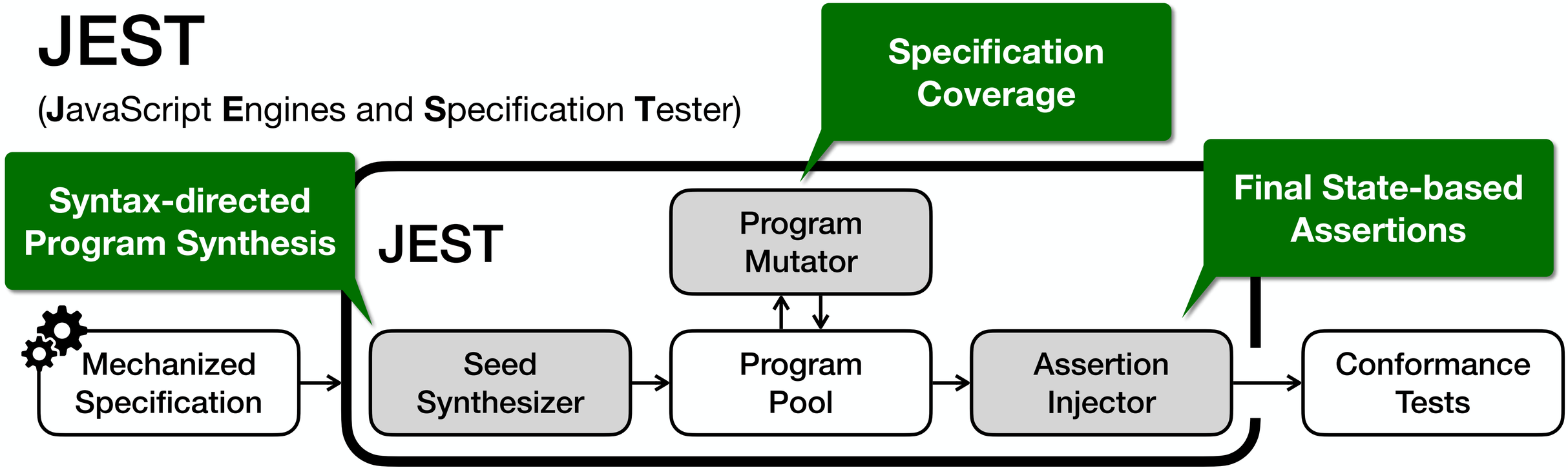
```
let x = 1 + 2;
```

```
let x = 42;
```

```
let x = ![];
```

JEST

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



Program Pool

```
let x = 1 + 2;
assert(x == 3);
```

```
let x = 42;
assert(x == 42);
```

```
let x = ![];
assert(x == false);
```

JEST - Specification Coverage

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

...

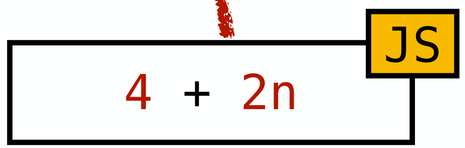
3. Let *lnum* be ? ToNumeric(*lval*).
4. Let *rnum* be ? ToNumeric(*rval*).
5. If Type(*lnum*) is not Type(*rnum*), throw a **TypeError** exception.
6. If *lnum* is a BigInt, then
   ...
7. Else,
   ...

JEST - Specification Coverage

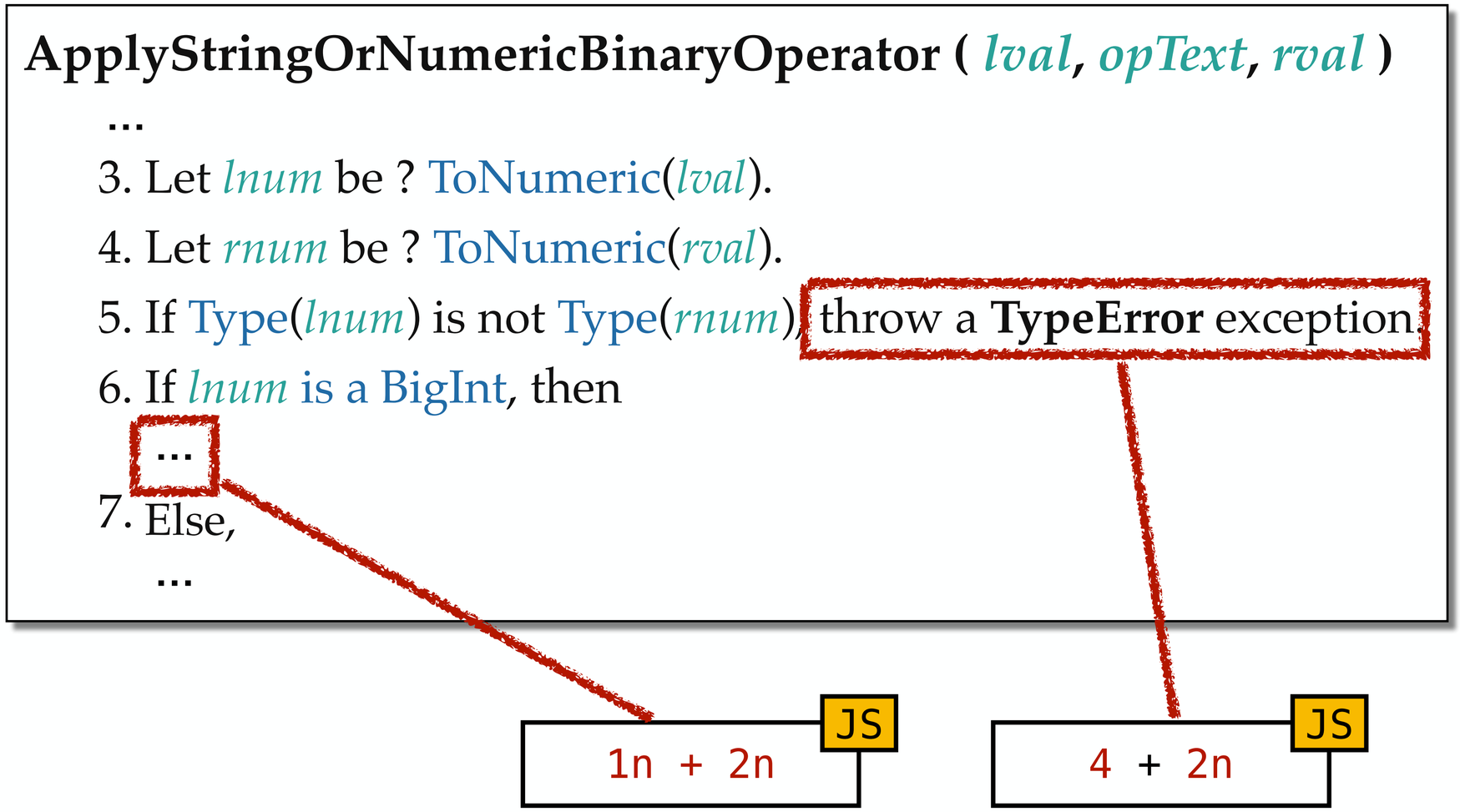
ApplyStringOrNumericBinaryOperator (*lval*, *opText*, *rval*)

...

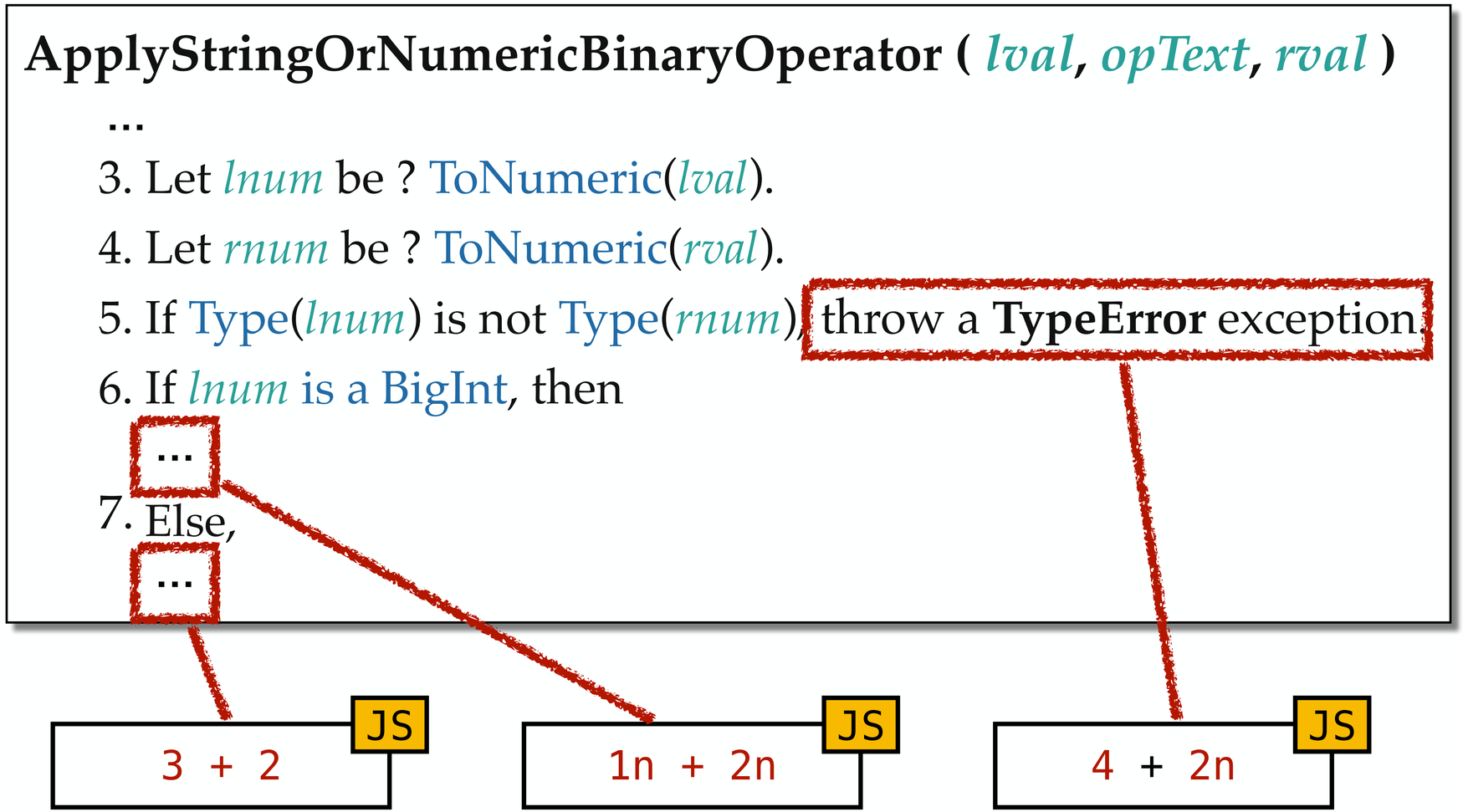
3. Let *lnum* be ? ToNumeric(*lval*).
4. Let *rnum* be ? ToNumeric(*rval*).
5. If Type(*lnum*) is not Type(*rnum*), throw a **TypeError** exception.
6. If *lnum* is a BigInt, then
   ...
7. Else,
   ...



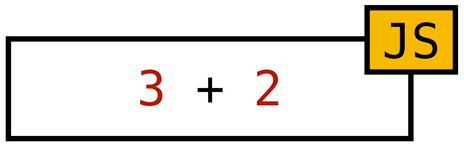
JEST - Specification Coverage

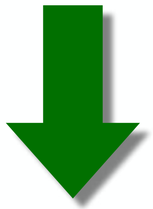


JEST - Specification Coverage

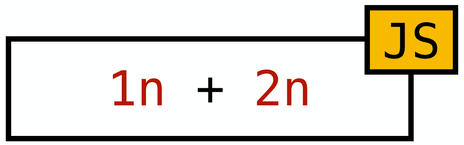


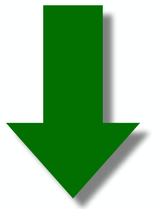
JEST - Final State-based Assertion Injection



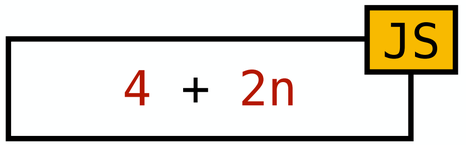


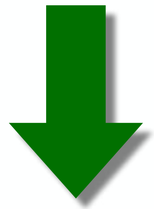
```
  var x = 3 + 2;
+ $assert.equal(x, 5);
```





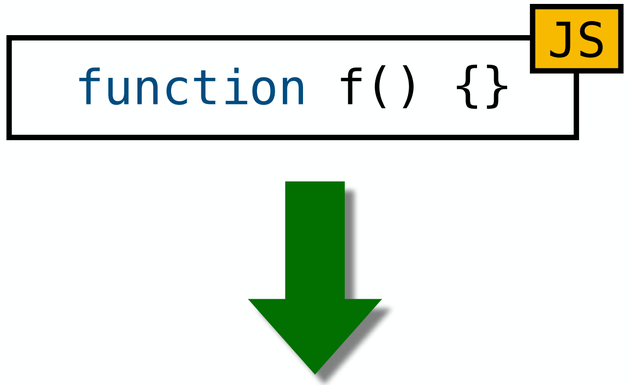
```
  var x = 1n + 2n;
+ $assert.equal(x, 3n);
```





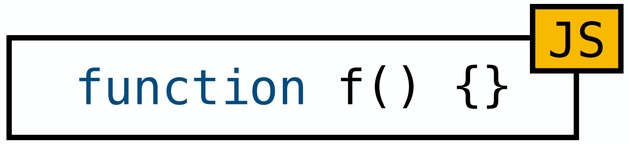
```
  var x = 4 + 2n;
+ // [THROW] TypeError
```

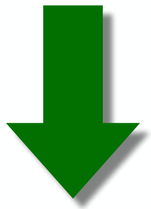
JEST - Final State-based Assertion Injection




```
  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 - Final State-based Assertion Injection

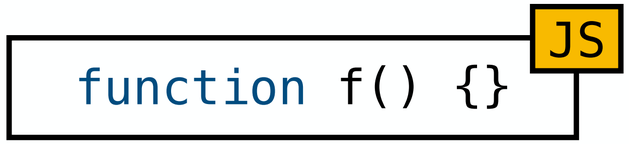


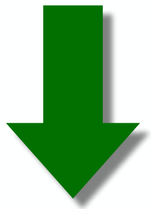


Prototype Chain

```
  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);

+ ...
```

JEST - Final State-based Assertion Injection



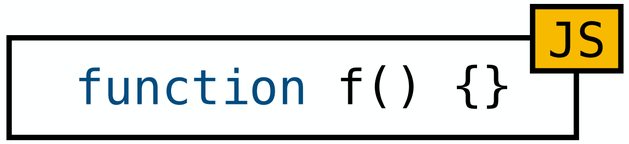


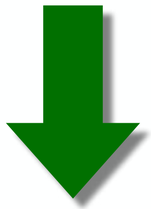
Prototype Chain

```
  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 - Final State-based Assertion Injection





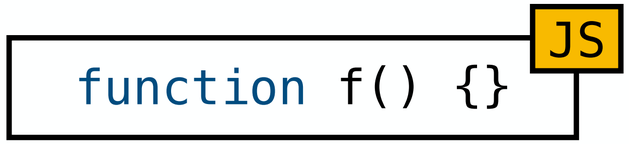
```
  function f() {}

+ $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);

+ ...
```

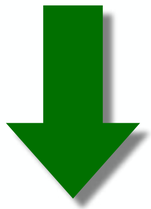
Prototype Chain

Property Descriptor

Property Order

JEST - Final State-based Assertion Injection





```
  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 In Engines

TABLE II: The number of engine bugs detected by JEST

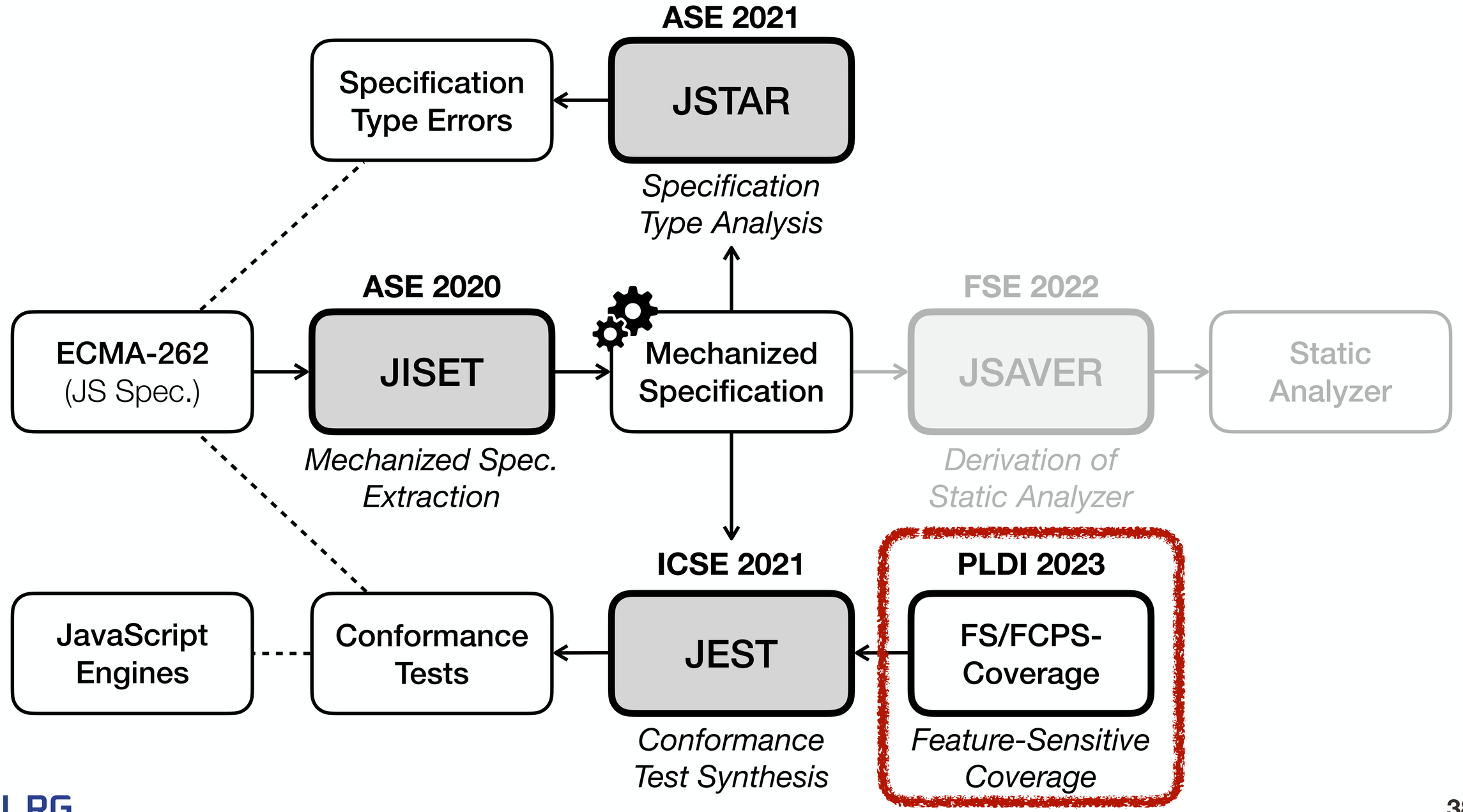
Engines	Exc	Abort	Var	Obj	Desc	Key	In	Total
V8	0	0	0	0	0	2	0	2
GraalVM	6	0	0	0	2	8	0	16
QuickJS	3	0	1	0	0	2	0	6
Moddable XS	12	0	0	0	3	5	0	20
Total	21	0	1	0	5	17	0	44

27 Bugs In Spec.

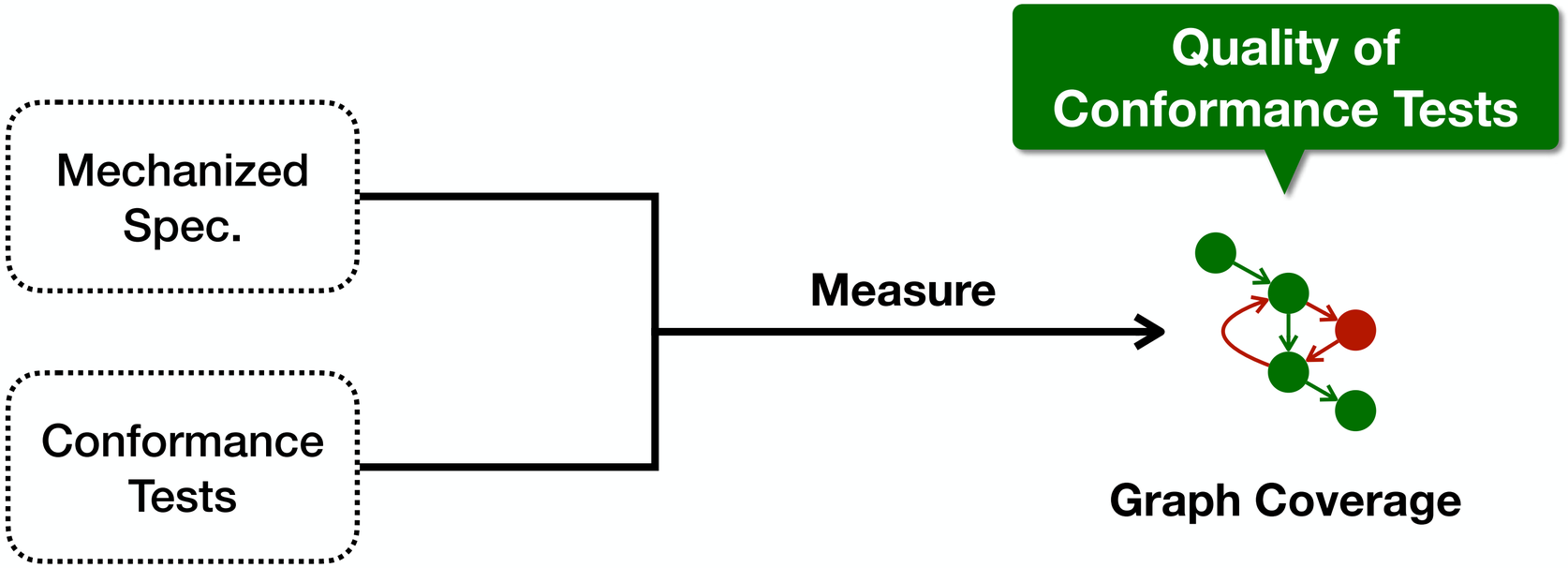
TABLE III: Specification bugs in ECMAScript 2020 (ES11) detected by JEST

Name	Feature	#	Assertion	Known	Created	Resolved	Existed
ES11-1	Function	12	Key	O	2019-02-07	2020-04-11	429 days
ES11-2	Function	8	Key	O	2015-06-01	2020-04-11	1,776 days
ES11-3	Loop	1	Exc	O	2017-10-17	2020-04-30	926 days
ES11-4	Expression	4	Abort	O	2019-09-27	2020-04-23	209 days
ES11-5	Expression	1	Exc	O	2015-06-01	2020-04-28	1,793 days
ES11-6	Object	1	Exc	X	2019-02-07	2020-11-05	637 days

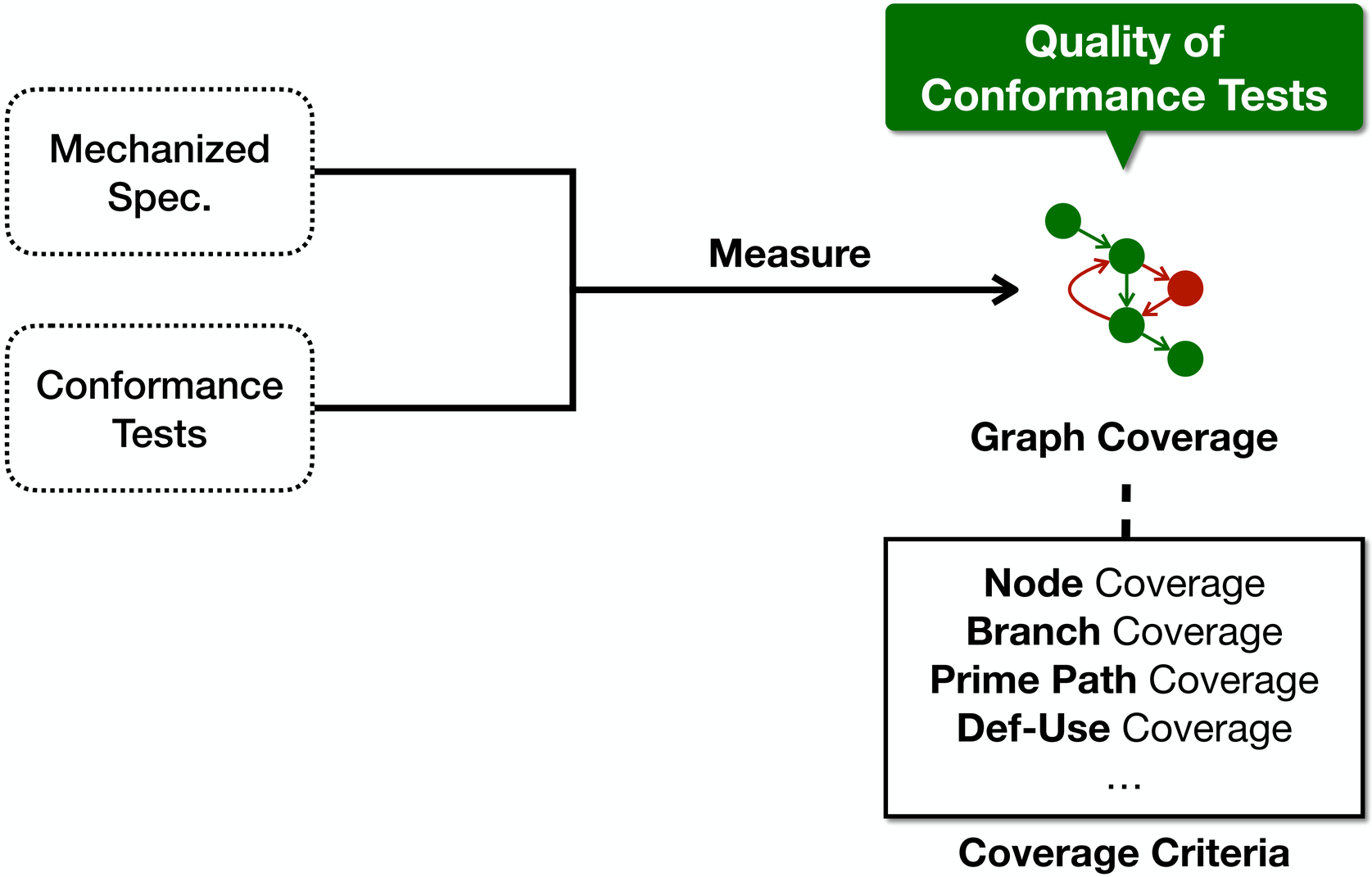
[PLDI 2023] J. Park et al., “Feature-Sensitive Coverage for Conformance Testing of Programming Language Implementations”



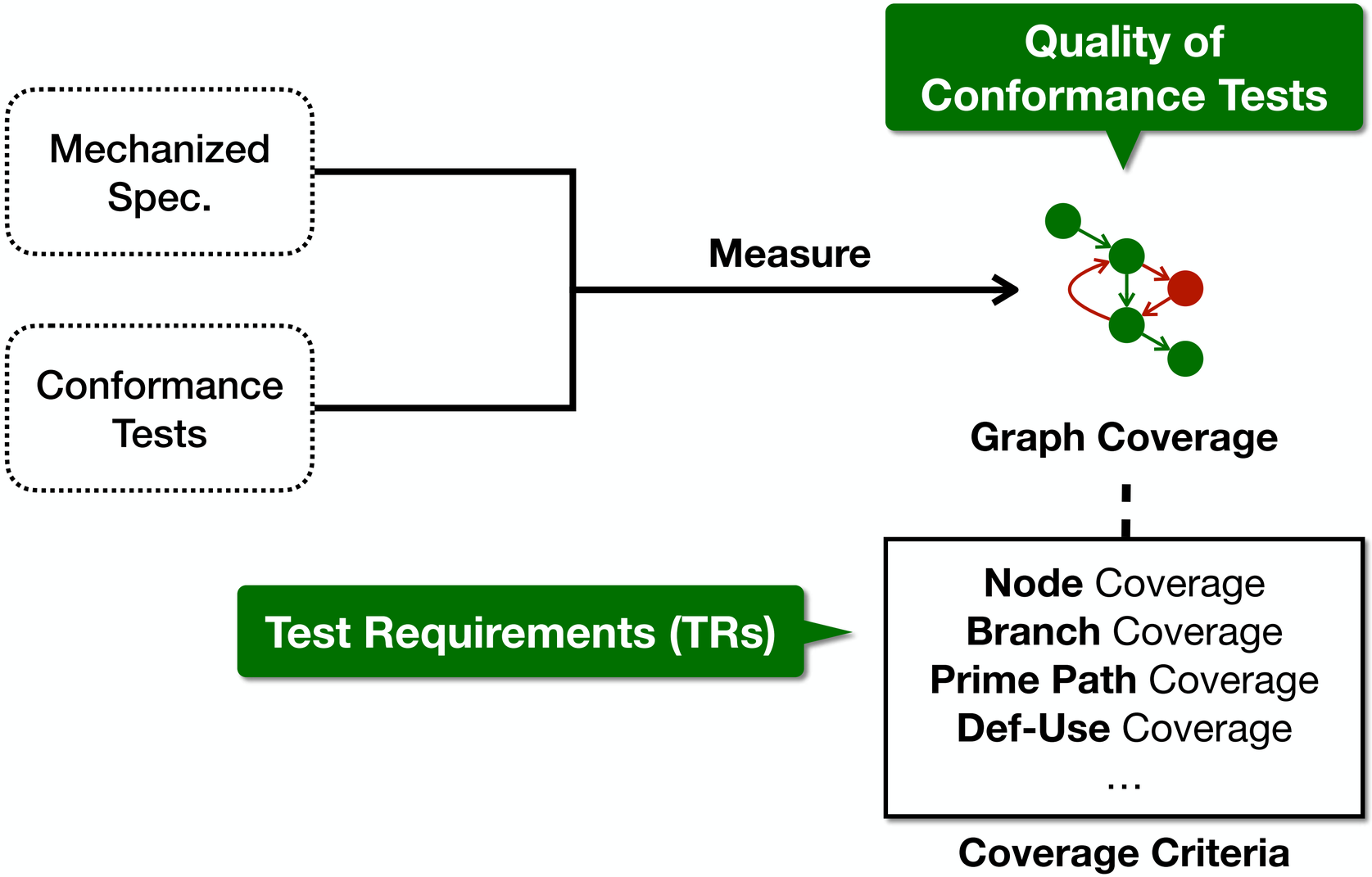
Graph Coverage for Language Specification



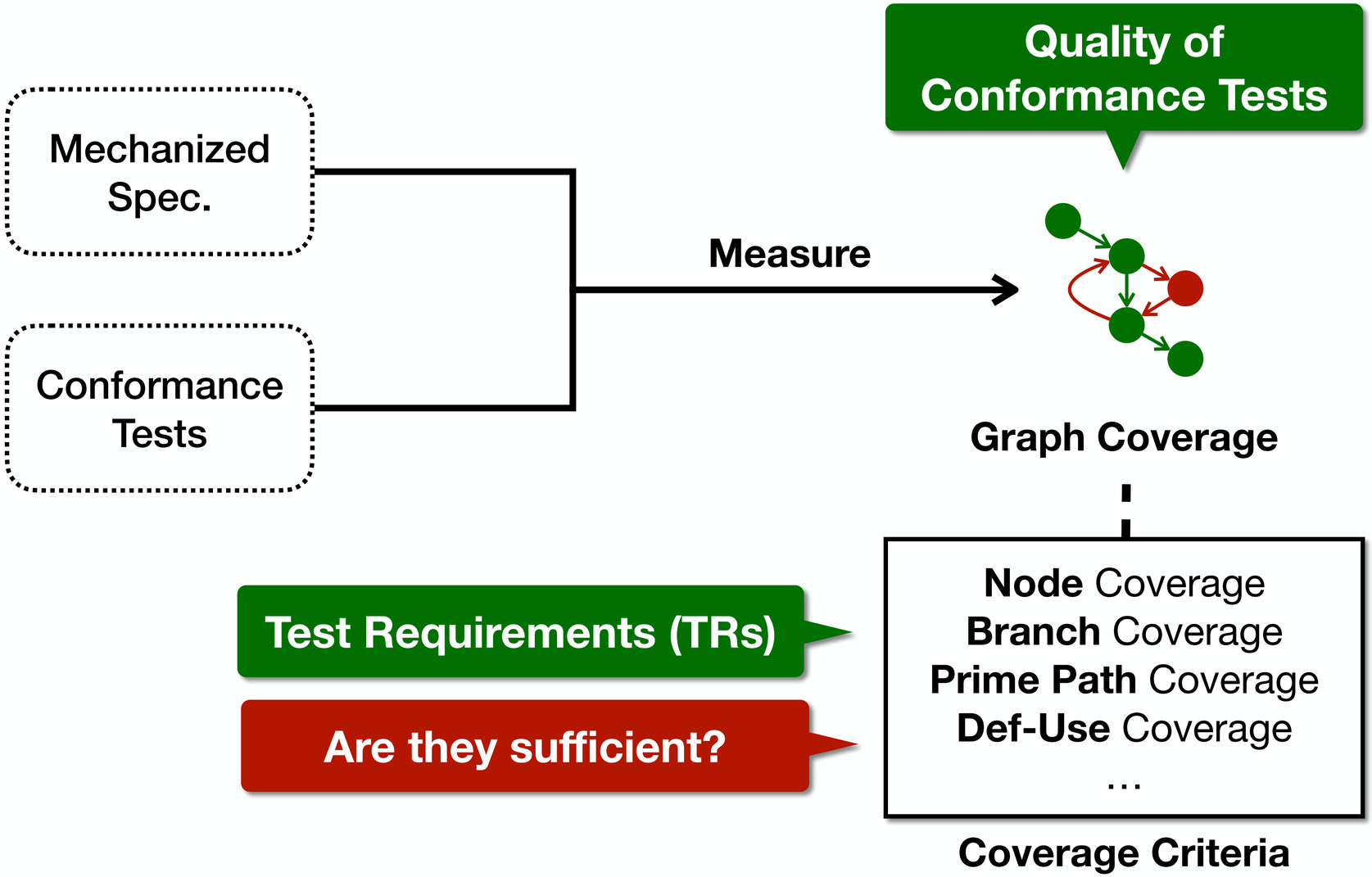
Graph Coverage for Language Specification



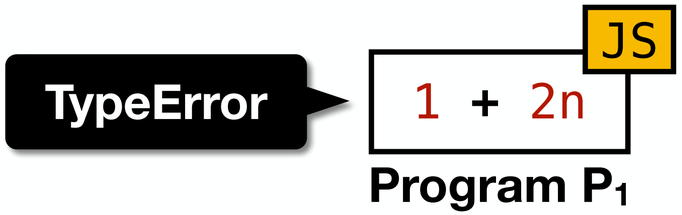
Graph Coverage for Language Specification



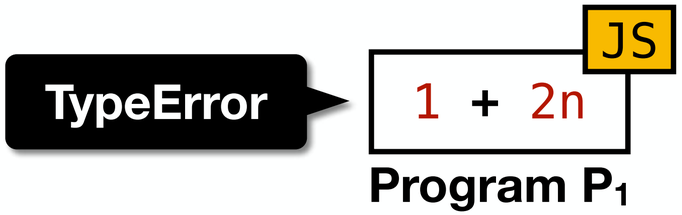
Graph Coverage for Language Specification



Motivating Example 1 with Node Coverage



Motivating Example 1 with Node Coverage



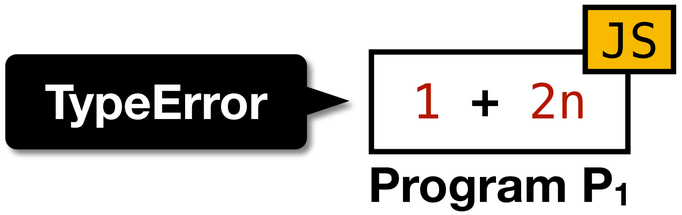
feat **ADD**

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



feat: **ADD**

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

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

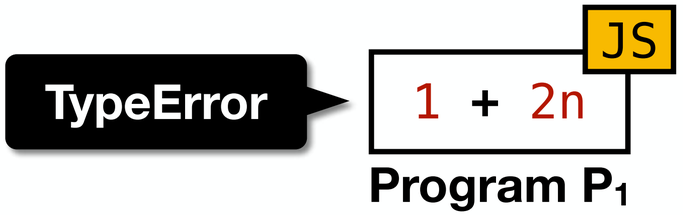
EvalStrOrNumBinExpr (*lval*, *opText*, *rval*)

…

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

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

Motivating Example 1 with Node Coverage



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*)

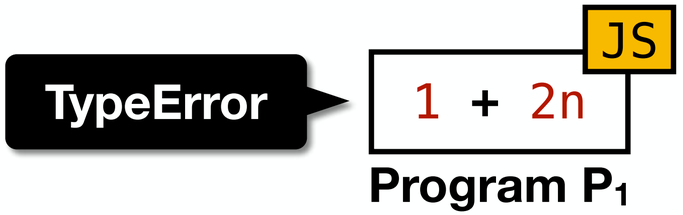
…

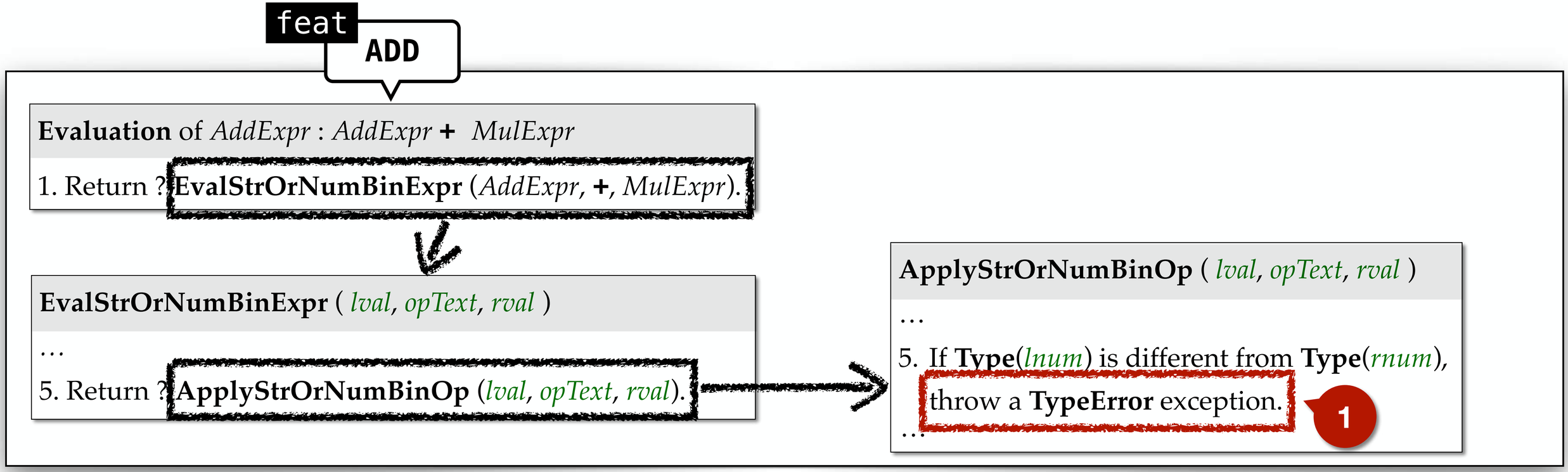
5. If **Type**(*lnum*) is different from **Type**(*rnum*), throw a **TypeError** exception.

…

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

Motivating Example 1 with Node Coverage





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

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.

…

1

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

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.

…

1

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

Motivating Example 1 with Node Coverage

Node Coverage

TR = Node

TypeError `1 + 2n` JS

Program P_1 — 1 — **cover**

TypeError `1 - 2n` JS

Program P_2

`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*)

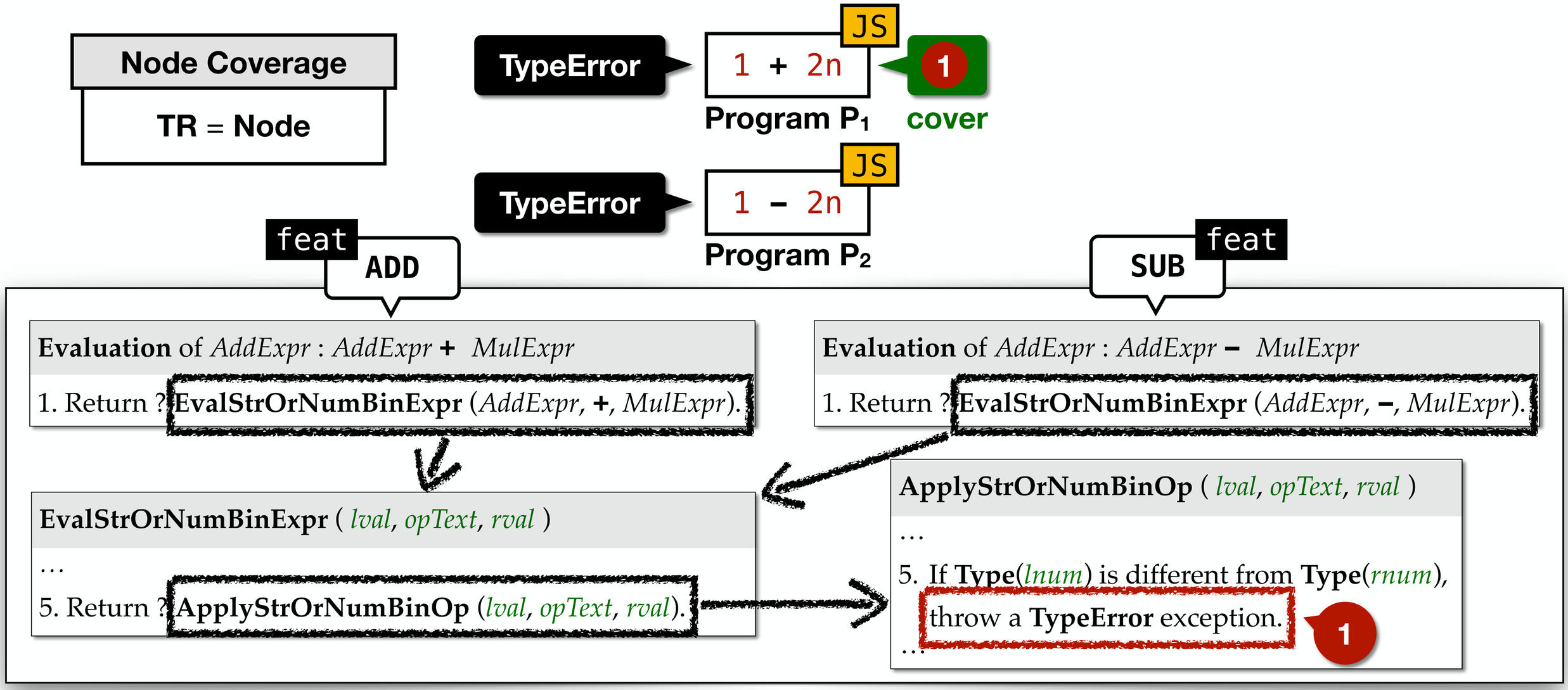
…

5. If **Type**(*lnum*) is different from **Type**(*rnum*),
   throw a **TypeError** exception. — 1

…

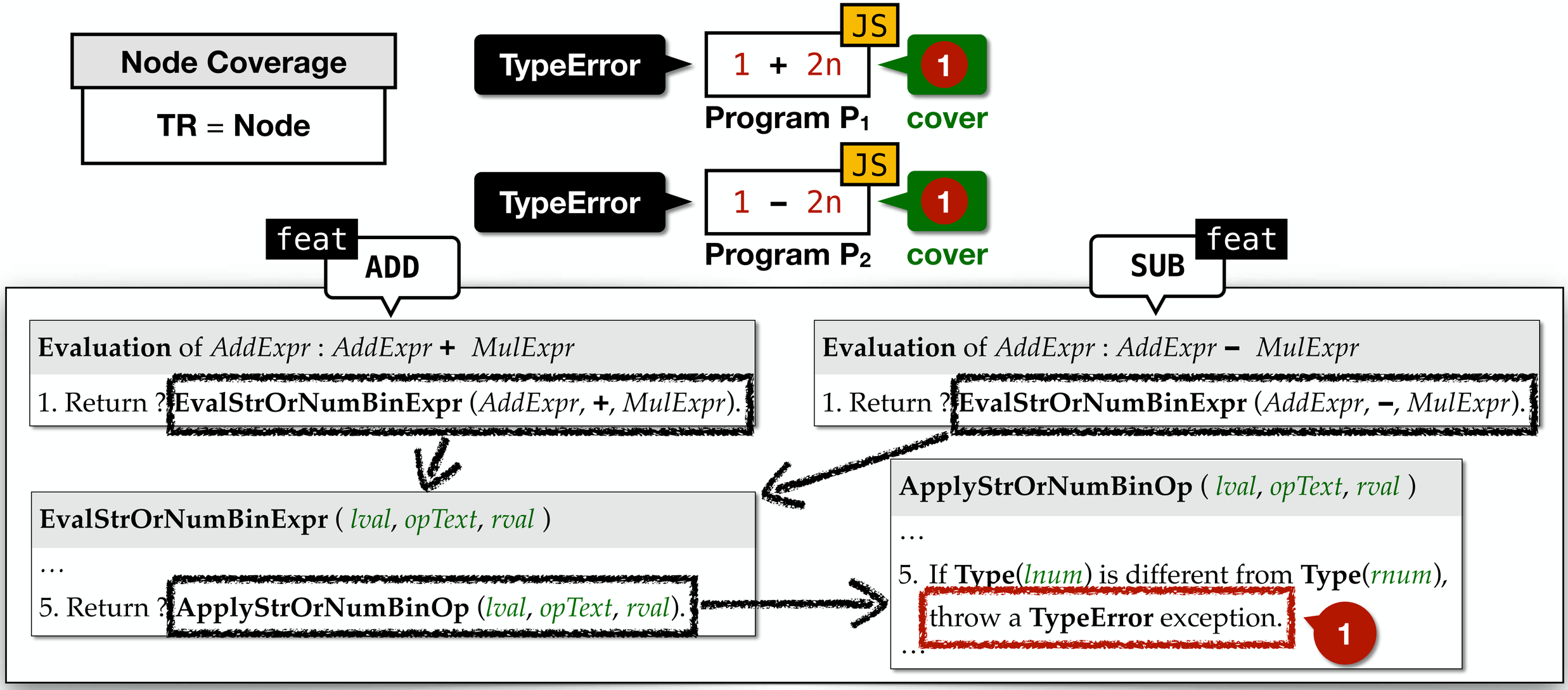
Abstract Algorithms in **ECMA-262** (ES13, 2022), **JavaScript Language Specification**

Motivating Example 1 with Node Coverage



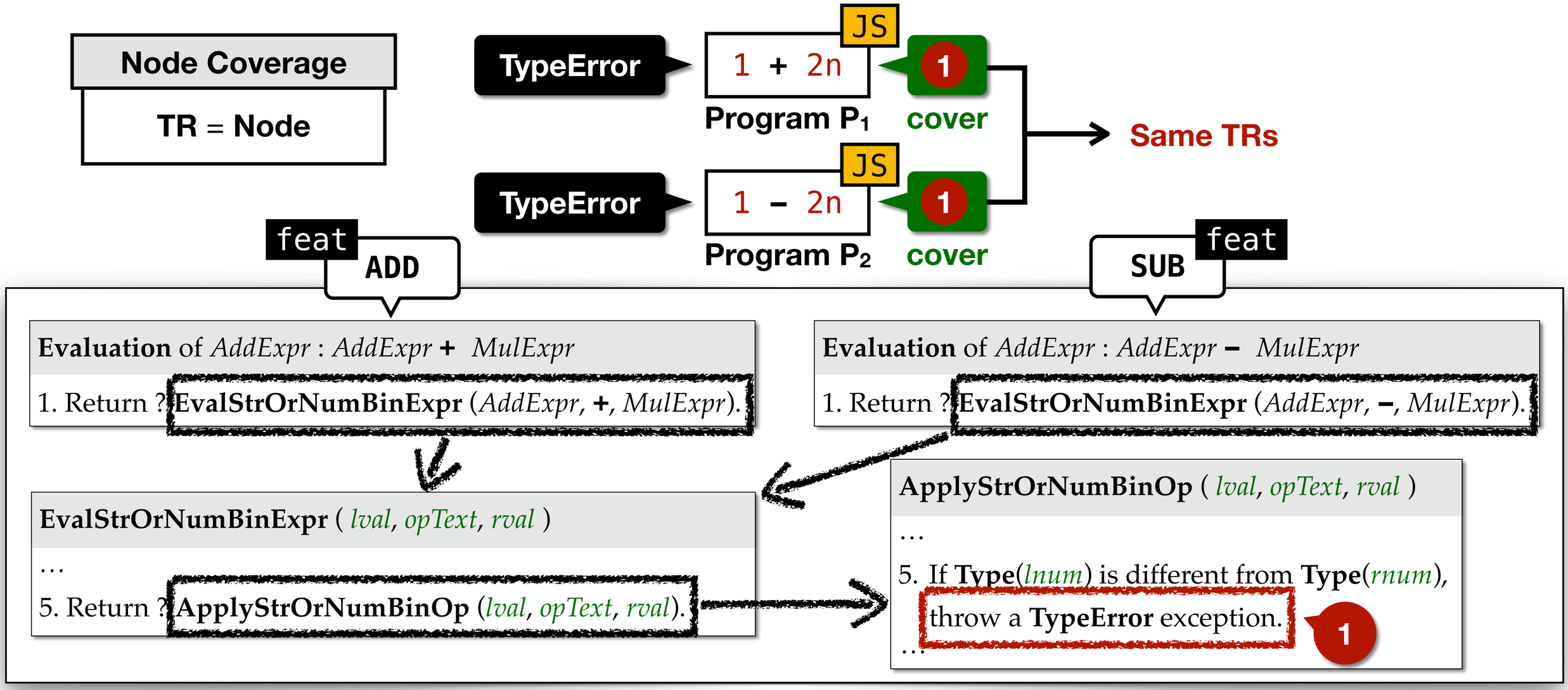
Abstract Algorithms in **ECMA-262** (ES13, 2022), **JavaScript Language Specification**

Motivating Example 1 with Node Coverage



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

Motivating Example 1 with Node Coverage



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**

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

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

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

…

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

feat **SUB**

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

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

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

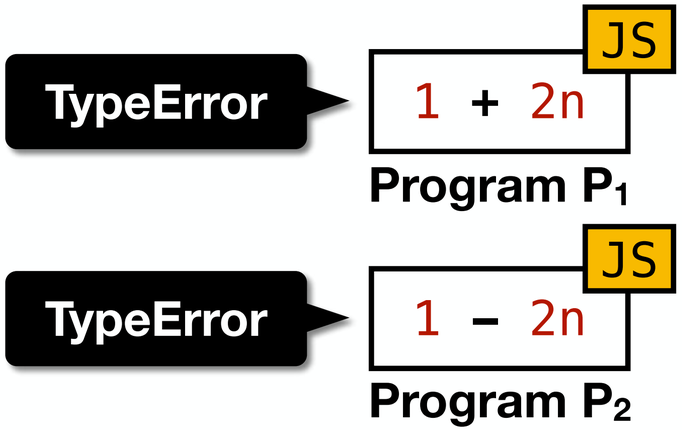
…

5. If **Type**(*lnum*) is different from **Type**(*rnum*),
   throw a **TypeError** exception. — 1

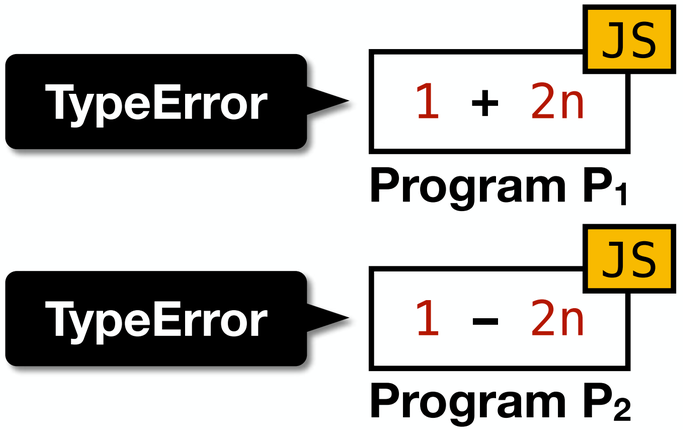
…

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

Feature-Sensitive (FS) Coverage



Feature-Sensitive (FS) Coverage

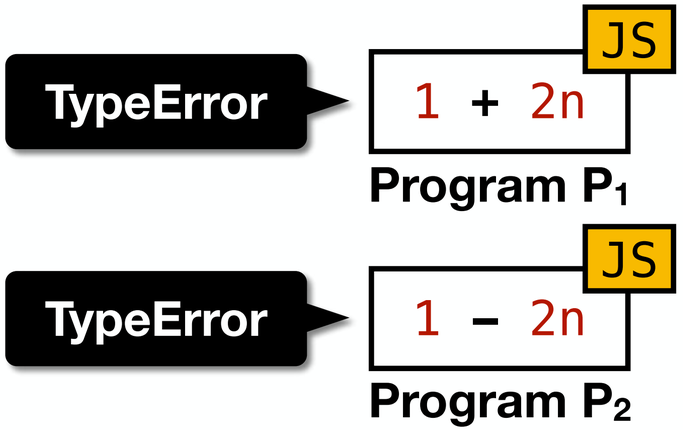


- Feature-Sensitive **(FS)** coverage criterion **divides**

  the given TRs with the **innermost enclosing** language **features**

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

Feature-Sensitive (FS) Coverage

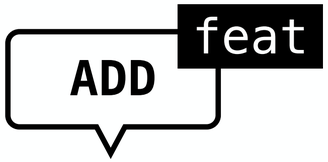


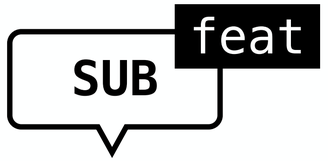
- Feature-Sensitive **(FS)** coverage criterion **divides**

  the given TRs with the **innermost enclosing** language **features**

FS Coverage

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





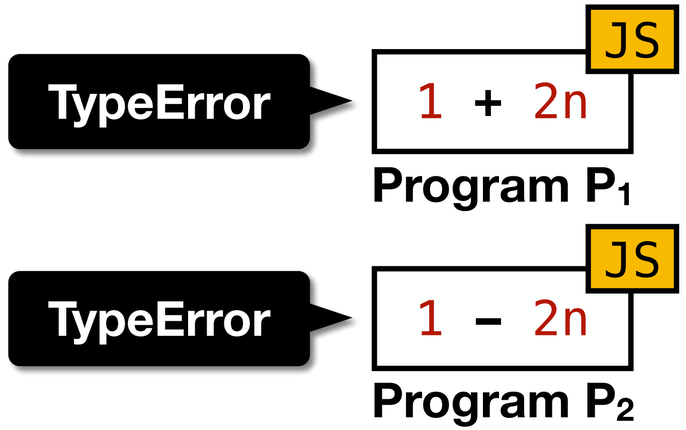
Evaluation of *AddExpr* : *AddExpr* **+** *MulExpr*
1. Return ? **EvalStrOrNumBinExpr** (*AddExpr*, **+**, *MulExpr*).

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

Feature-Sensitive (FS) Coverage

FS Node Coverage

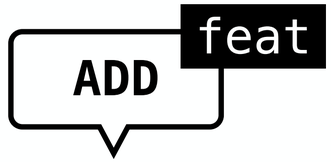
TR = (Feature, Node)

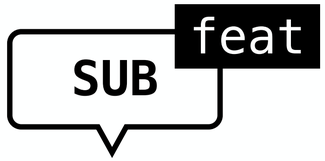


- **F**eature-**S**ensitive **(FS)** coverage criterion **divides** the given TRs with the **innermost enclosing** language **features**

FS Coverage

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





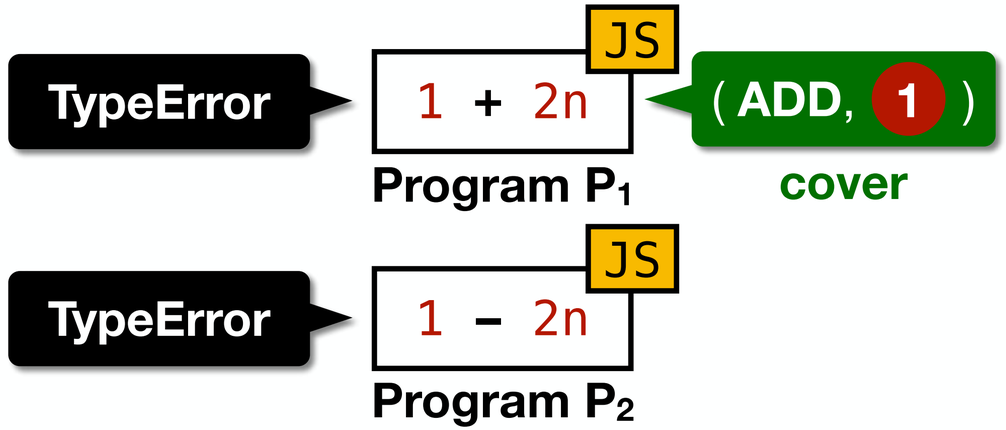
Evaluation of *AddExpr* : *AddExpr* **+** *MulExpr*
1. Return ? **EvalStrOrNumBinExpr** (*AddExpr*, **+**, *MulExpr*).

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

Feature-Sensitive (FS) Coverage

FS Node Coverage

TR = (Feature, Node)

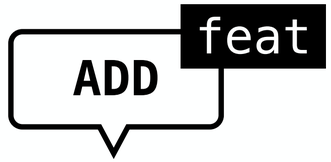


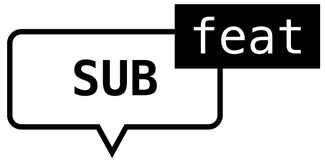
- Feature-Sensitive **(FS)** coverage criterion **divides**

  the given TRs with the **innermost enclosing** language **features**

FS Coverage

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





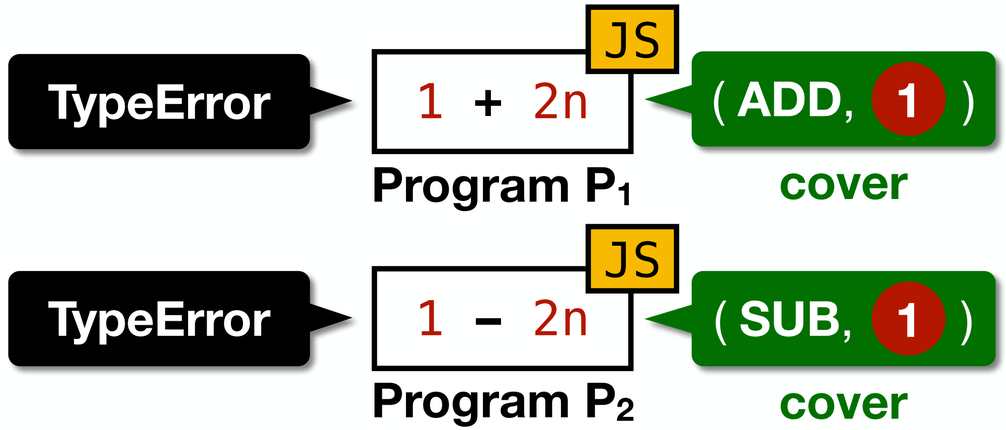
Evaluation of *AddExpr* : *AddExpr* **+** *MulExpr*
1. Return ? **EvalStrOrNumBinExpr** (*AddExpr*, **+**, *MulExpr*).

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

Feature-Sensitive (FS) Coverage

FS Node Coverage

TR = (Feature, Node)

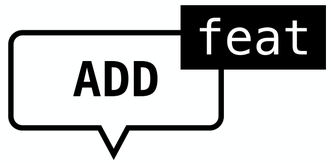


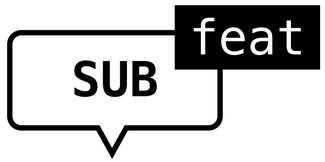
- Feature-Sensitive **(FS)** coverage criterion **divides**

  the given TRs with the **innermost enclosing** language **features**

FS Coverage

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





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

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

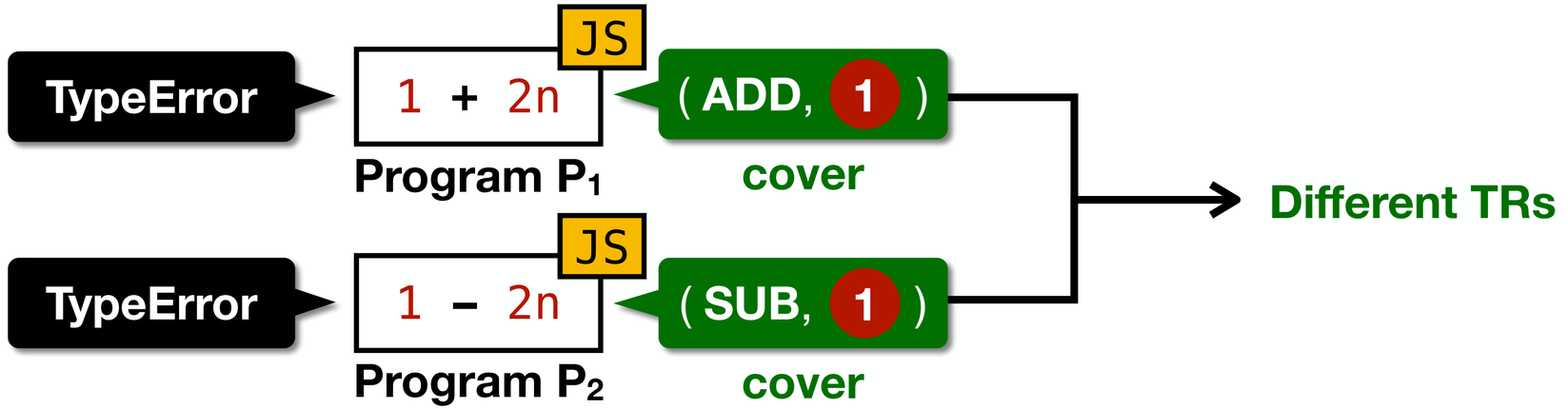
Evaluation of *AddExpr* : *AddExpr* **–** *MulExpr*

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

Feature-Sensitive (FS) Coverage

FS Node Coverage

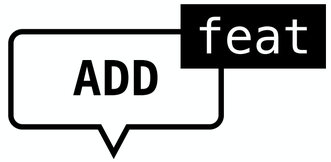
TR = (Feature, Node)

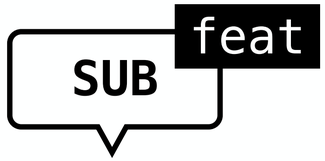


- **F**eature-**S**ensitive **(FS)** coverage criterion **divides** the given TRs with the **innermost enclosing** language **features**

FS Coverage

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

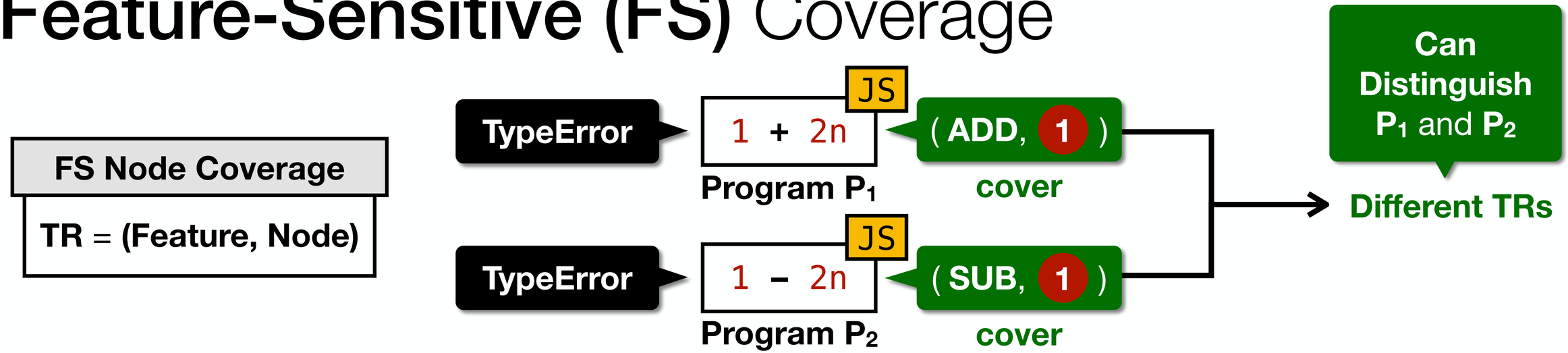




Evaluation of *AddExpr* : *AddExpr* **+** *MulExpr*
1. Return ? **EvalStrOrNumBinExpr** (*AddExpr*, **+**, *MulExpr*).

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

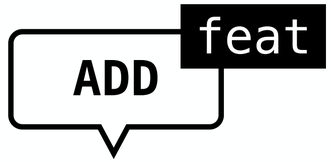
Feature-Sensitive (FS) Coverage

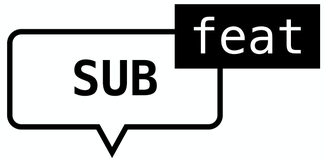


- Feature-Sensitive **(FS)** coverage criterion **divides** the given TRs with the **innermost enclosing** language **features**

FS Coverage

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





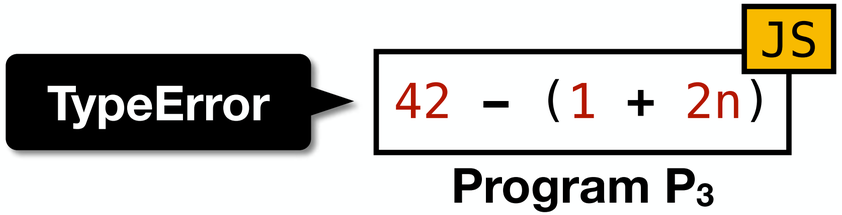
Evaluation of *AddExpr* : *AddExpr* **+** *MulExpr*

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

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

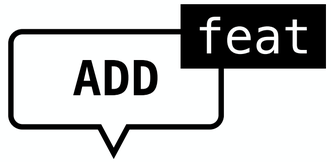
1. Return ? **EvalStrOrNumBinExpr** (*AddExpr*, **–**, *MulExpr*).

k-Feature-Sensitive (k-FS) Coverage

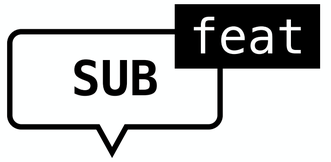


- k-**F**eature-**S**ensitive (**k-FS**) coverage criterion **divides** the given TRs with at most **k-innermost enclosing** language **features**

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$^{\leq 2}$, 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$^{\leq k}$, given TR)

feat **ADD**

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

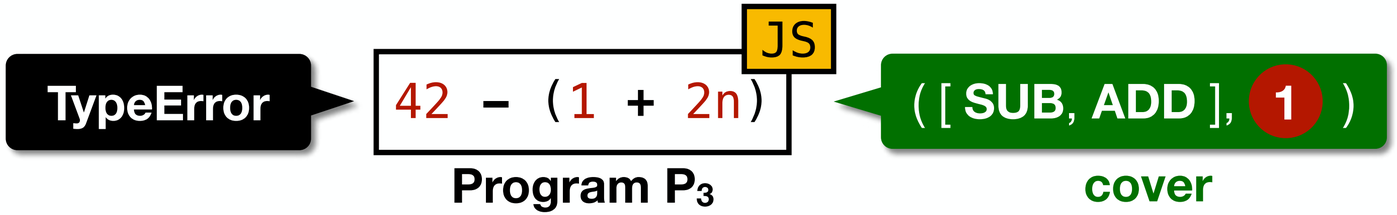
feat **SUB**

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

k-Feature-Sensitive (k-FS) Coverage

2-FS Node Coverage

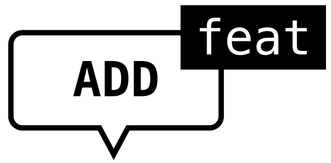
TR = (Feature$^{\leq 2}$, Node)

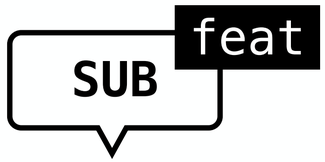


- k-**F**eature-**S**ensitive (**k-FS**) coverage criterion **divides** the given TRs with at most **k-innermost enclosing** language **features**

k-FS Coverage

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

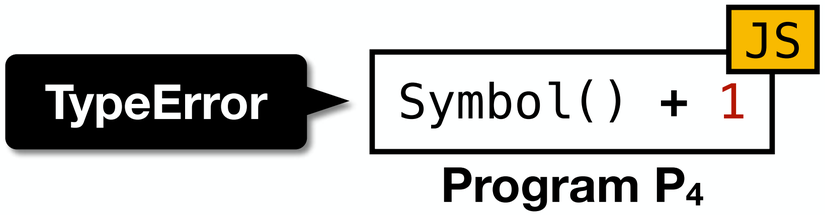




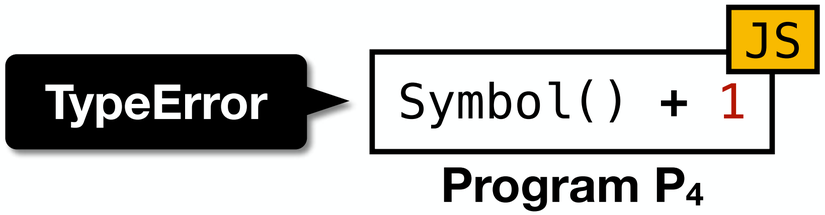
Evaluation of *AddExpr* : *AddExpr* **+** *MulExpr*
1. Return ? **EvalStrOrNumBinExpr** (*AddExpr*, **+**, *MulExpr*).

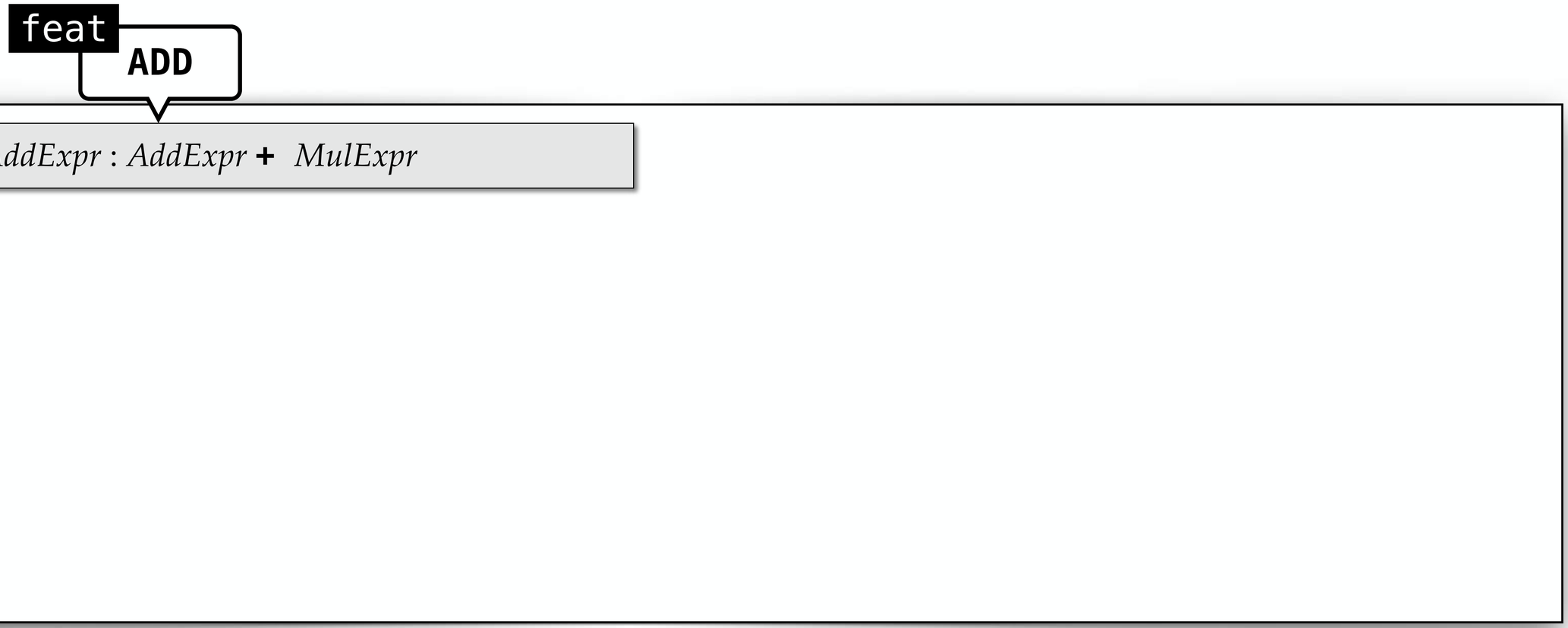
Evaluation of *AddExpr* : *AddExpr* **–** *MulExpr*
1. Return ? **EvalStrOrNumBinExpr** (*AddExpr*, **–**, *MulExpr*).

Motivating Example 2

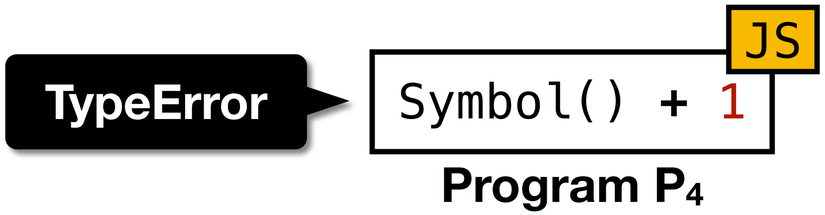


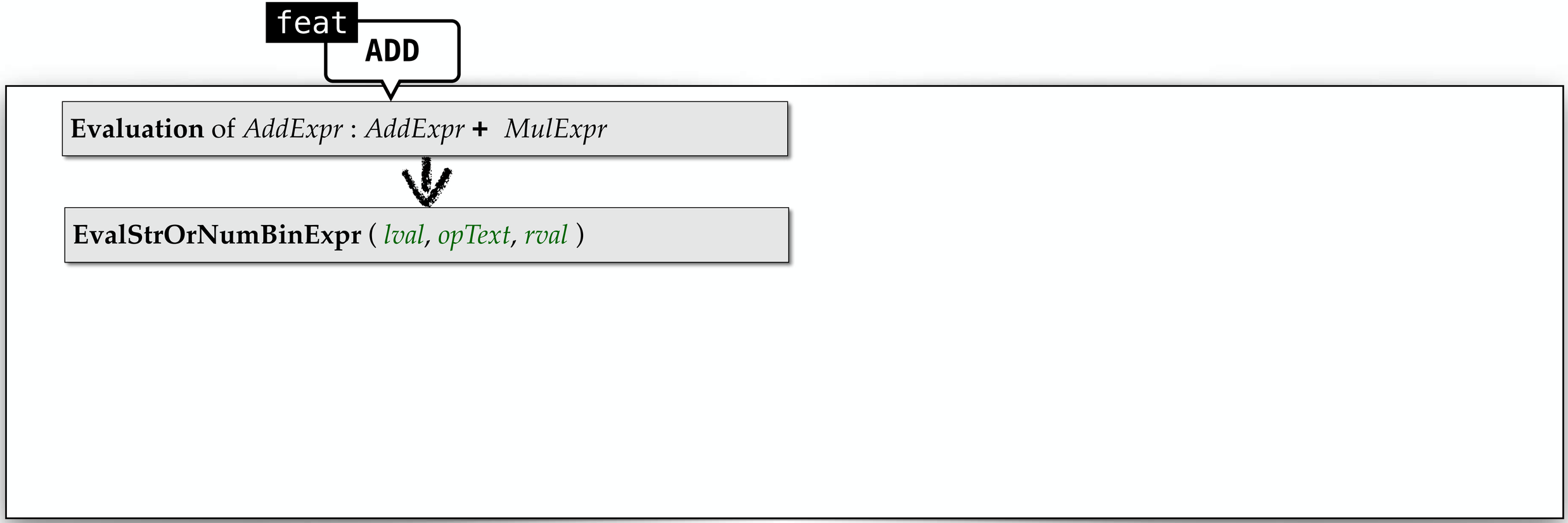
Motivating Example 2



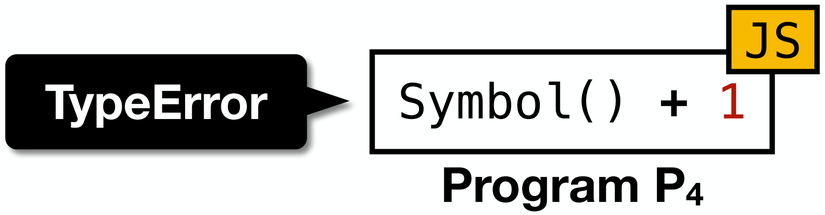


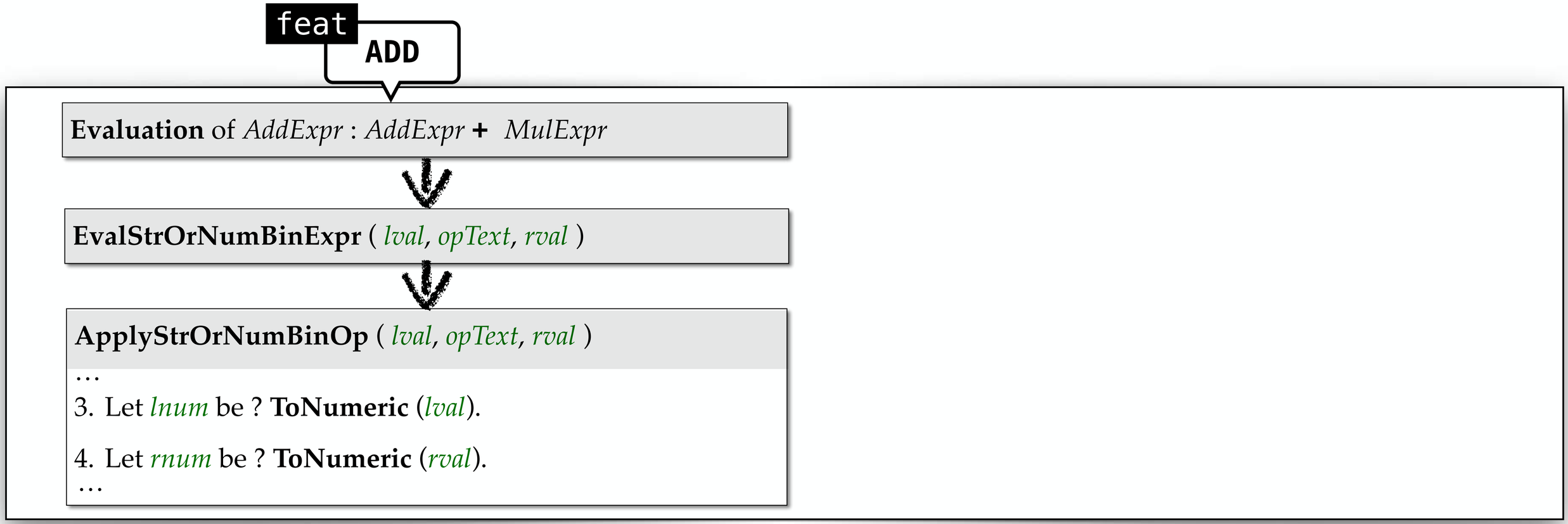
Motivating Example 2



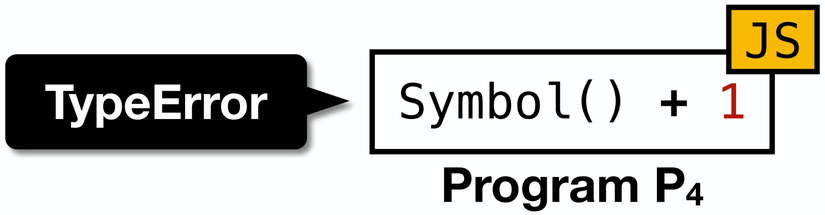


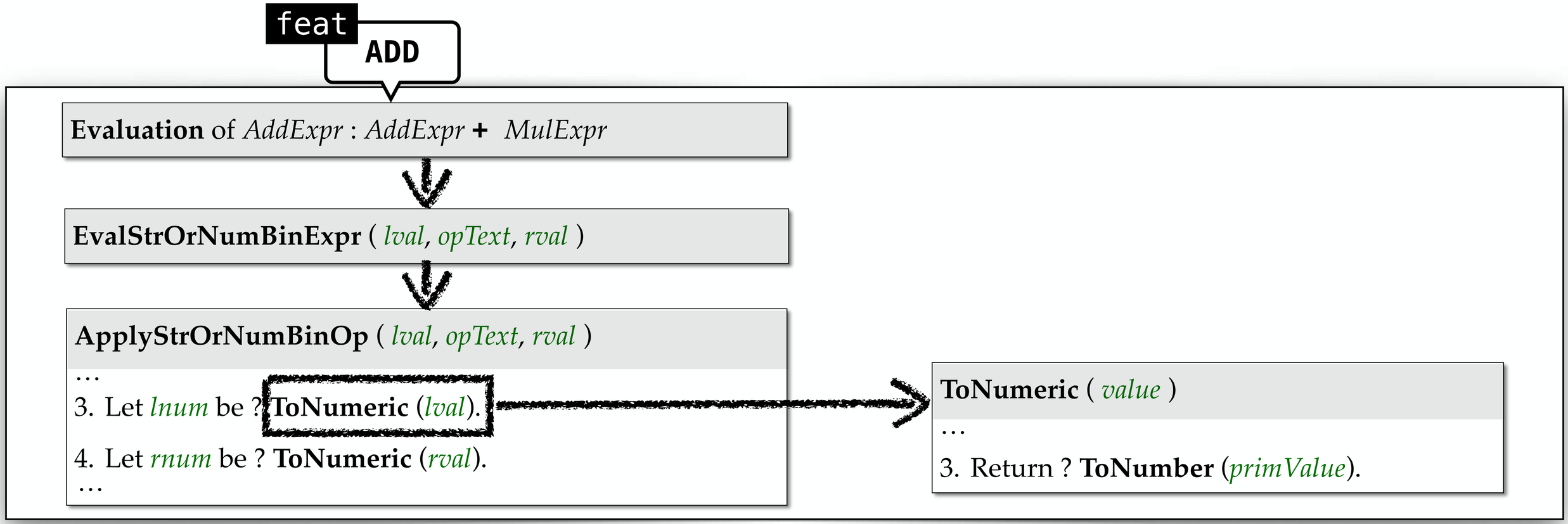
Motivating Example 2



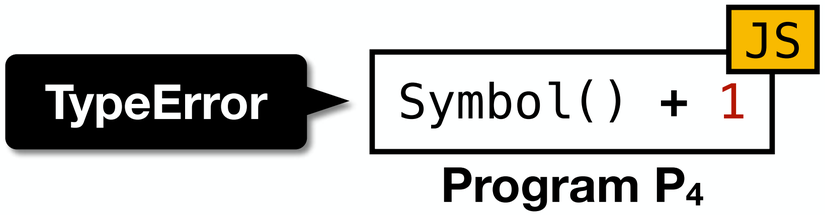


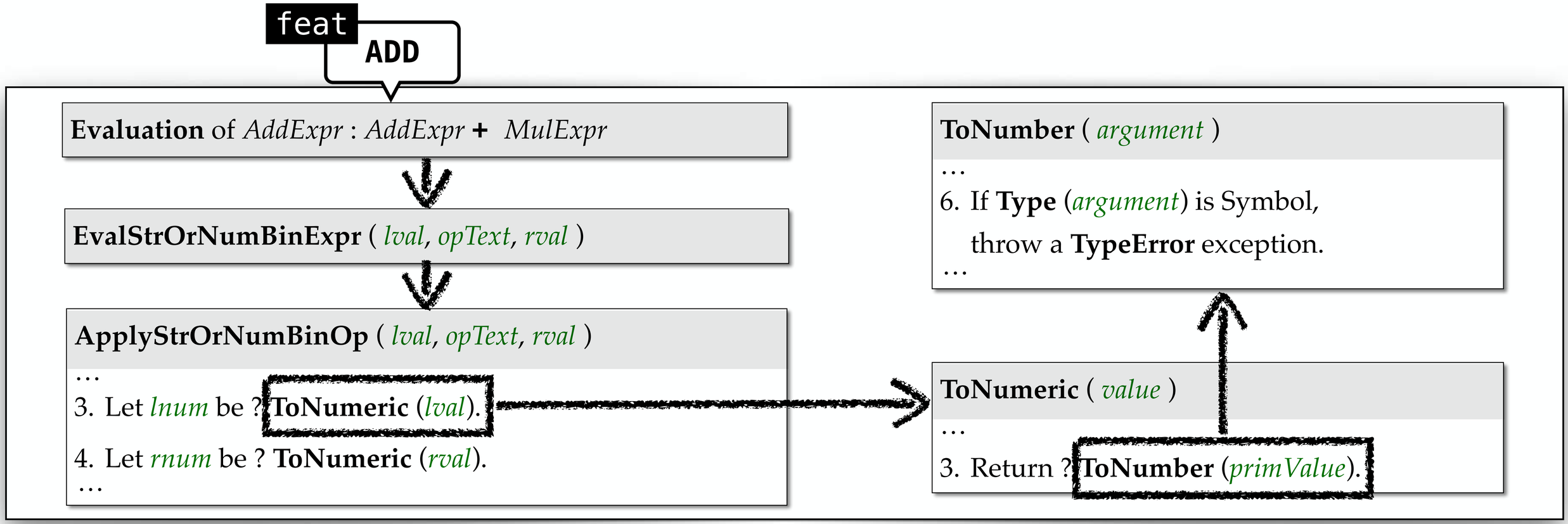
Motivating Example 2



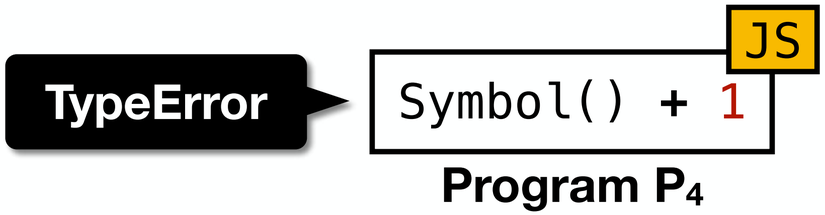


Motivating Example 2





Motivating Example 2



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

`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

(ADD, 2)

cover

`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 `Symbol() + 1` JS

Program P_4

(**ADD**, **2**) **cover**

TypeError `1 + Symbol()` JS

Program P_5

`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 `Symbol() + 1` JS

Program P_4

(**ADD**, **2**) **cover**

TypeError `1 + Symbol()` JS

Program P_5

`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 `Symbol() + 1` JS — **Program P_4** — (**ADD**, **2**) **cover**

TypeError `1 + Symbol()` JS — **Program P_5** — (**ADD**, **2**) **cover**

`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*)

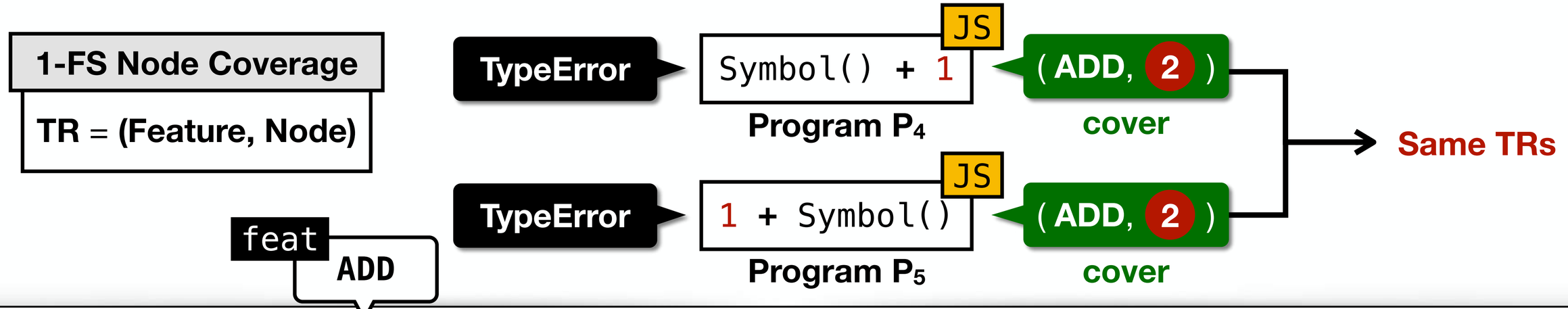
…

6. If **Type** (*argument*) is Symbol,

throw a **TypeError** exception. **2**

…

Motivating Example 2



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 `Symbol() + 1` (JS) — **Program P_4** — **(ADD, 2)** **cover**

TypeError `1 + Symbol()` (JS) — **Program P_5** — **(ADD, 2)** **cover**

Cannot Distinguish P_4 and P_5

Same TRs

`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*)

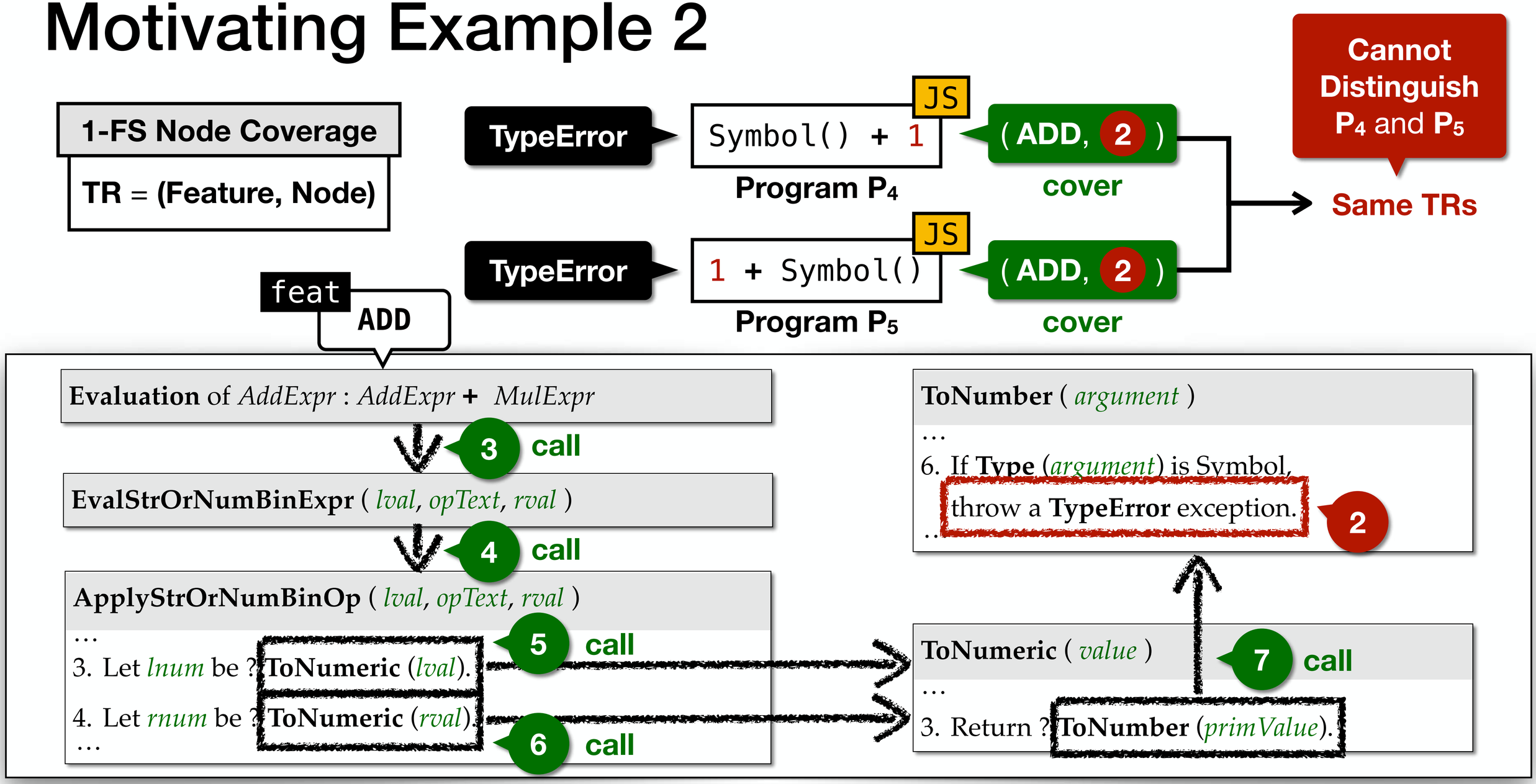
…

6. If **Type** (*argument*) is Symbol,

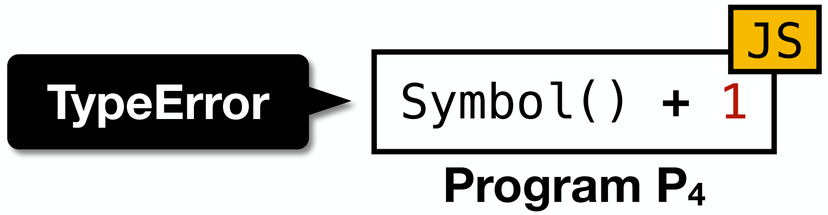
throw a **TypeError** exception. 2

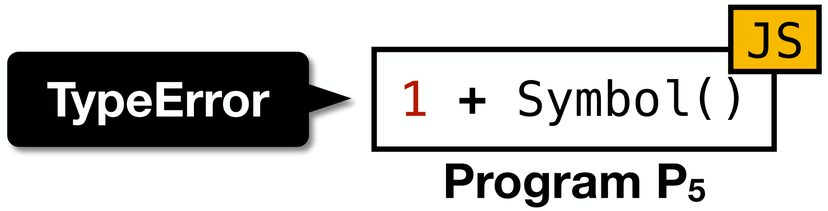
…

Motivating Example 2



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



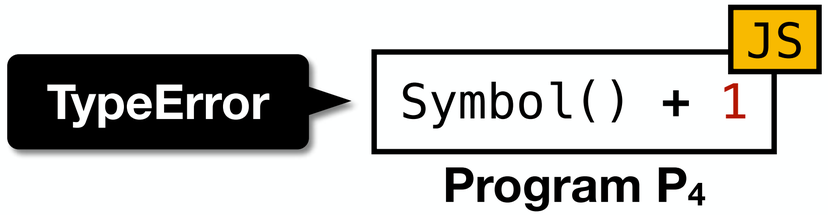


- k-**F**eature-**C**all-**P**ath-**S**ensitive (k-**FCPS**) coverage criterion divides the k-FS TRs with the **call-paths from** the innermost enclosing language feature

k-FCPS Coverage

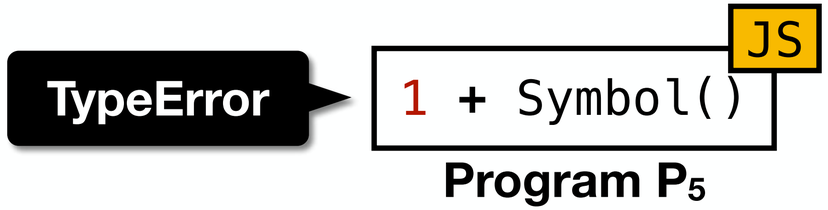
TR = (Feature$^{\leq k}$, Call-Path, given **TR)**

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



1-FCPS Node Coverage

TR = (Feature, Call-Path, Node)

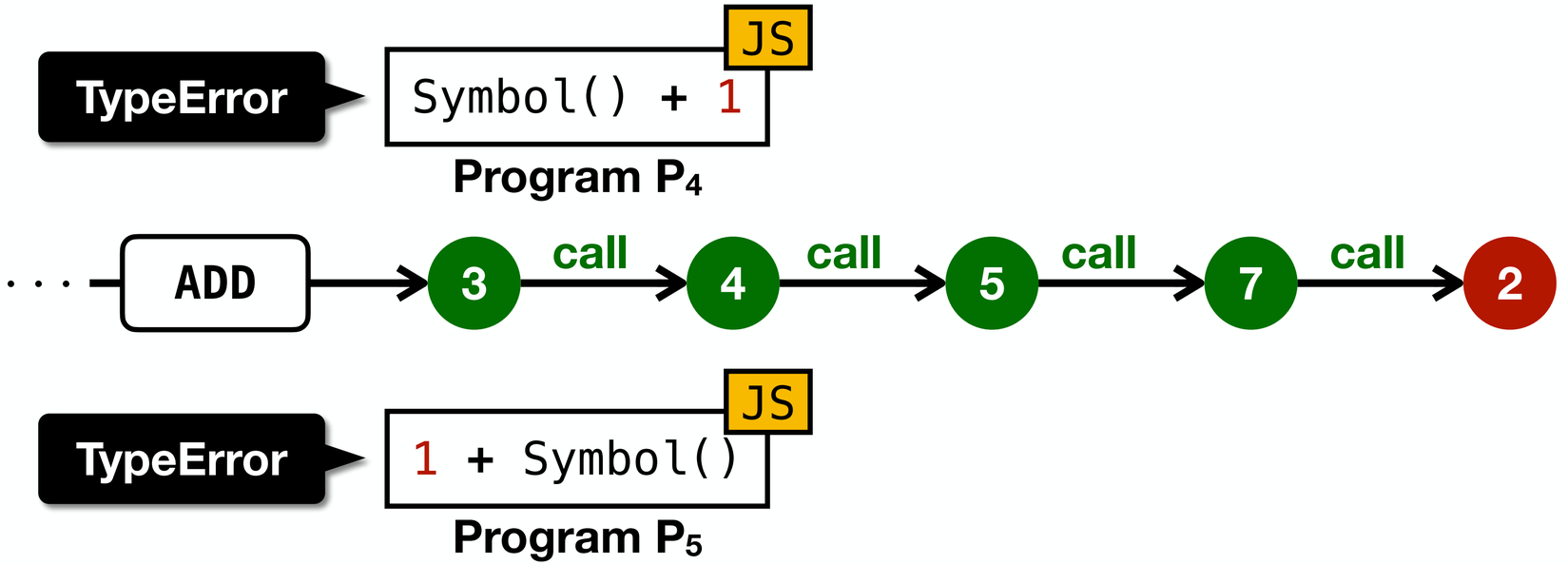


- k-**F**eature-**C**all-**P**ath-**S**ensitive (k-**FCPS**) coverage criterion divides the k-FS TRs with the **call-paths from** the innermost enclosing language feature

k-FCPS Coverage

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

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



1-FCPS Node Coverage

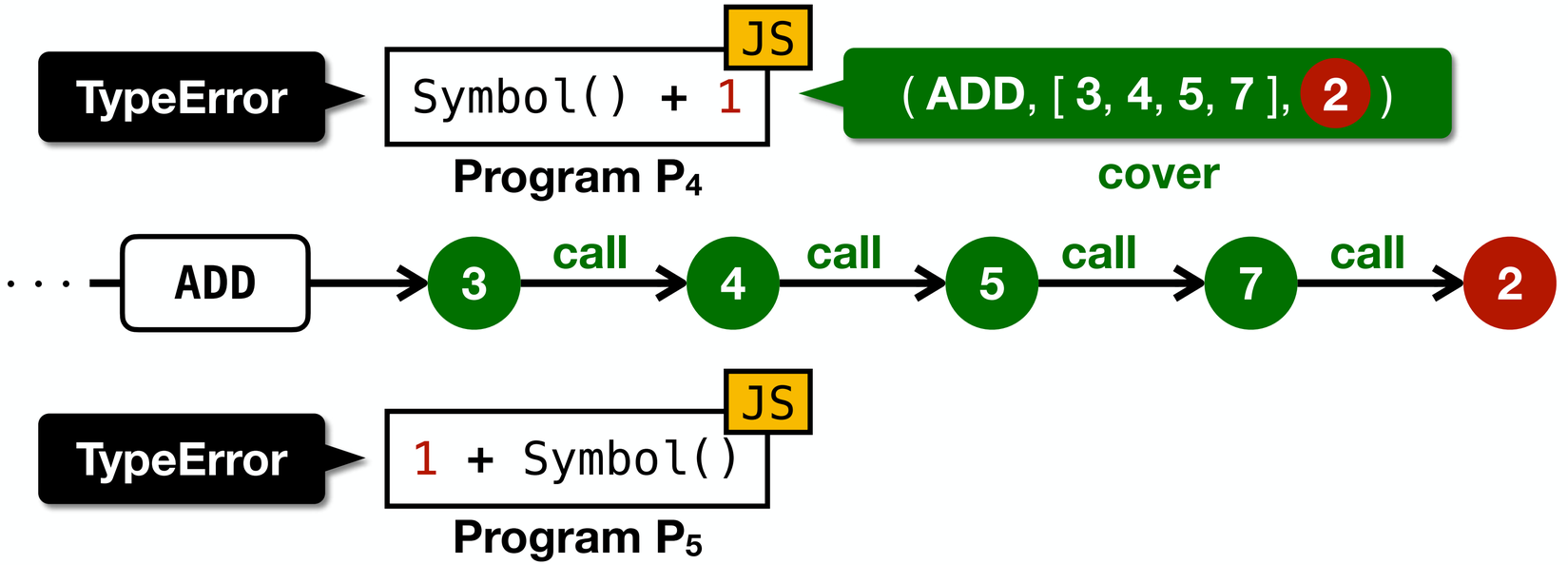
TR = (Feature, Call-Path, Node)

- 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

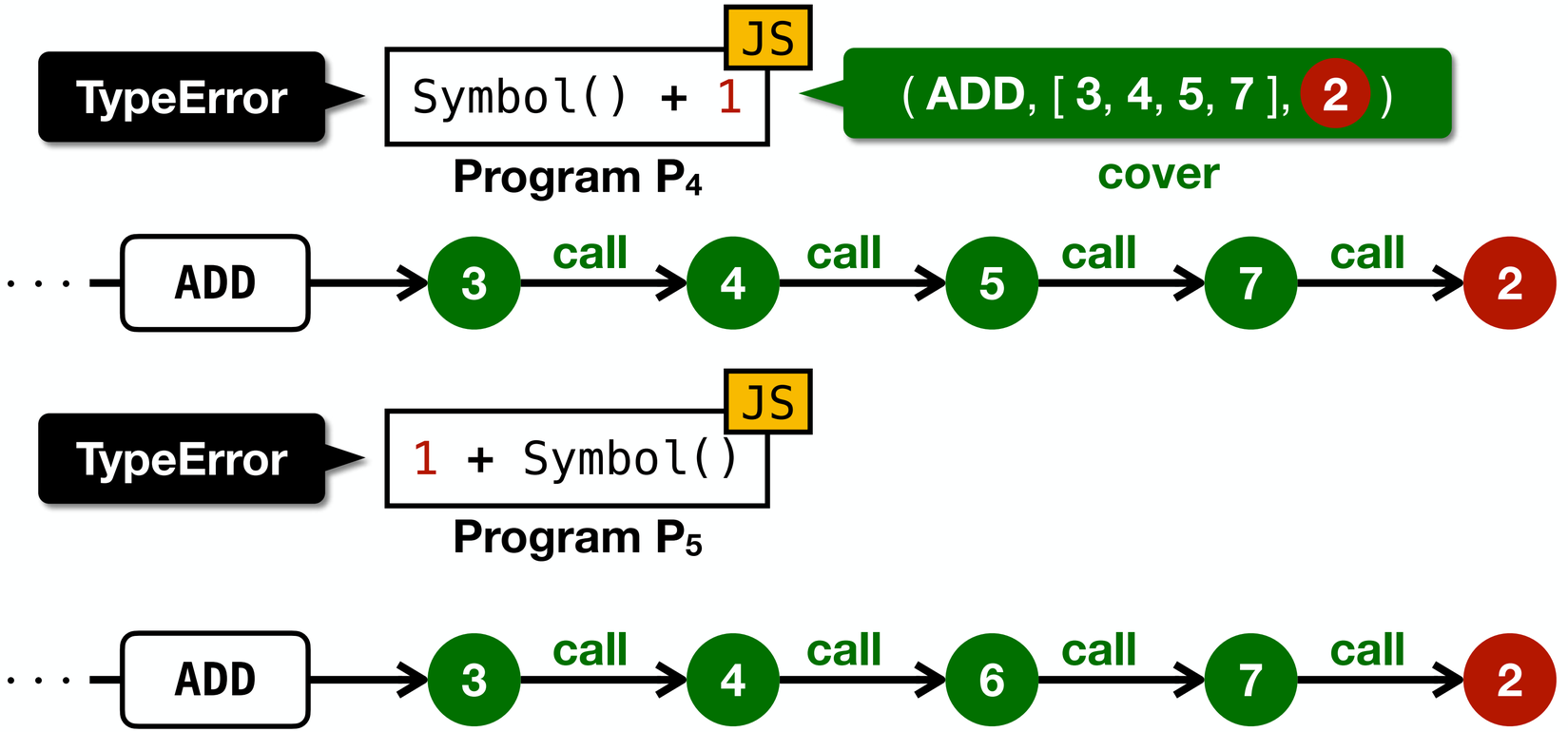
TR = (Feature, Call-Path, Node)

- 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



1-FCPS Node Coverage

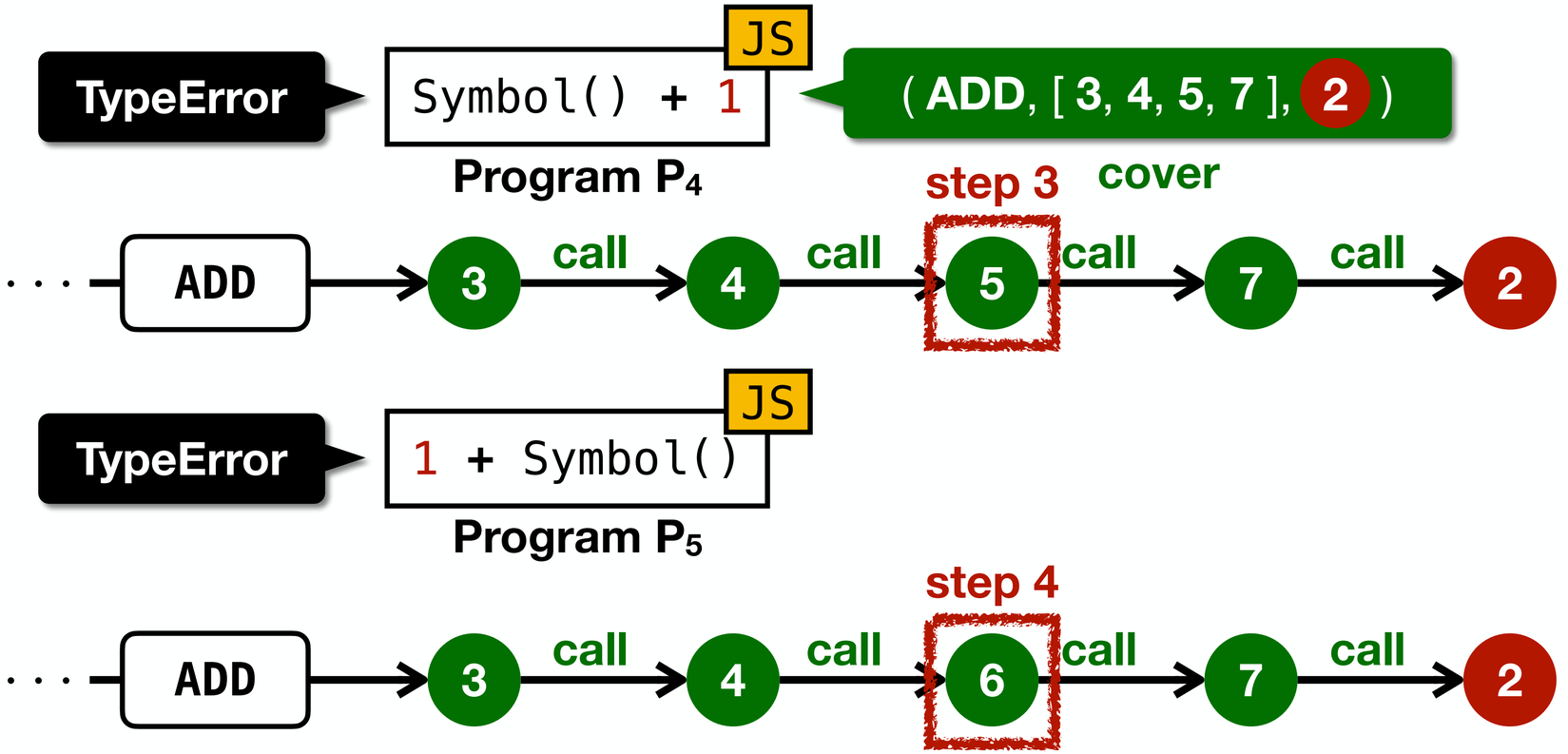
TR = (Feature, Call-Path, Node)

- 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

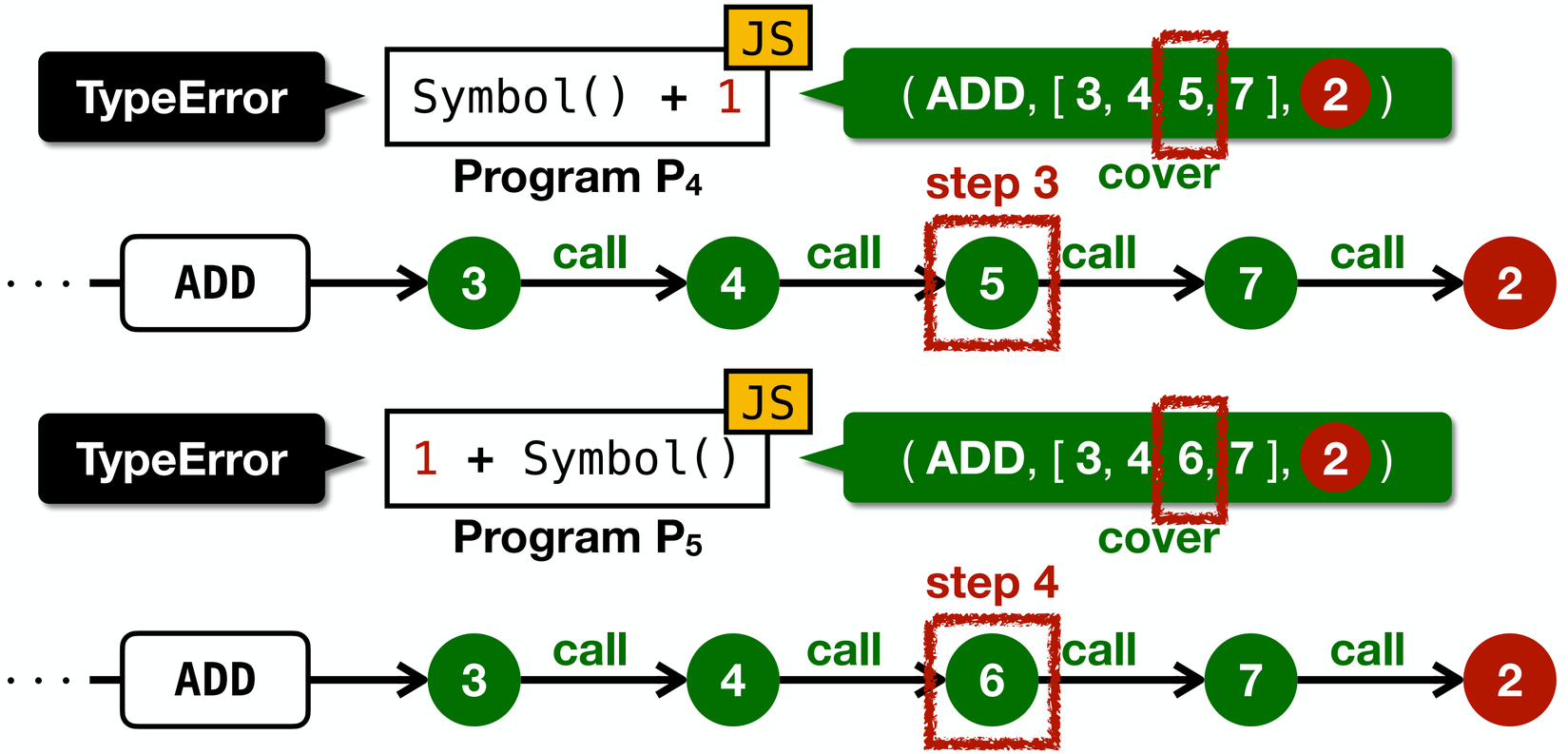
TR = (Feature, Call-Path, Node)

- 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



1-FCPS Node Coverage

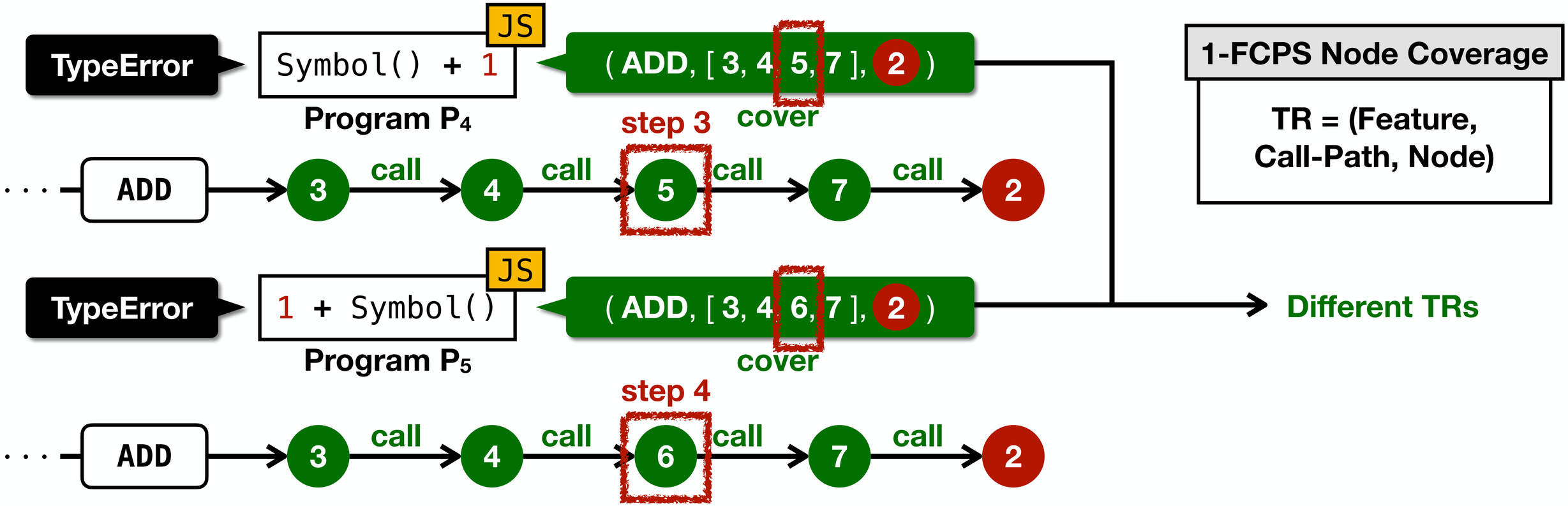
TR = (Feature, Call-Path, Node)

- 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

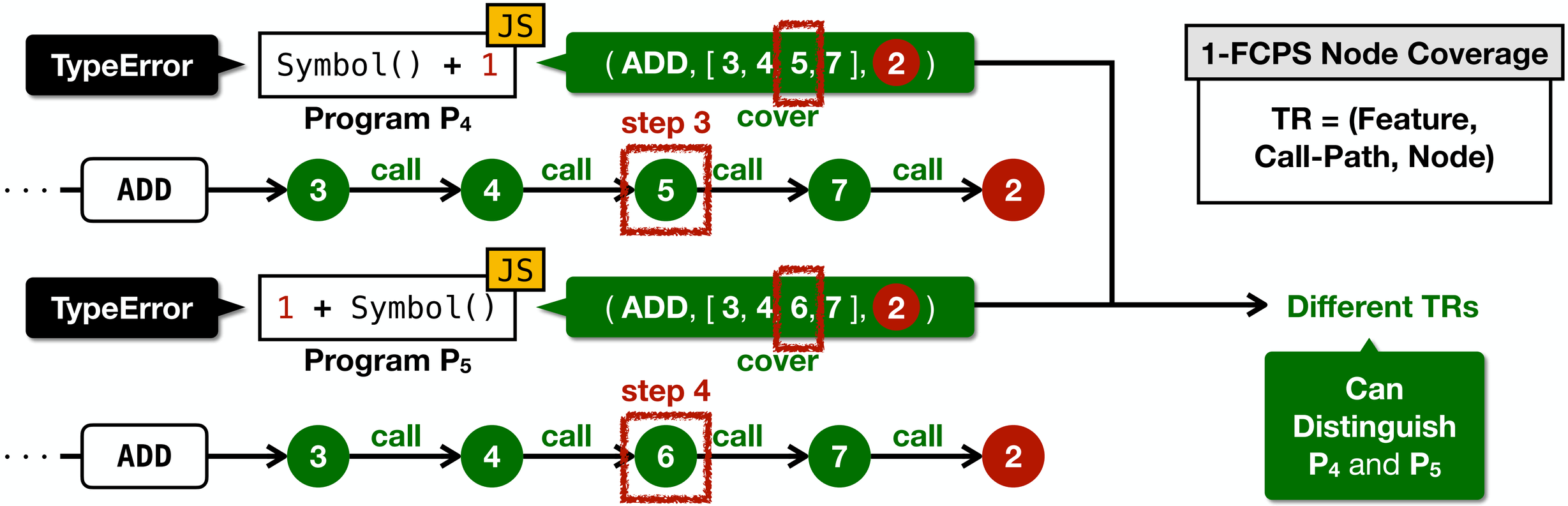


- k-**F**eature-**C**all-**P**ath-**S**ensitive (k-**FCPS**) coverage criterion divides the k-FS TRs with the **call-paths from** the innermost enclosing language feature

k-FCPS Coverage

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

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

- **Evaluation** with **ES2022** in 50 hours with **0-FS** / **1-FS** / **2-FS** / **1-FCPS** / **2-FCPS**

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**

Evaluation

- **Evaluation** with **ES2022** in 50 hours with **0-FS** / **1-FS** / **2-FS** / **1-FCPS** / **2-FCPS**

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**

Evaluation

- **Evaluation** with **ES2022** in 50 hours with **0-FS** / **1-FS** / **2-FS** / **1-FCPS** / **2-FCPS**

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**

Evaluation

- **Evaluation** with **ES2022** in 50 hours with **0-FS** / **1-FS** / **2-FS** / **1-FCPS** / **2-FCPS**

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**

Effectiveness of k-FS / k-FCPS Coverage Criteria

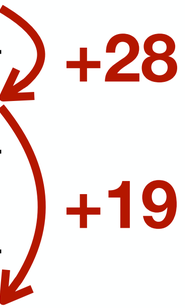
Coverage Criteria $C_{\mathbb{G}}$	# Syn. Test	# Bug
0-FS node-or-branch (0-fs)	2,111	55
1-FS node-or-branch (1-fs)	6,766	83
1-FCPS node-or-branch (1-fcps)	9,092	87
2-FS node-or-branch (2-fs)	97,423	102
2-FCPS node-or-branch (2-fcps)	122,589	111

Effectiveness of k-FS / k-FCPS Coverage Criteria

Coverage Criteria $C_{\mathbb{G}}$	# Syn. Test	# Bug
0-FS node-or-branch (0-fs)	2,111	55
1-FS node-or-branch (1-fs)	6,766	83
1-FCPS node-or-branch (1-fcps)	9,092	87
2-FS node-or-branch (2-fs)	97,423	102
2-FCPS node-or-branch (2-fcps)	122,589	111

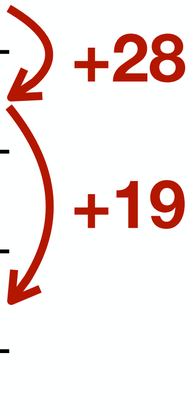
Effectiveness of k-FS / k-FCPS Coverage Criteria

Coverage Criteria $C_{\mathbb{G}}$	# Syn. Test	# Bug
0-FS node-or-branch (0-fs)	2,111	55
1-FS node-or-branch (1-fs)	6,766	83
1-FCPS node-or-branch (1-fcps)	9,092	87
2-FS node-or-branch (2-fs)	97,423	102
2-FCPS node-or-branch (2-fcps)	122,589	111

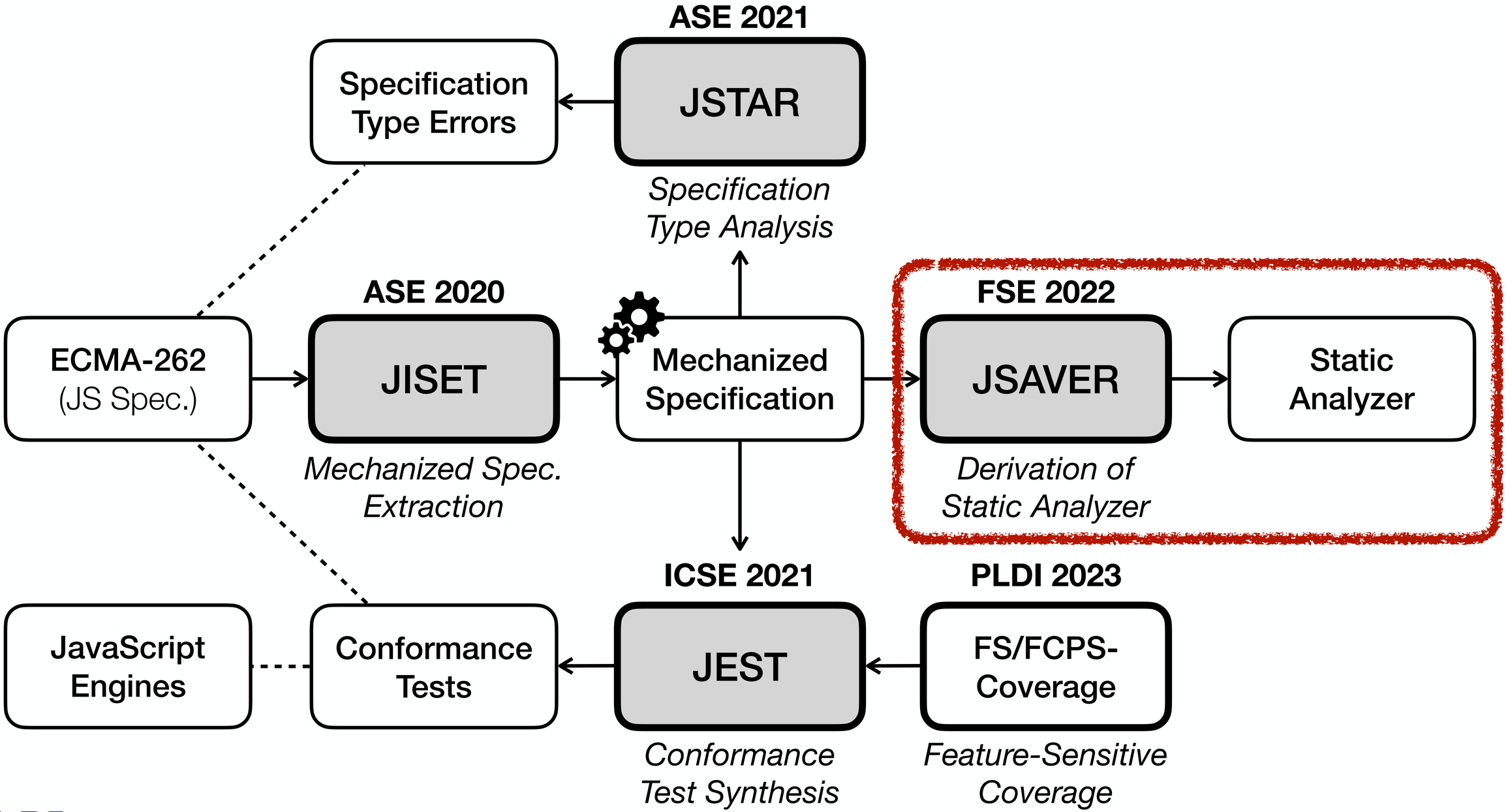


Effectiveness of k-FS / k-FCPS Coverage Criteria

Coverage Criteria $C_{\mathbb{G}}$	# Syn. Test	# Bug
0-FS node-or-branch (0-fs)	2,111	55
1-FS node-or-branch (1-fs)	6,766	83
1-FCPS node-or-branch (1-fcps)	9,092	87
2-FS node-or-branch (2-fs)	97,423	102
2-FCPS node-or-branch (2-fcps)	122,589	111



[FSE 2022] J. Park et al., “Automatically Deriving JavaScript Static Analyzers from Specifications using Meta-level Static Analysis”



Meta-Level Static Analysis

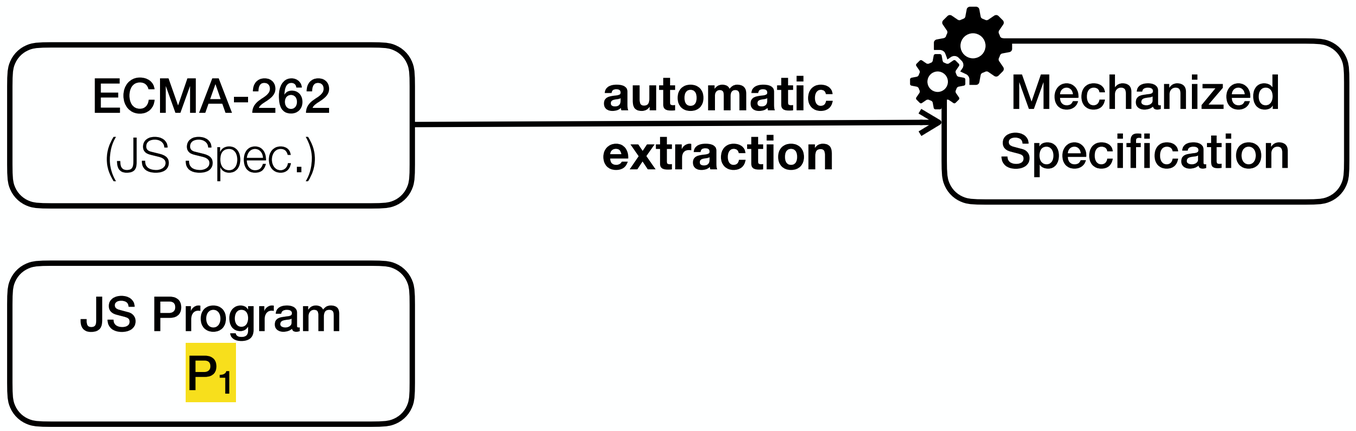
How to perform **static analysis** on **JavaScript programs** using **language specification**?

ECMA-262
(JS Spec.)

JS Program
P_1

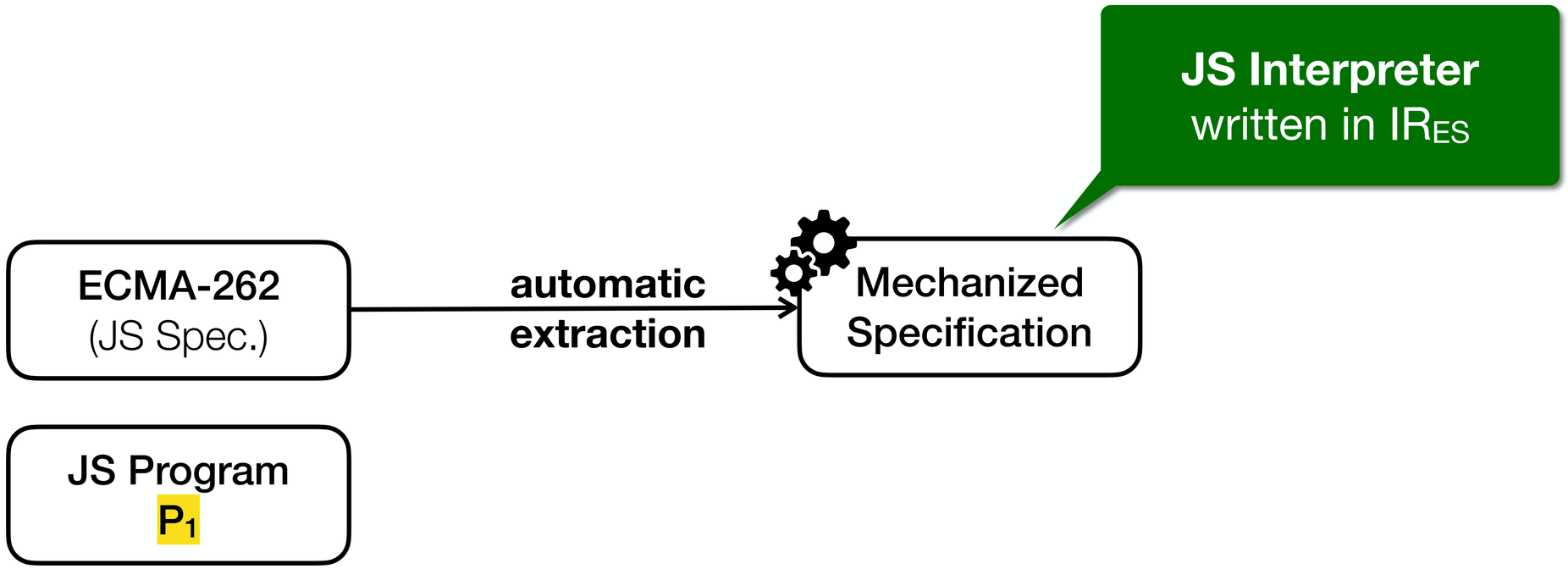
Meta-Level Static Analysis

How to perform **static analysis** on **JavaScript programs** using **language specification**?



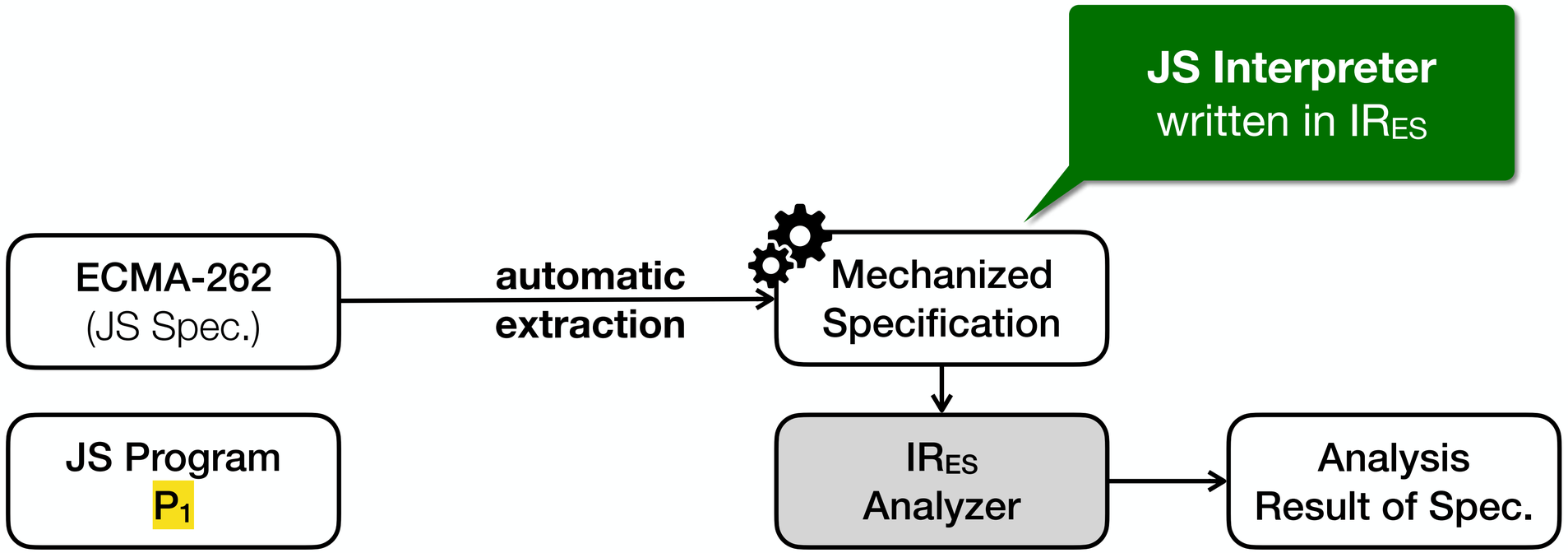
Meta-Level Static Analysis

How to perform **static analysis** on **JavaScript programs** using **language specification**?



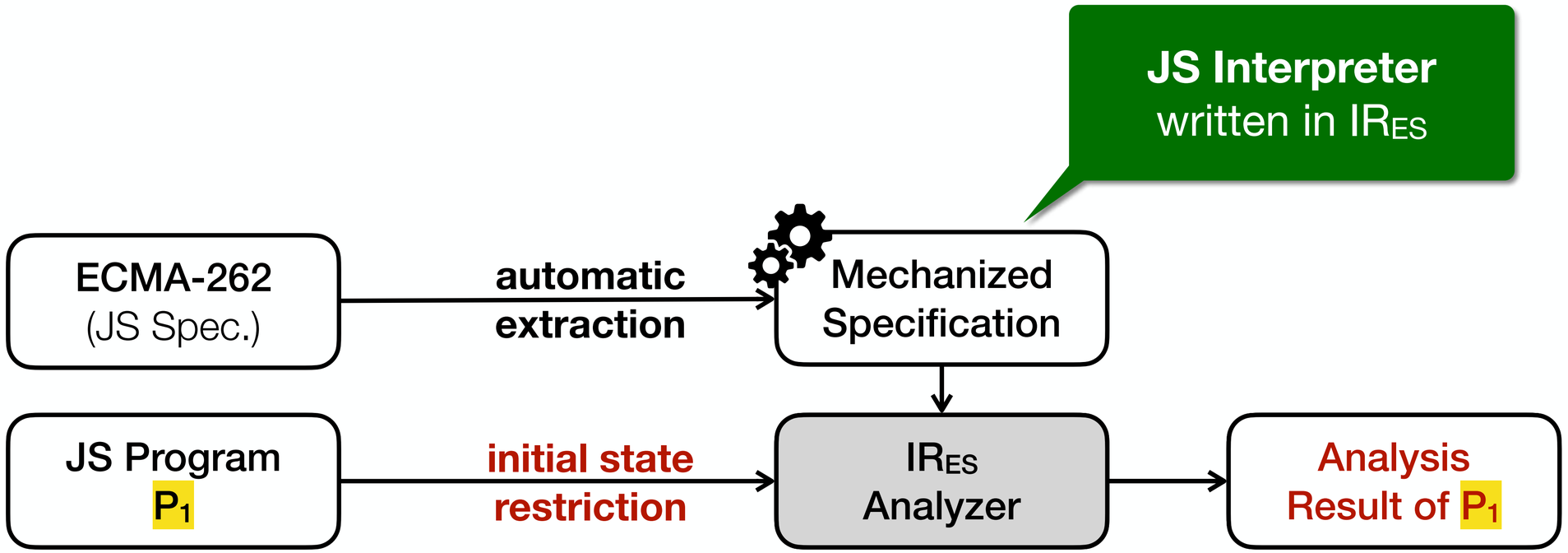
Meta-Level Static Analysis

How to perform **static analysis** on **JavaScript programs** using **language specification**?



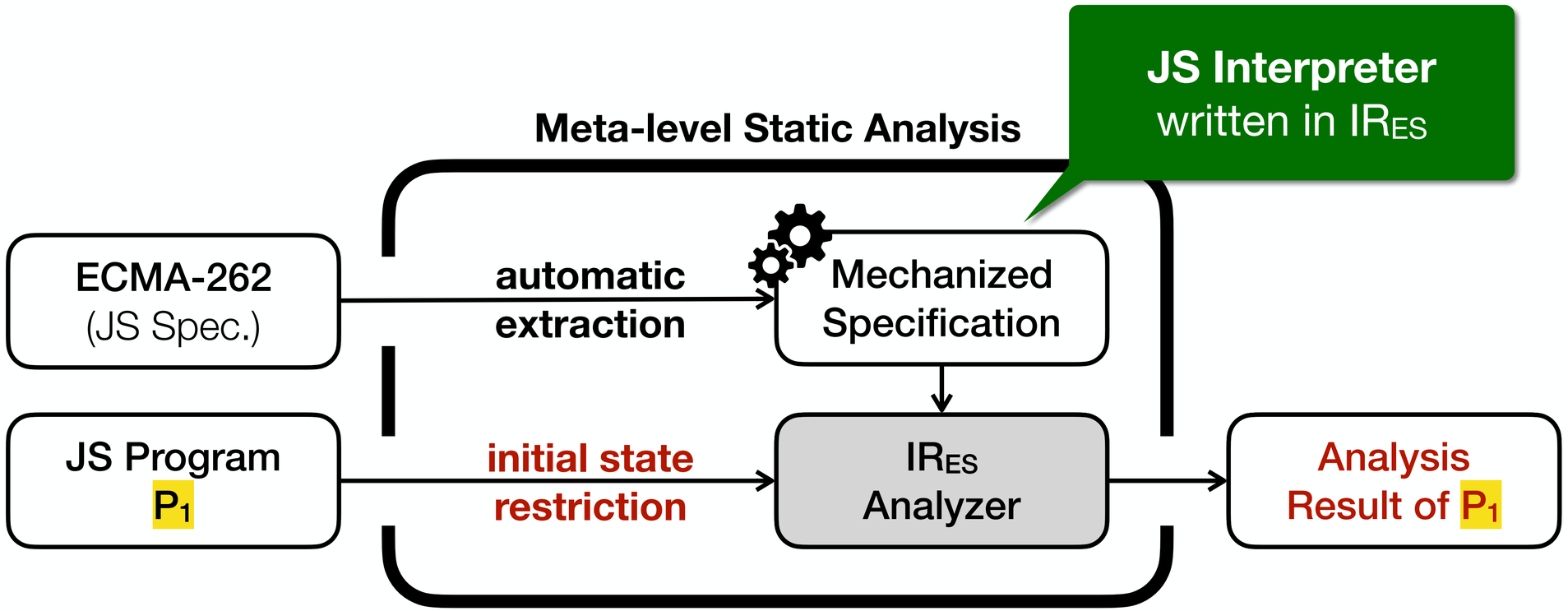
Meta-Level Static Analysis

How to perform **static analysis** on **JavaScript programs** using **language specification**?



Meta-Level Static Analysis

How to perform **static analysis** on **JavaScript** **programs** using **language specification**?



Meta-Level Static Analysis

JS Program P_1

```
x ||= y
```

JavaScript

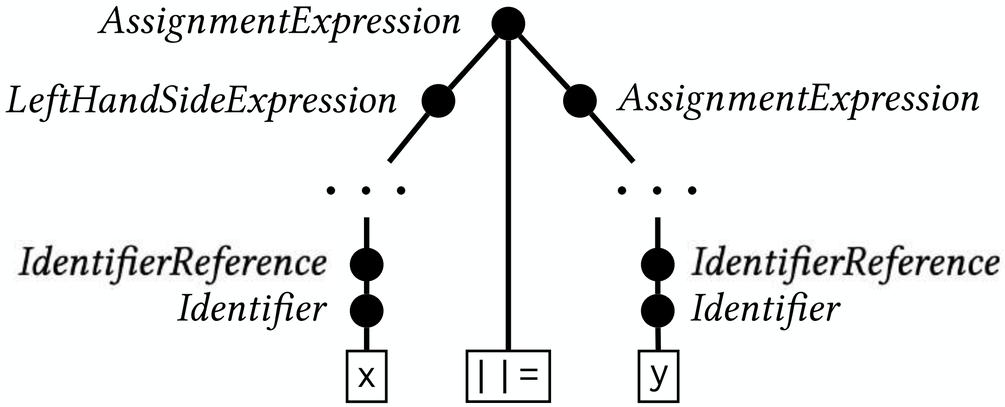
IR_{ES}

Meta-Level Static Analysis

JS Program P_1

```
x ||= y
```

parse



JavaScript

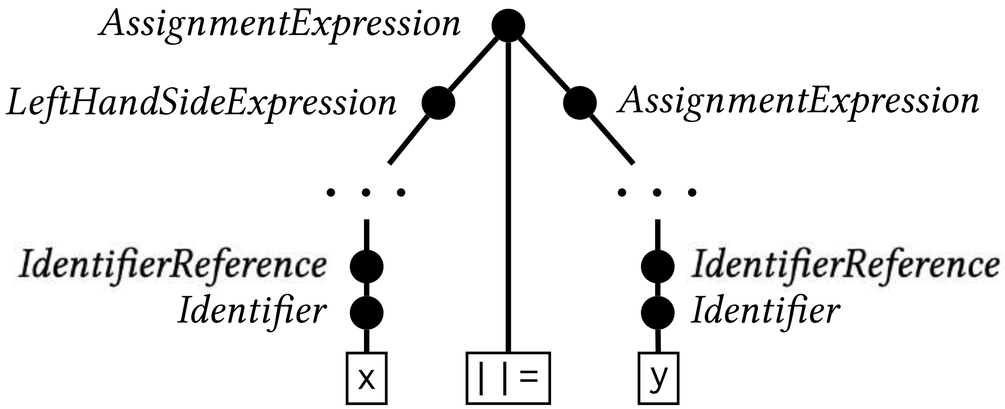
IR_{ES}

Meta-Level Static Analysis

JS Program P_1

```
x ||= y
```

parse



JavaScript

IR$_{ES}$

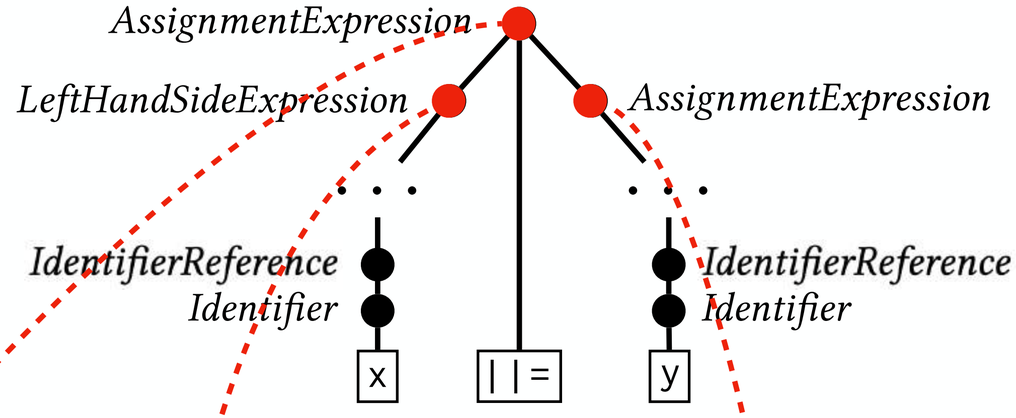
```
syntax def AssignmentExpression[8].Evaluation(
  this, LeftHandSideExpression, AssignmentExpression
) {
  let lref = (LeftHandSideExpression.Evaluation)
  let lval = [? (GetValue lref)]
  let lbool = [! (ToBoolean lval)]
  if (= lbool true) return lval
  ...
}
```

Meta-Level Static Analysis

JS Program P_1

```
x ||= y
```

parse



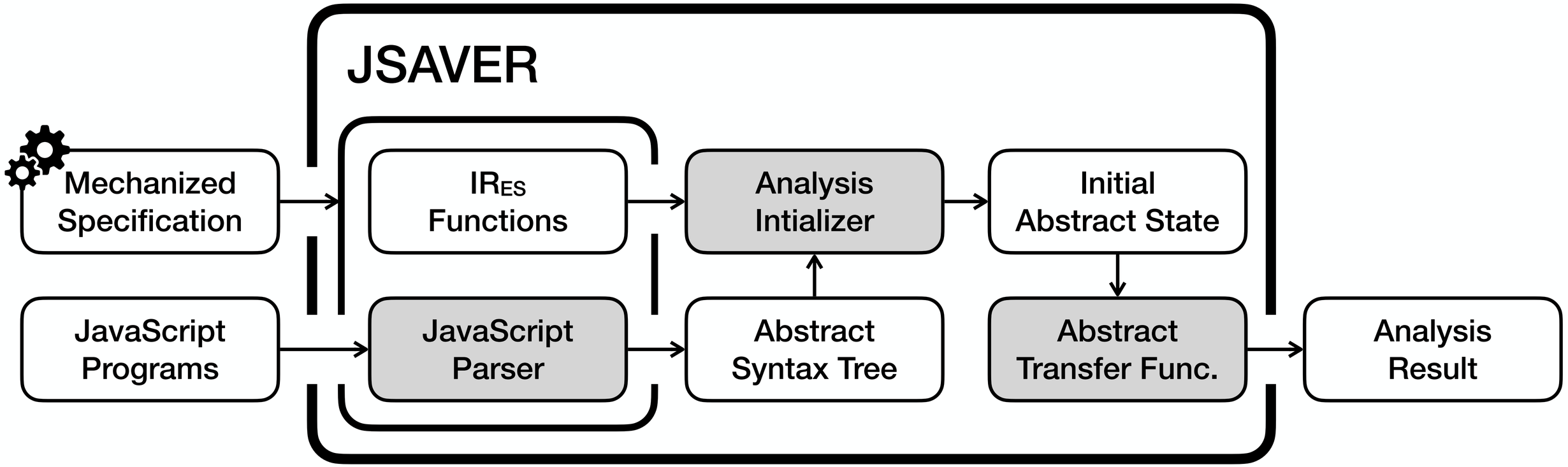
JavaScript

IR$_{ES}$

```
syntax def AssignmentExpression[8].Evaluation(
  this, LeftHandSideExpression, AssignmentExpression
) {
  let lref = (LeftHandSideExpression.Evaluation)
  let lval = [? (GetValue lref)]
  let lbool = [! (ToBoolean lval)]
  if (= lbool true) return lval
  ...
}
```

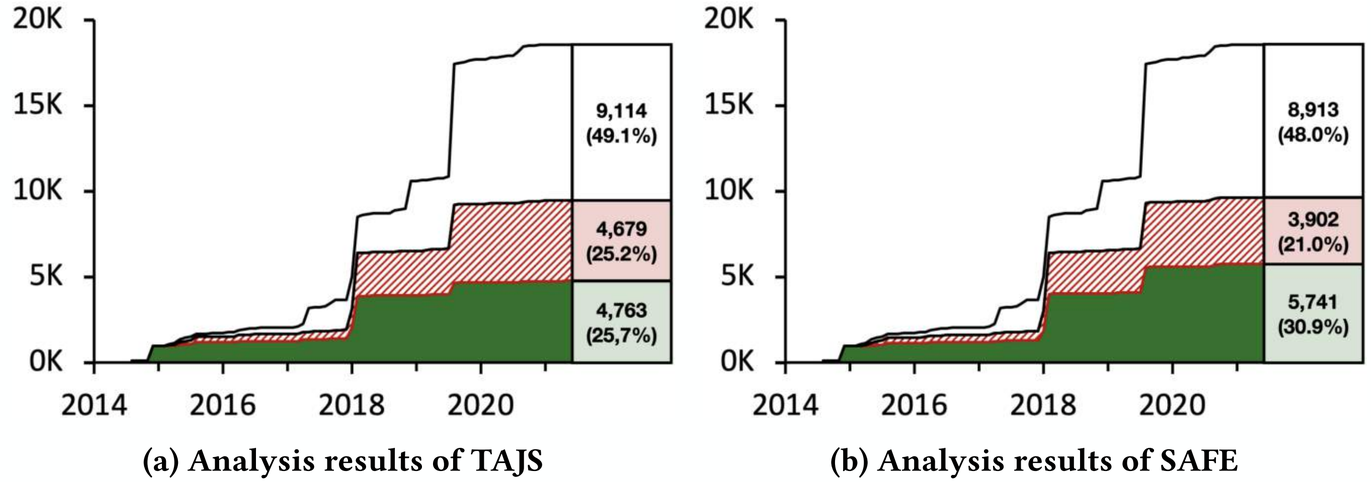
JSAVER

(**J**avaScript **S**tatic **A**nalyzer **v**ia **E**CMAScript **R**epresentation)



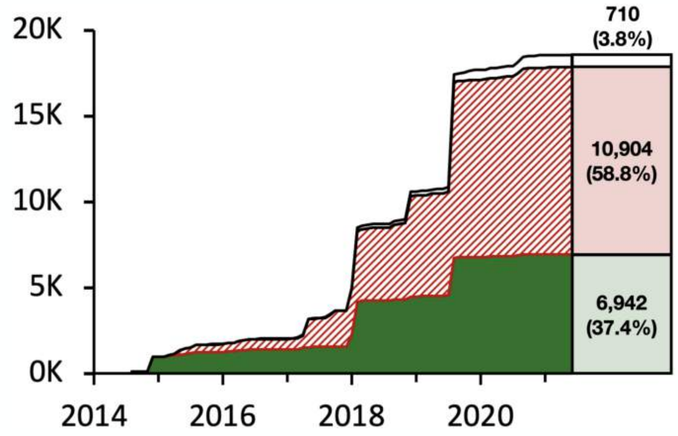
JSAVER - Soundness

Manual

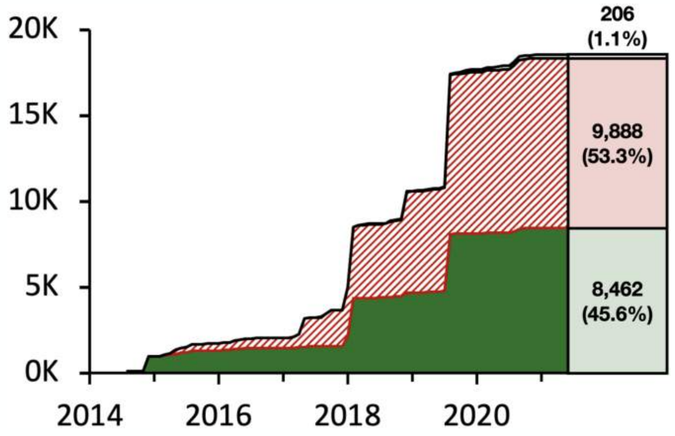


(a) Analysis results of TAJS

(b) Analysis results of SAFE

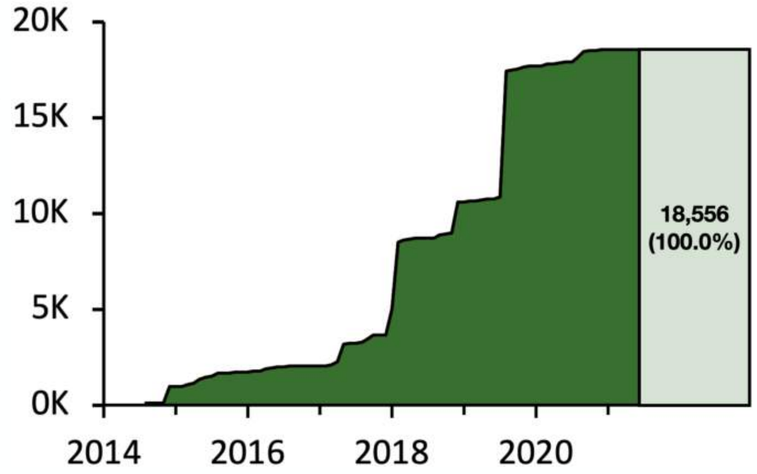


(d) Analysis results of TAJS with Babel



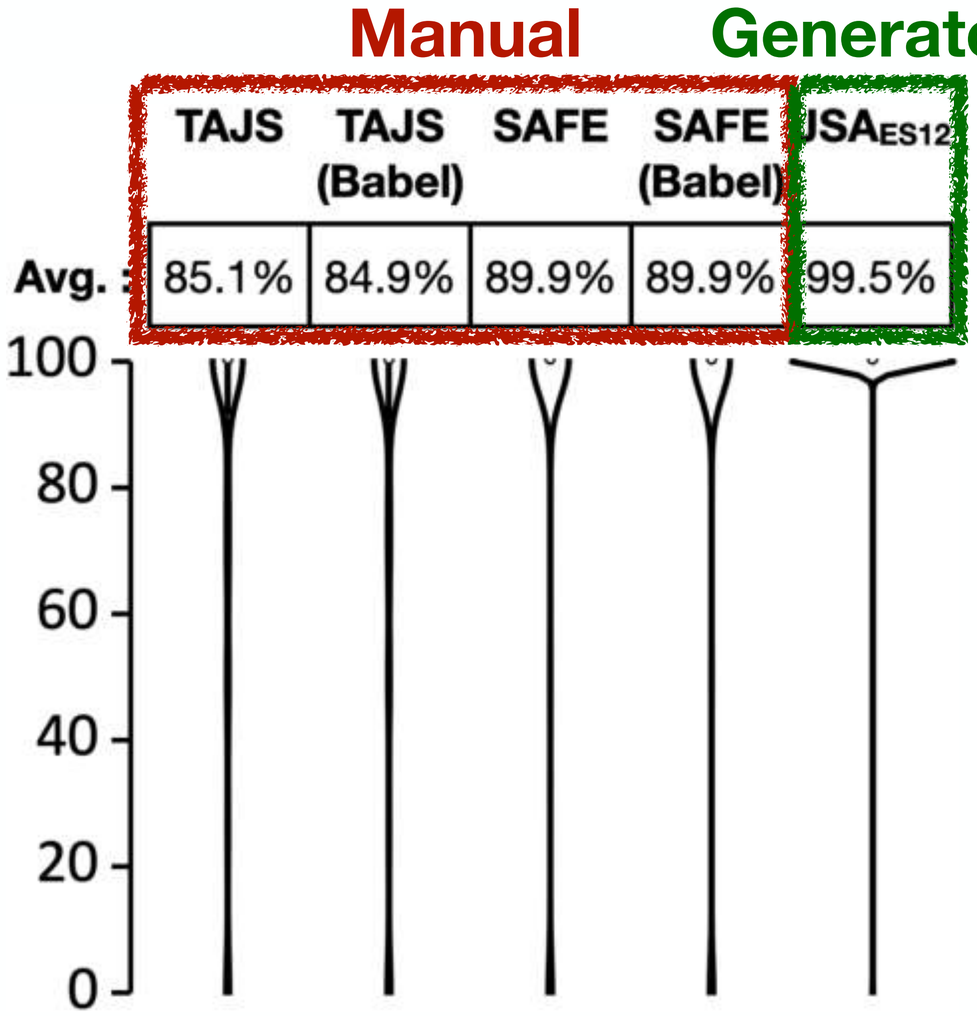
(e) Analysis results of SAFE with Babel

Generated

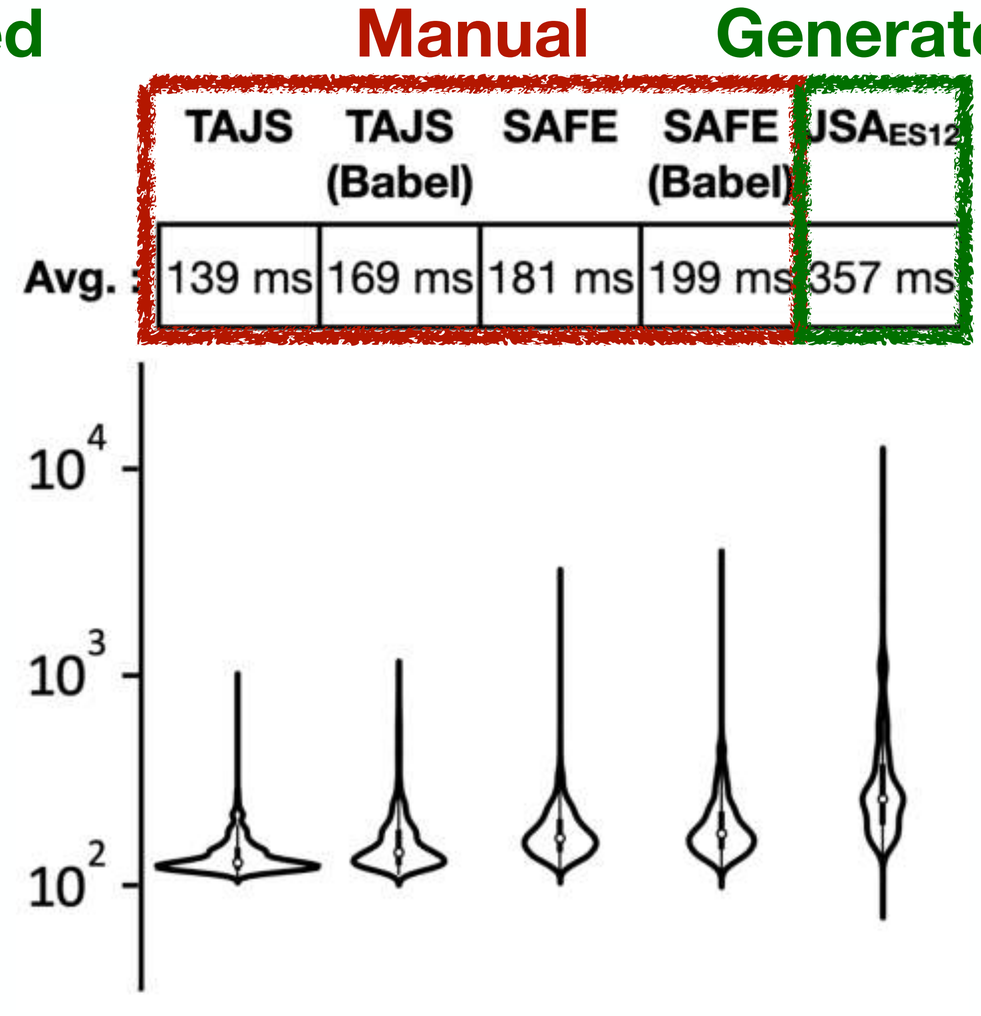


(c) Analysis results of JSA_{ES12}

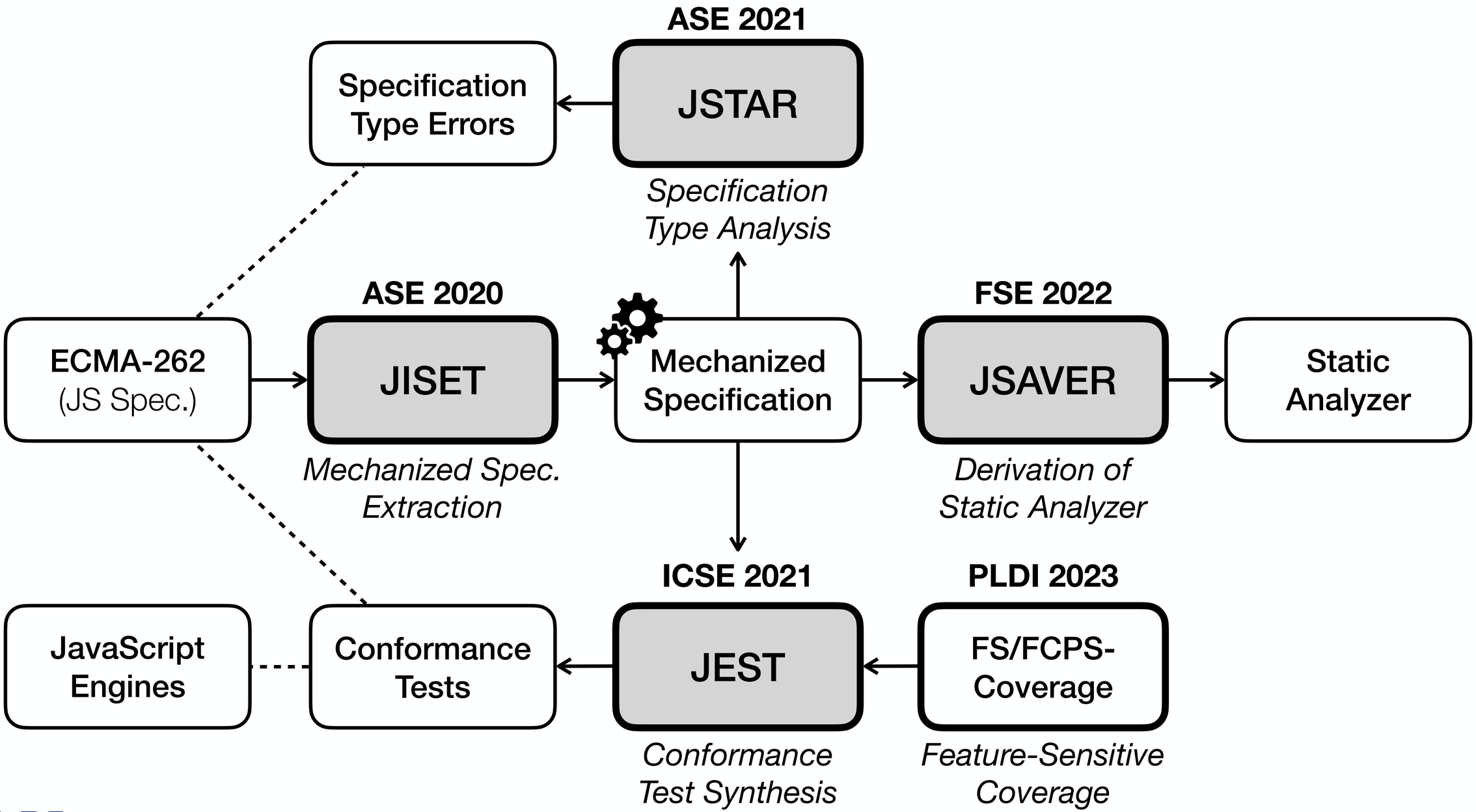
JSAVER - Precision vs Performance

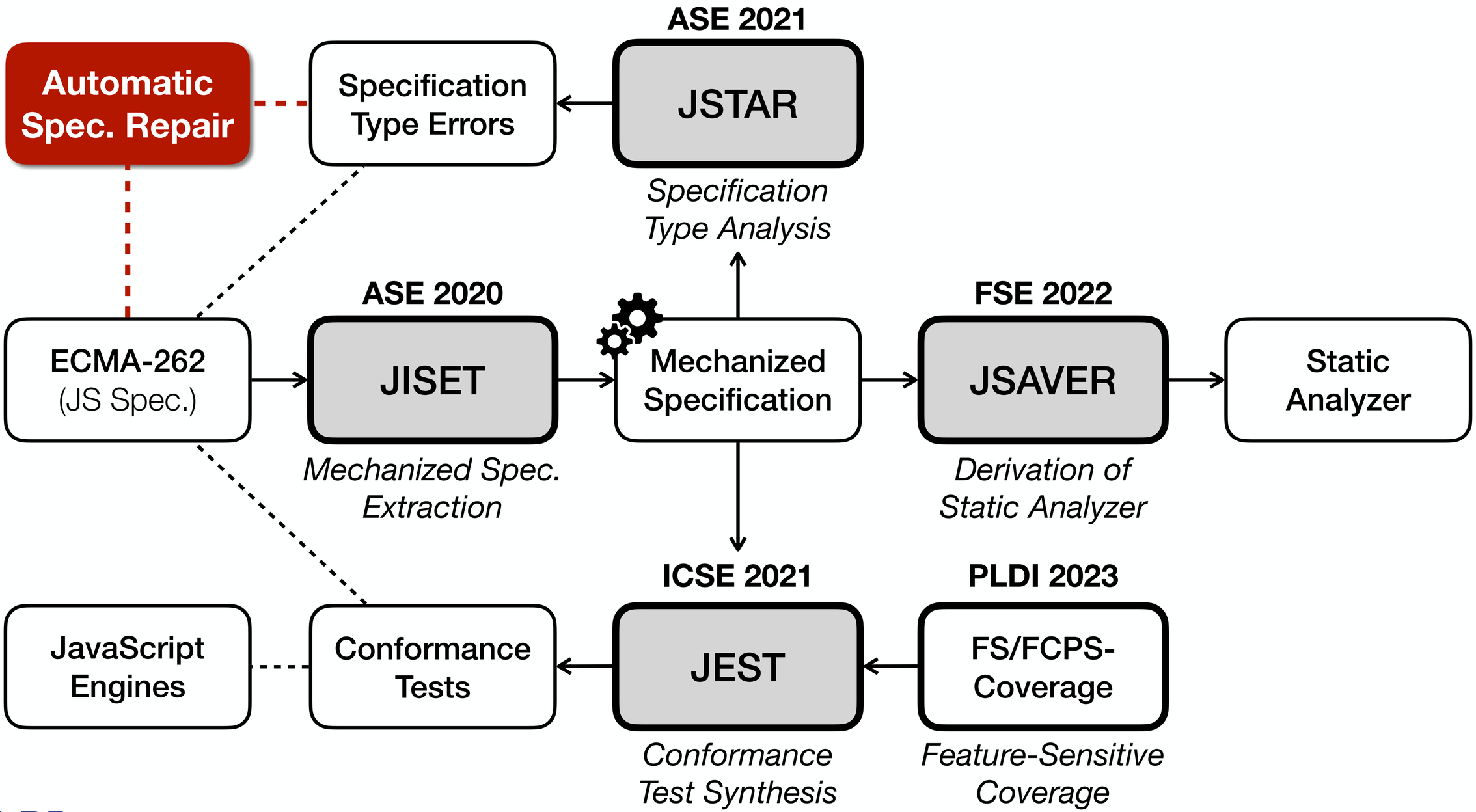


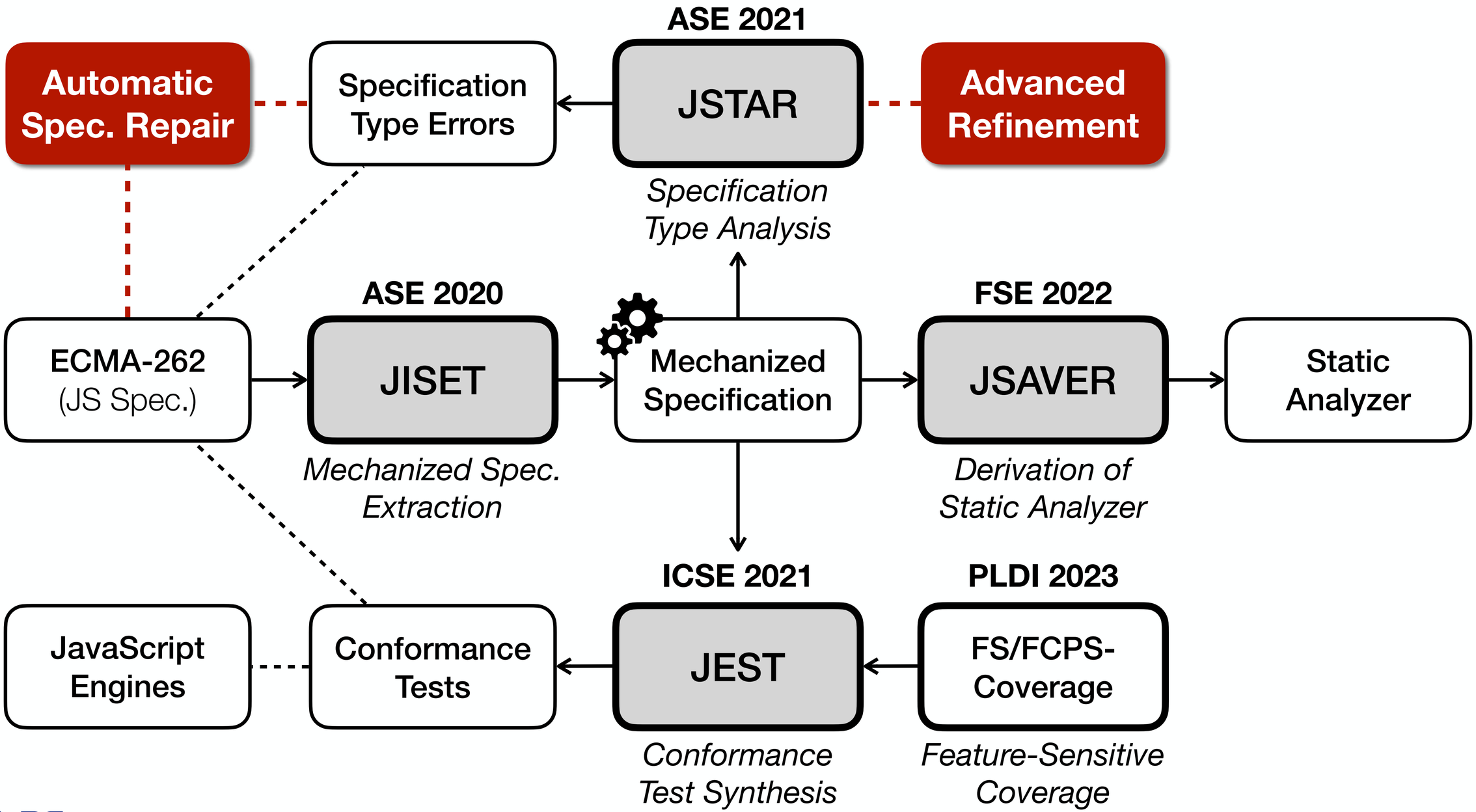
(a) The analysis precision

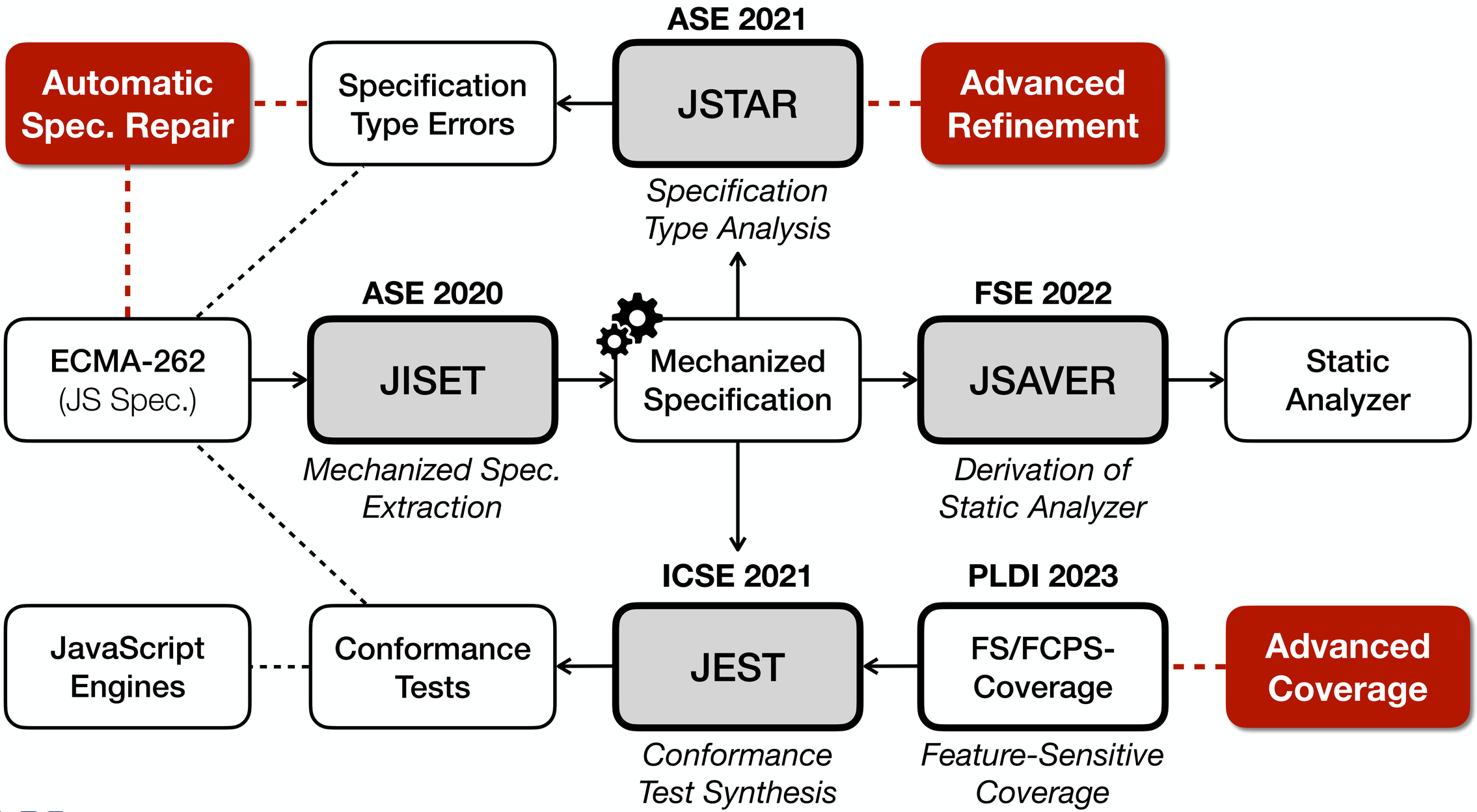


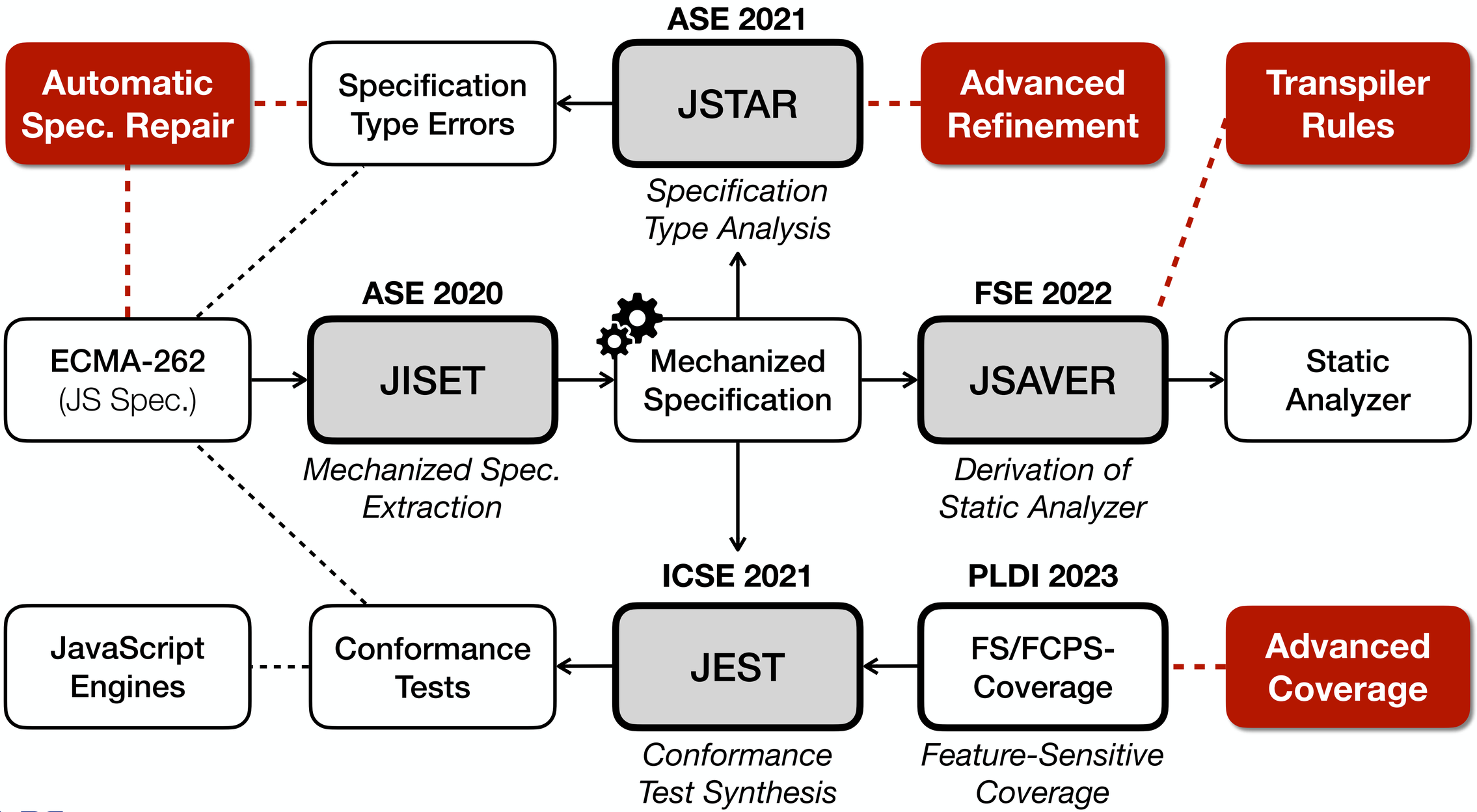
(b) The analysis performance

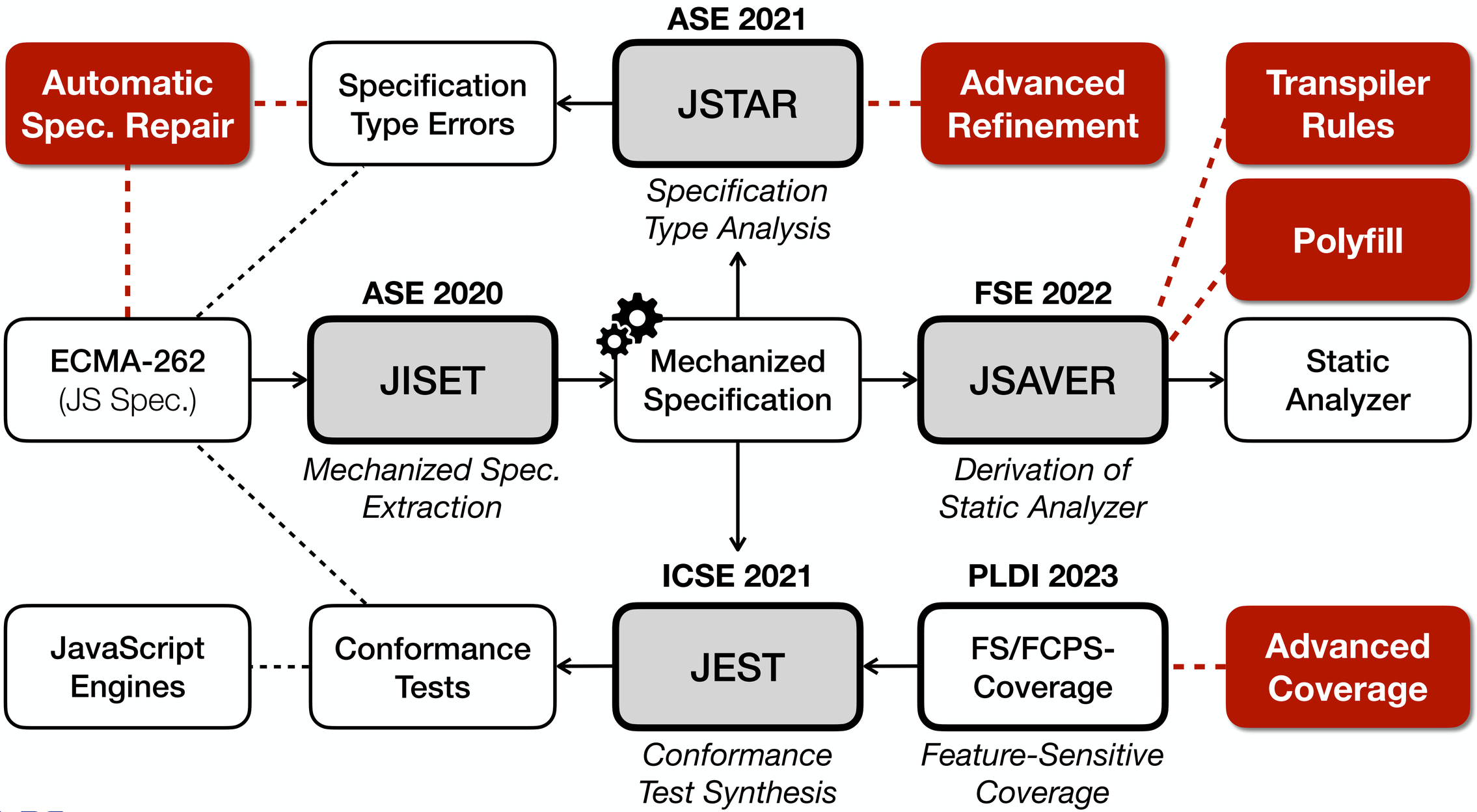

ASE 2021
Specification Type Errors
JSTAR
Specification Type Analysis
ASE 2020
FSE 2022
ECMA-262 (JS Spec.)
JISET
Mechanized Specification
JSAVER
Static Analyzer
Mechanized Spec. Extraction
Derivation of Static Analyzer
ICSE 2021
PLDI 2023
JavaScript Engines
Conformance Tests
JEST
FS/FCPS-Coverage
Conformance Test Synthesis
Feature-Sensitive Coverage

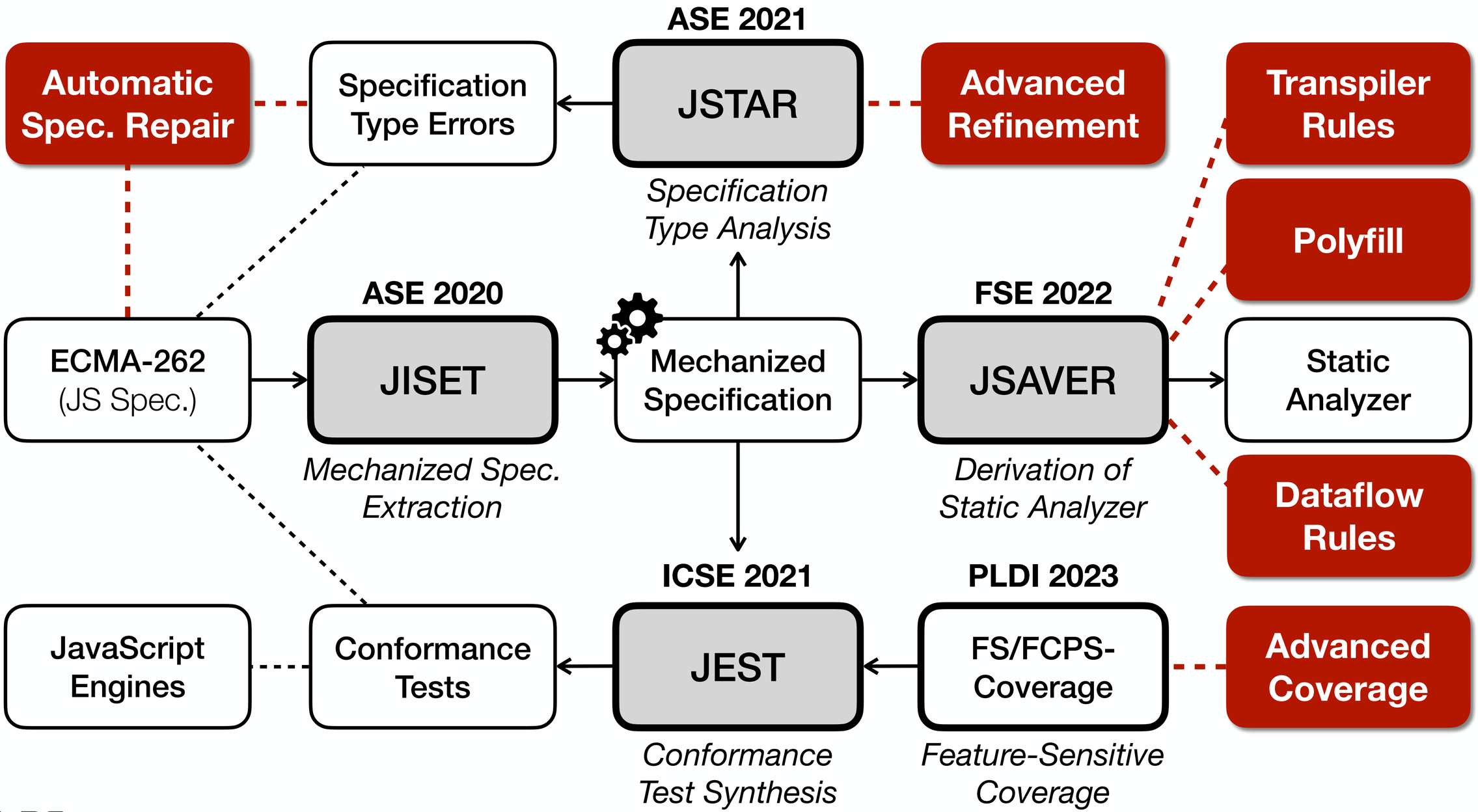

ASE 2021
Automatic Spec. Repair
Specification Type Errors
JSTAR
Specification Type Analysis
ASE 2020
FSE 2022
ECMA-262 (JS Spec.)
JISET
Mechanized Specification
JSAVER
Static Analyzer
Mechanized Spec. Extraction
Derivation of Static Analyzer
ICSE 2021
PLDI 2023
JavaScript Engines
Conformance Tests
JEST
FS/FCPS-Coverage
Conformance Test Synthesis
Feature-Sensitive Coverage

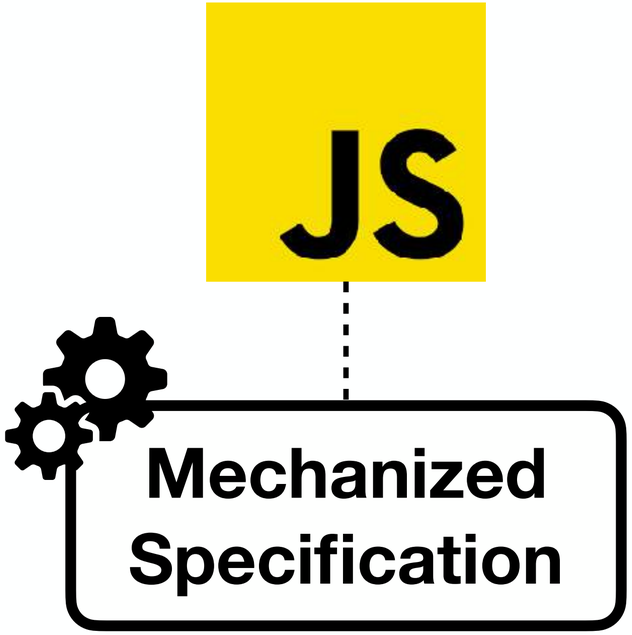

ASE 2021
Automatic Spec. Repair
Specification Type Errors
JSTAR
Advanced Refinement
Specification Type Analysis
ASE 2020
FSE 2022
ECMA-262 (JS Spec.)
JISET
Mechanized Specification
JSAVER
Static Analyzer
Mechanized Spec. Extraction
Derivation of Static Analyzer
ICSE 2021
PLDI 2023
JavaScript Engines
Conformance Tests
JEST
FS/FCPS-Coverage
Conformance Test Synthesis
Feature-Sensitive Coverage

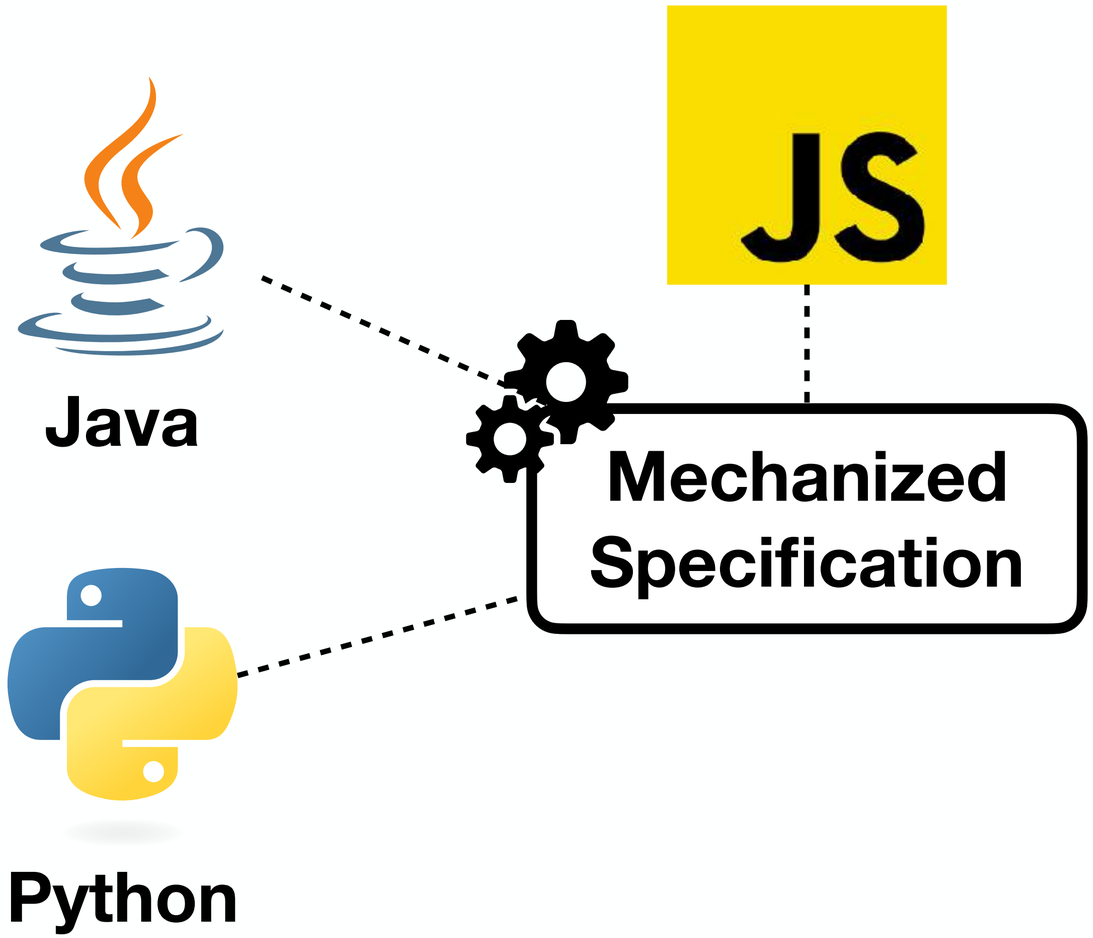

ASE 2021
Automatic Spec. Repair
Specification Type Errors
JSTAR
Advanced Refinement
Specification Type Analysis
ASE 2020
FSE 2022
ECMA-262 (JS Spec.)
JISET
Mechanized Specification
JSAVER
Static Analyzer
Mechanized Spec. Extraction
Derivation of Static Analyzer
ICSE 2021
PLDI 2023
JavaScript Engines
Conformance Tests
JEST
FS/FCPS-Coverage
Advanced Coverage
Conformance Test Synthesis
Feature-Sensitive Coverage

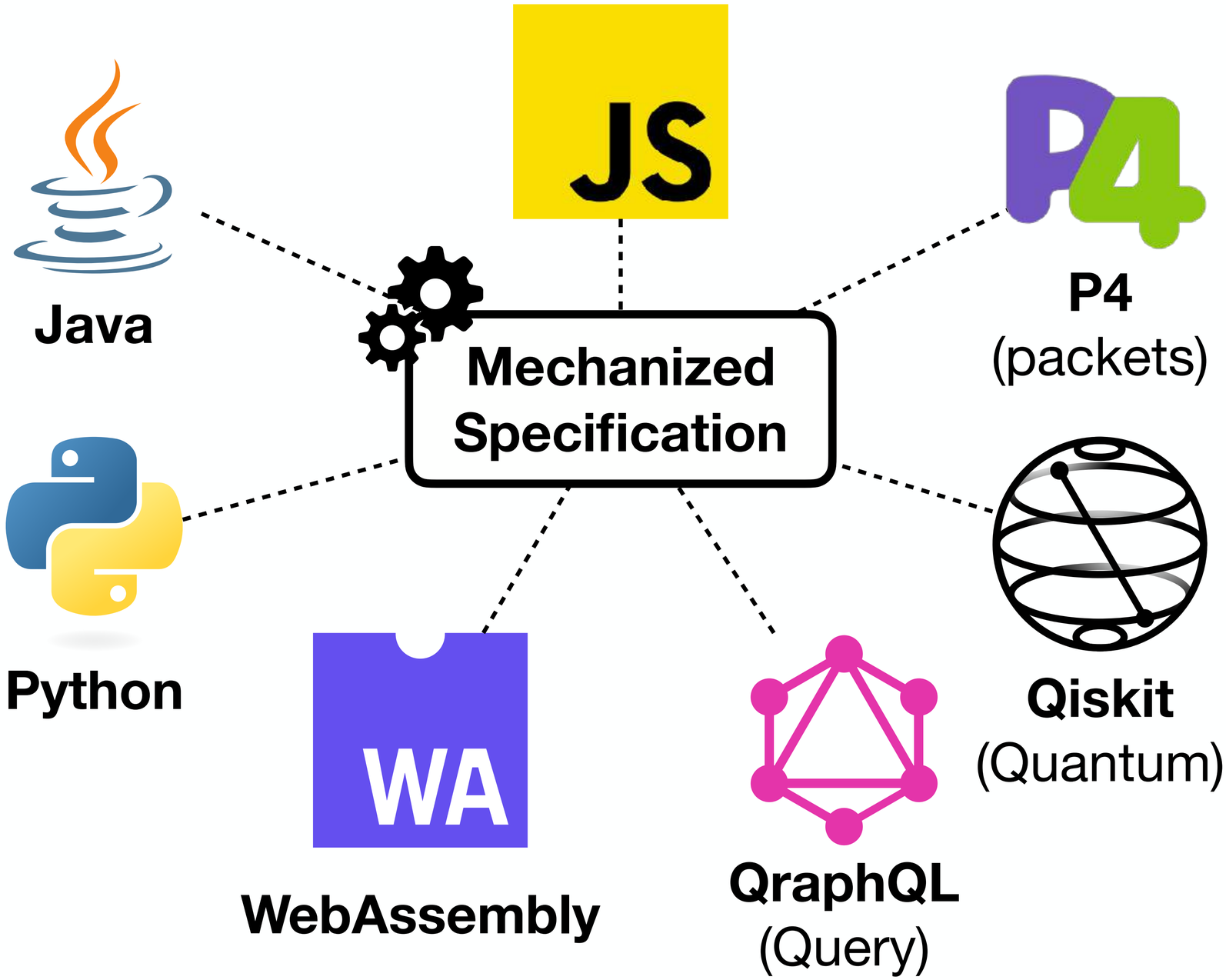

ASE 2021
Automatic Spec. Repair
Specification Type Errors
JSTAR
Advanced Refinement
Transpiler Rules
Specification Type Analysis
ASE 2020
FSE 2022
ECMA-262 (JS Spec.)
JISET
Mechanized Specification
JSAVER
Static Analyzer
Mechanized Spec. Extraction
Derivation of Static Analyzer
ICSE 2021
PLDI 2023
JavaScript Engines
Conformance Tests
JEST
FS/FCPS-Coverage
Advanced Coverage
Conformance Test Synthesis
Feature-Sensitive Coverage


ASE 2021
Automatic Spec. Repair
Specification Type Errors
JSTAR
Advanced Refinement
Transpiler Rules
Specification Type Analysis
Polyfill
ASE 2020
FSE 2022
ECMA-262 (JS Spec.)
JISET
Mechanized Specification
JSAVER
Static Analyzer
Mechanized Spec. Extraction
Derivation of Static Analyzer
ICSE 2021
PLDI 2023
JavaScript Engines
Conformance Tests
JEST
FS/FCPS-Coverage
Advanced Coverage
Conformance Test Synthesis
Feature-Sensitive Coverage

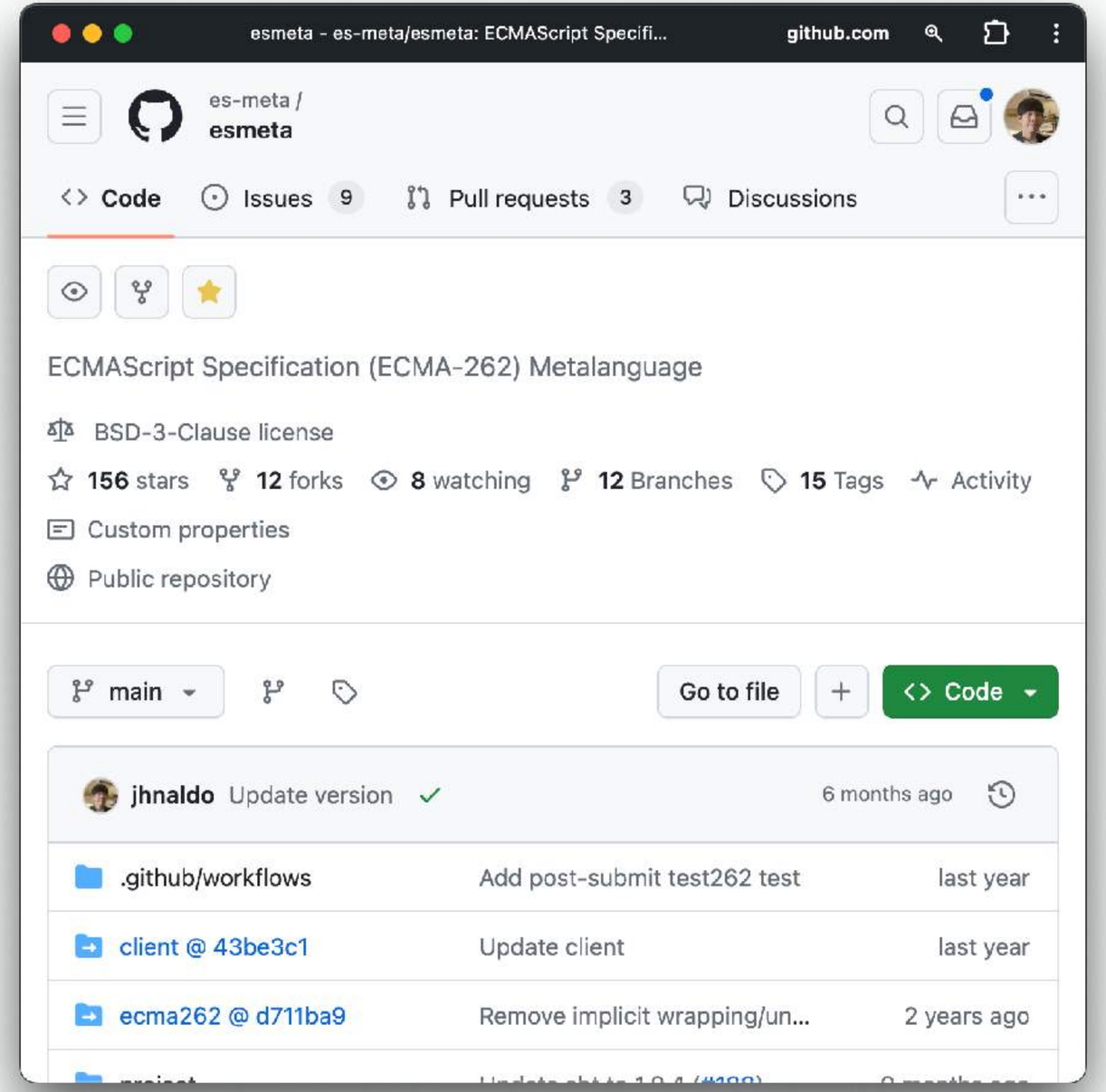

ASE 2021
JSTAR
Specification Type Errors
Automatic Spec. Repair
Advanced Refinement
Transpiler Rules
Specification Type Analysis
Polyfill
ASE 2020
FSE 2022
ECMA-262 (JS Spec.)
JISET
Mechanized Specification
JSAVER
Static Analyzer
Mechanized Spec. Extraction
Derivation of Static Analyzer
Dataflow Rules
ICSE 2021
PLDI 2023
JavaScript Engines
Conformance Tests
JEST
FS/FCPS-Coverage
Advanced Coverage
Conformance Test Synthesis
Feature-Sensitive Coverage


JS
Mechanized
Specification


JS
Java
Mechanized
Specification
Python


JS
Java
P4
(packets)
Mechanized
Specification
Python
Qiskit
(Quantum)
WA
WebAssembly
QraphQL
(Query)

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



Official tool used in CI system of ECMA-262 and Test262

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