



JavaScript Static Analysis for Evolving Language Specifications

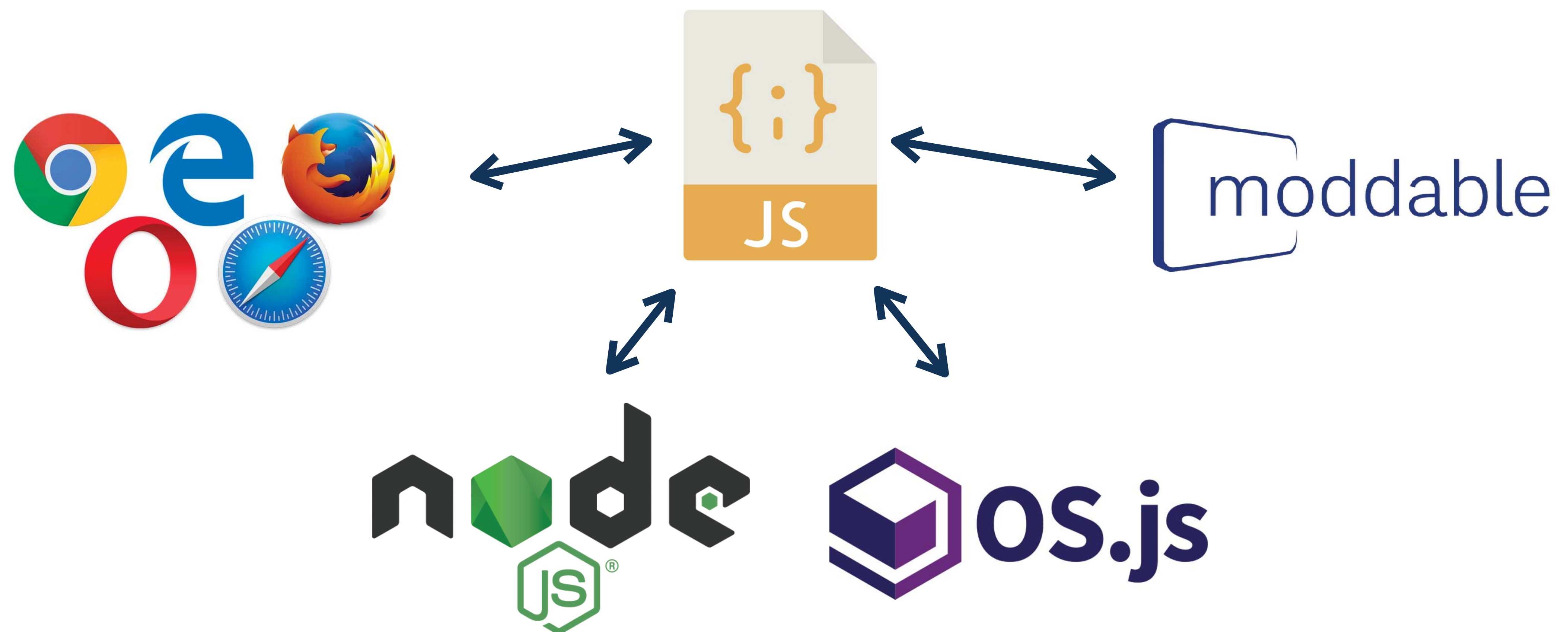
[SW재난연구센터] 겨울정기워크숍

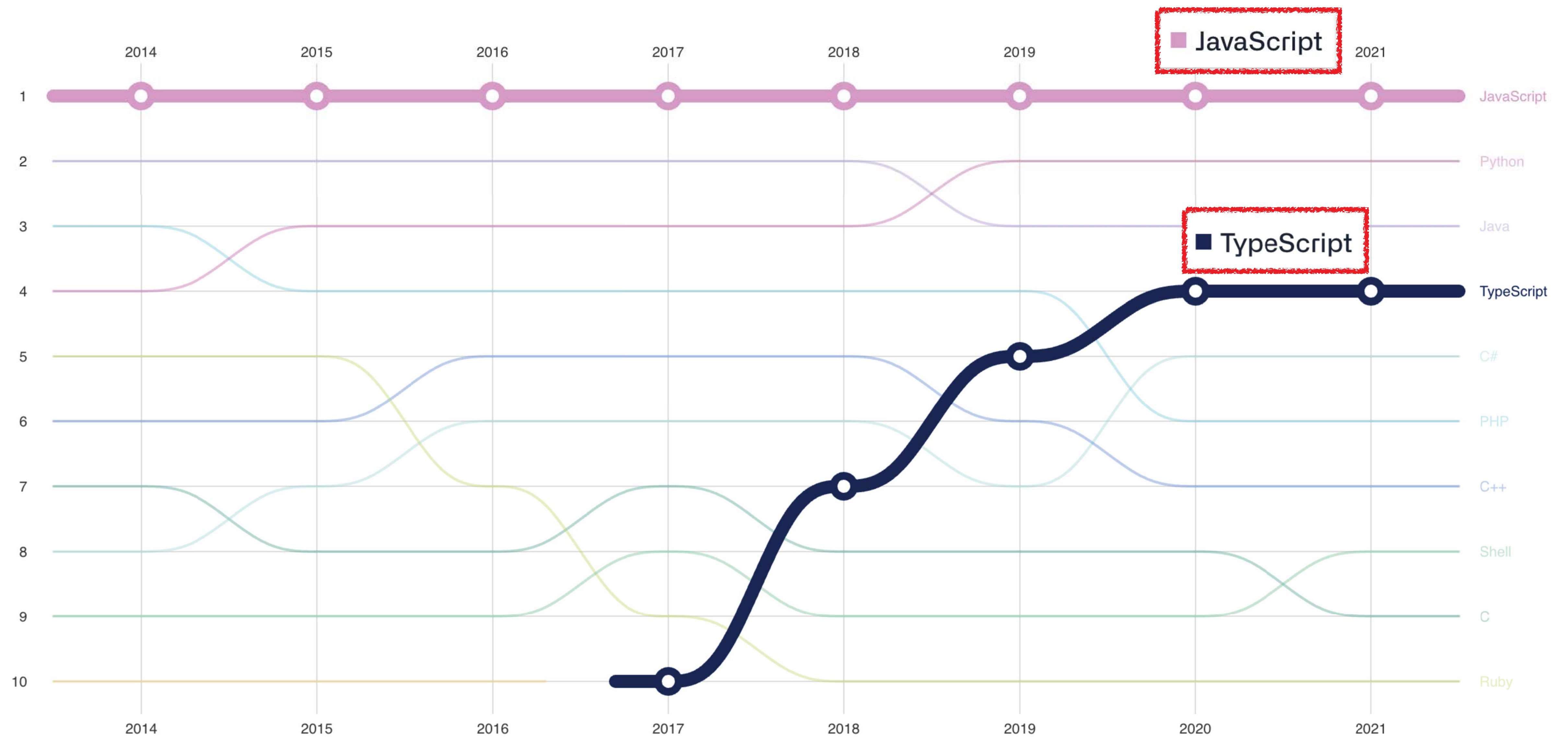
Jihyeok Park

PLRG @ KAIST

February 9, 2022

JavaScript Is Everywhere





<https://octoverse.github.com/>

JavaScript Complex Semantics

```
function f(x) { return x == !x; }
```

Always return **false**?

JavaScript Complex Semantics

```
function f(x) { return x == !x; }
```

Always return **false**?

NO!!

```
f( [] ) -> [] == ![ ]  
         -> [] == false  
         -> +[] == +false  
         -> 0 == 0  
         -> true
```

ECMA-262: ECMAScript Specification



Syntax

```
ArrayLiteral[Yield, Await] :  
  [ Elisionopt ]  
  [ ElementList[?Yield, ?Await] ]  
  [ ElementList[?Yield, ?Await] , Elisionopt ]
```

The production of *ArrayLiteral* in ES12

Semantics

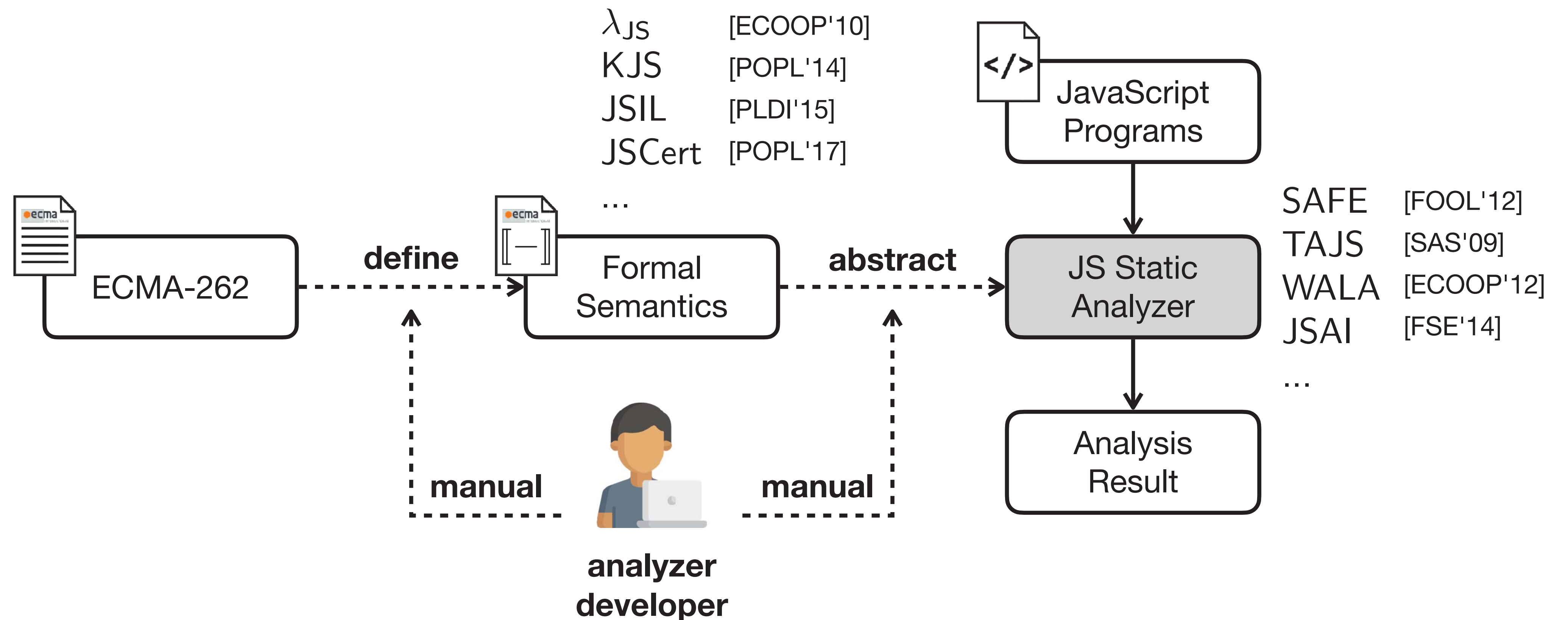
13.2.5.2 Runtime Semantics: Evaluation

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

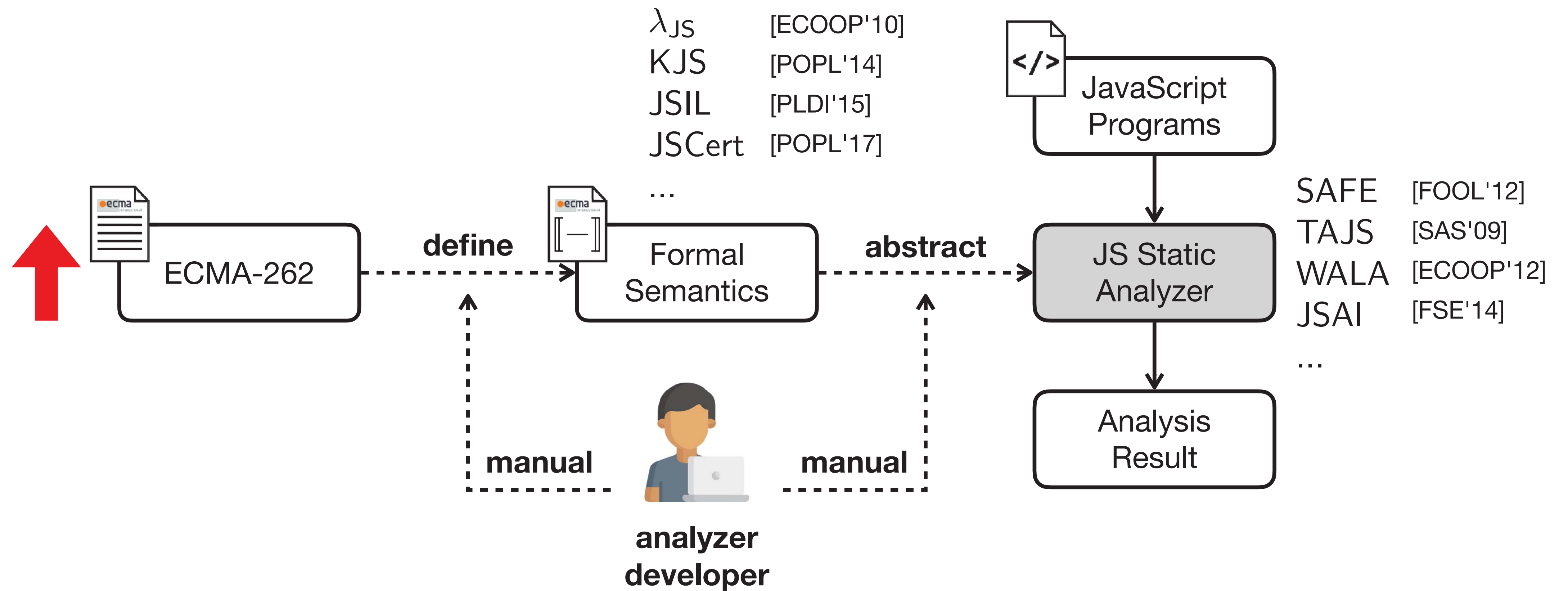
1. Let *array* be ! *ArrayCreate*(0).
2. Let *nextIndex* be the result of performing *ArrayAccumulation* for *ElementList* with arguments *array* and 0.
3. *ReturnIfAbrupt*(*nextIndex*).
4. If *Elision* is present, then
 - a. Let *len* be the result of performing *ArrayAccumulation* for *Elision* with arguments *array* and *nextIndex*.
 - b. *ReturnIfAbrupt*(*len*).
5. Return *array*.

The Evaluation algorithm for
the third alternative of *ArrayLiteral* in ES12

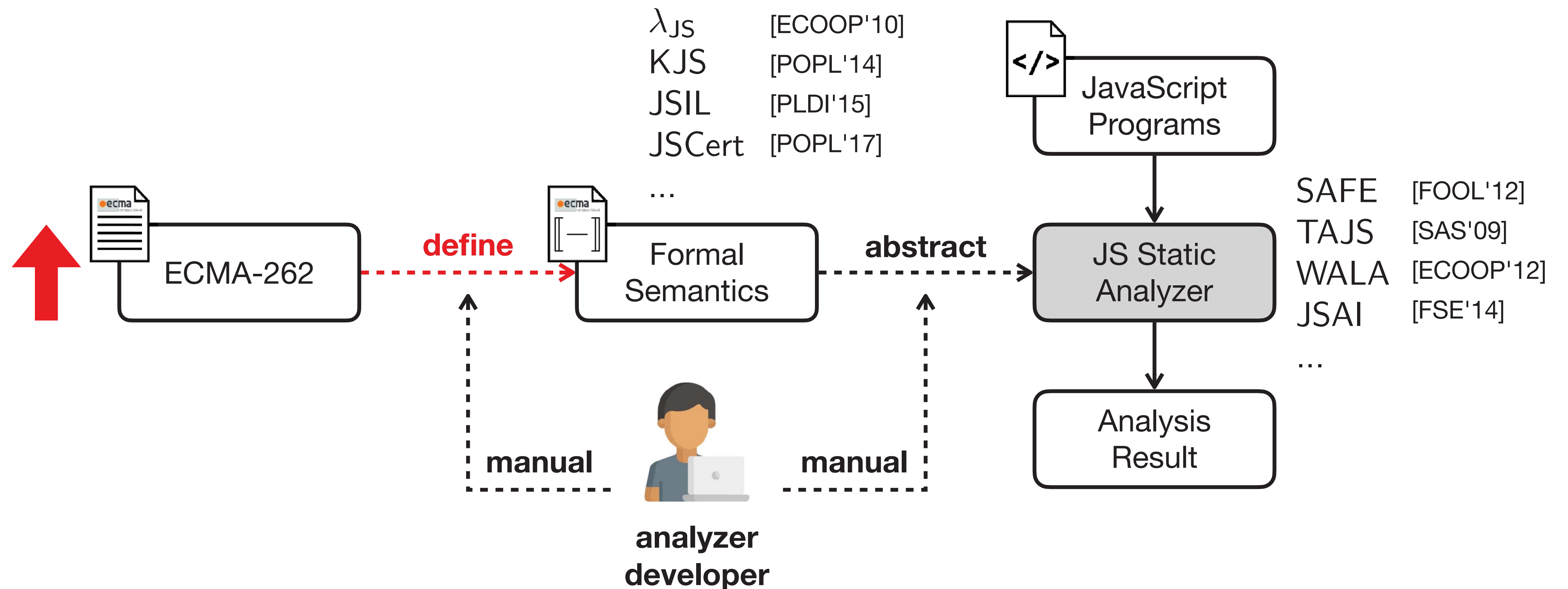
Problem: Manual JavaScript Static Analyzer



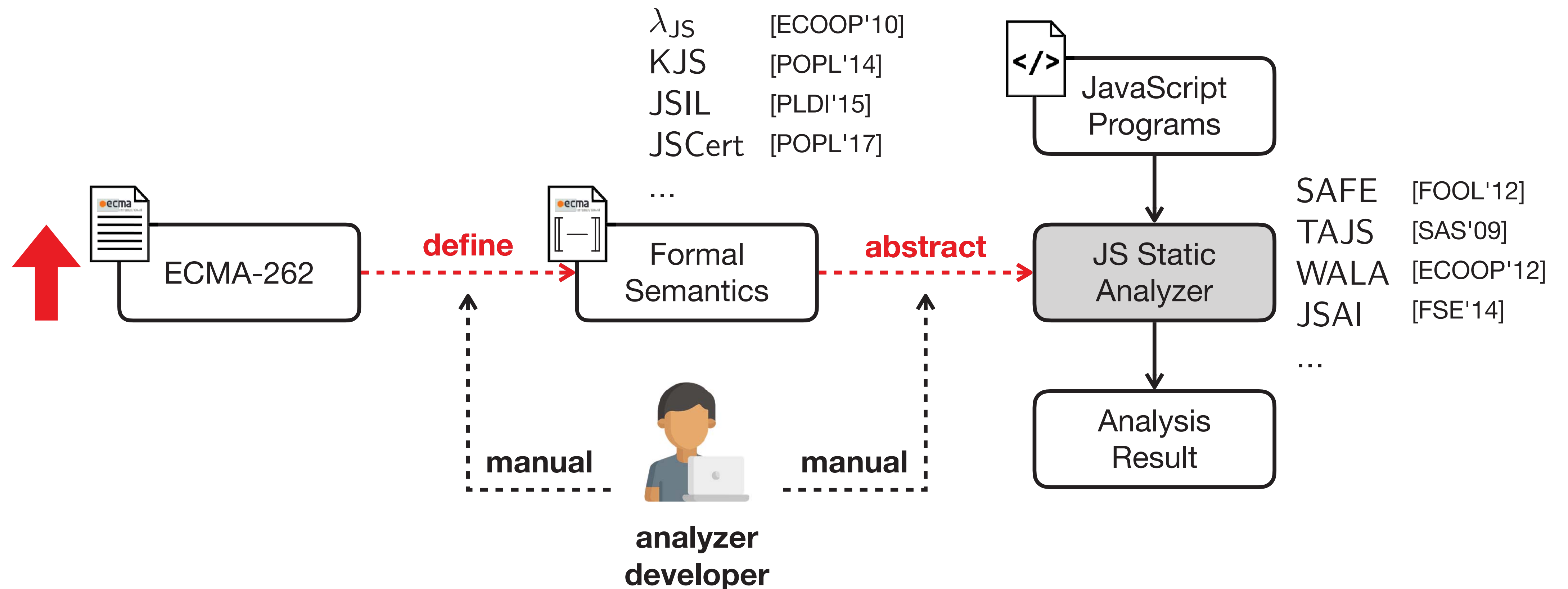
Problem: Manual JavaScript Static Analyzer



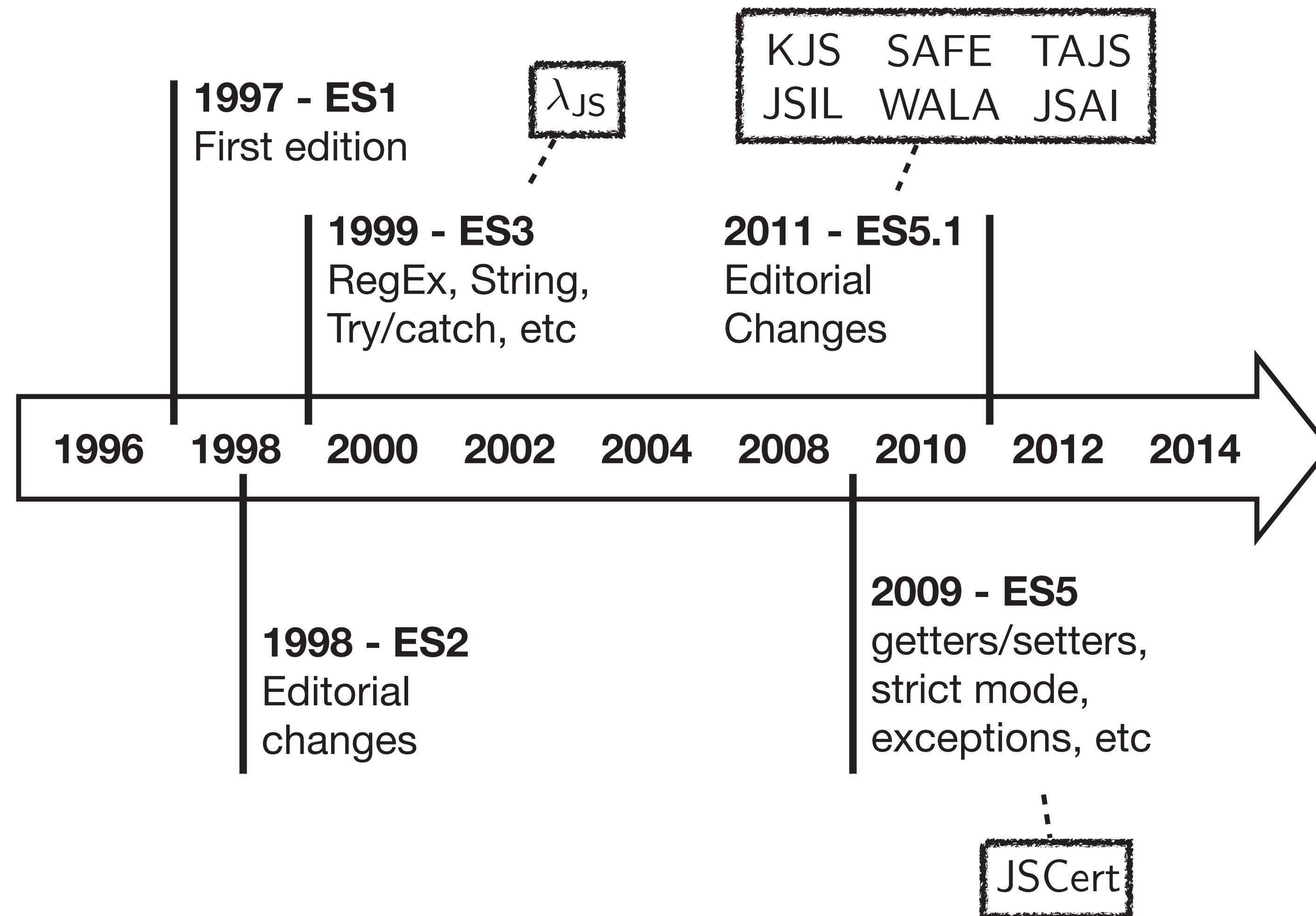
Problem: Manual JavaScript Static Analyzer



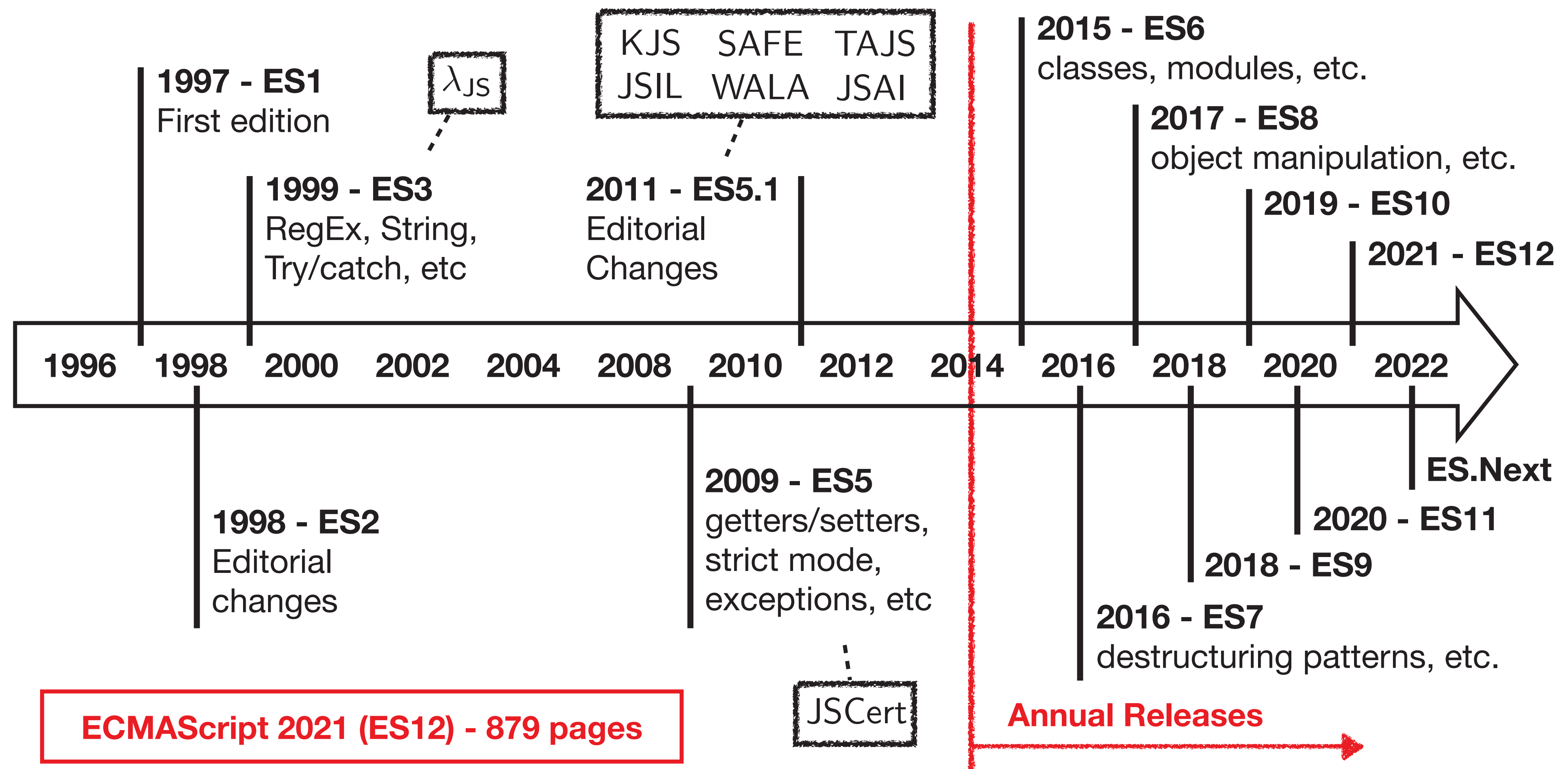
Problem: Manual JavaScript Static Analyzer



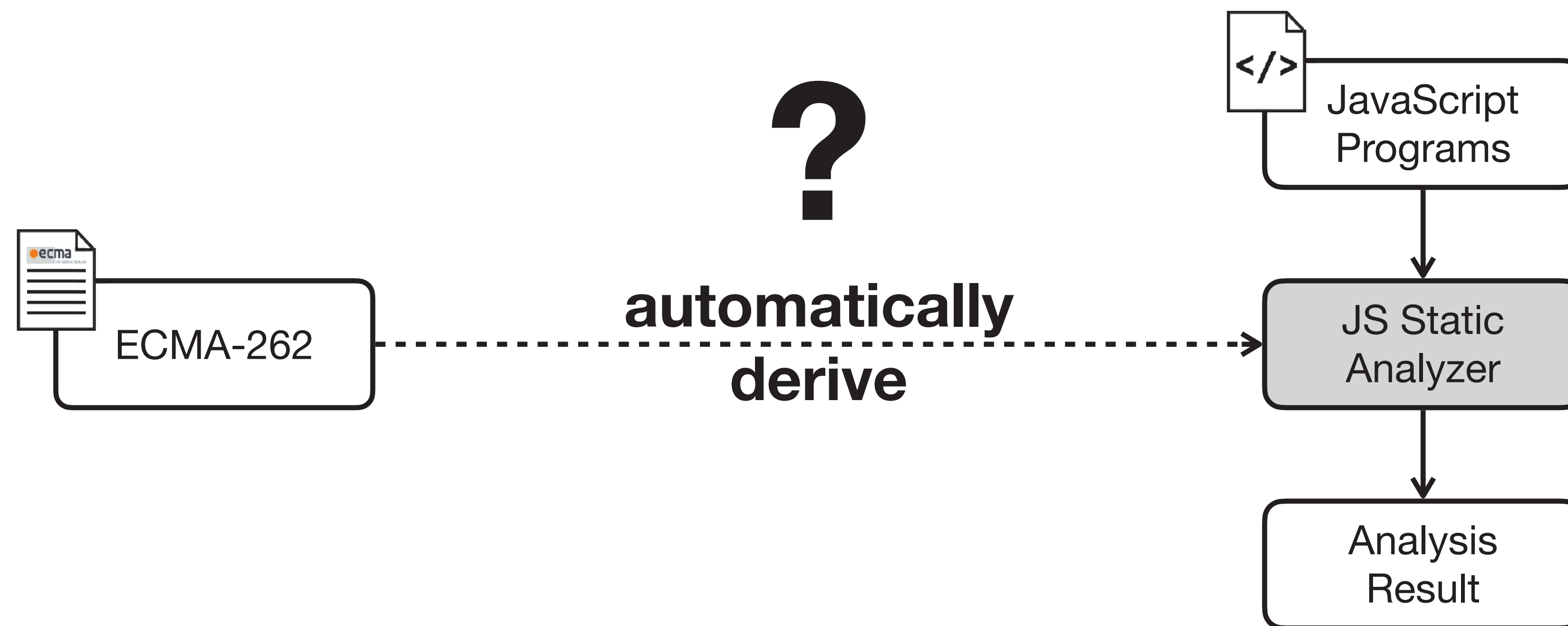
Problem: Fast Evolving JavaScript



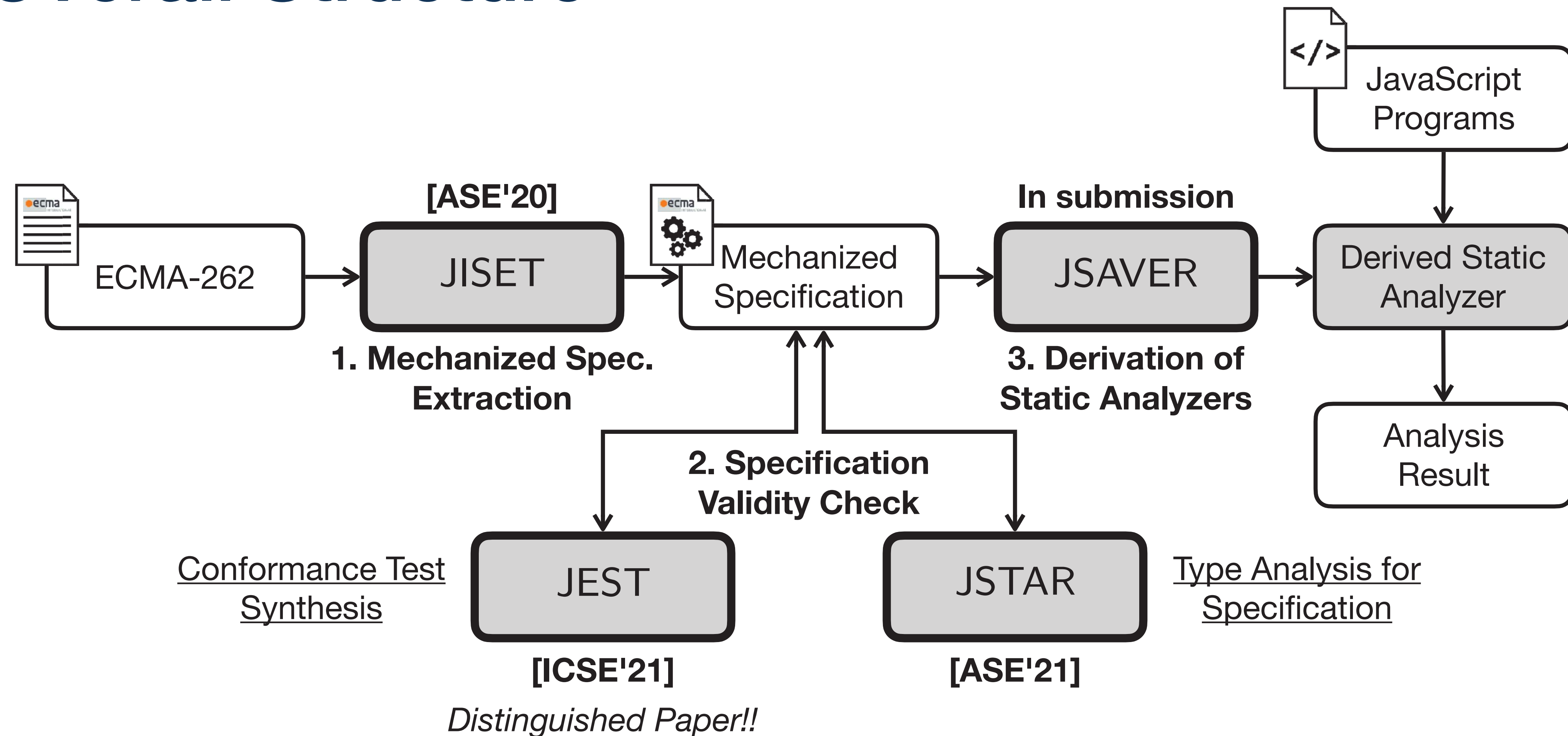
Problem: Fast Evolving JavaScript



Main Idea: Deriving Static Analyzer from Spec.

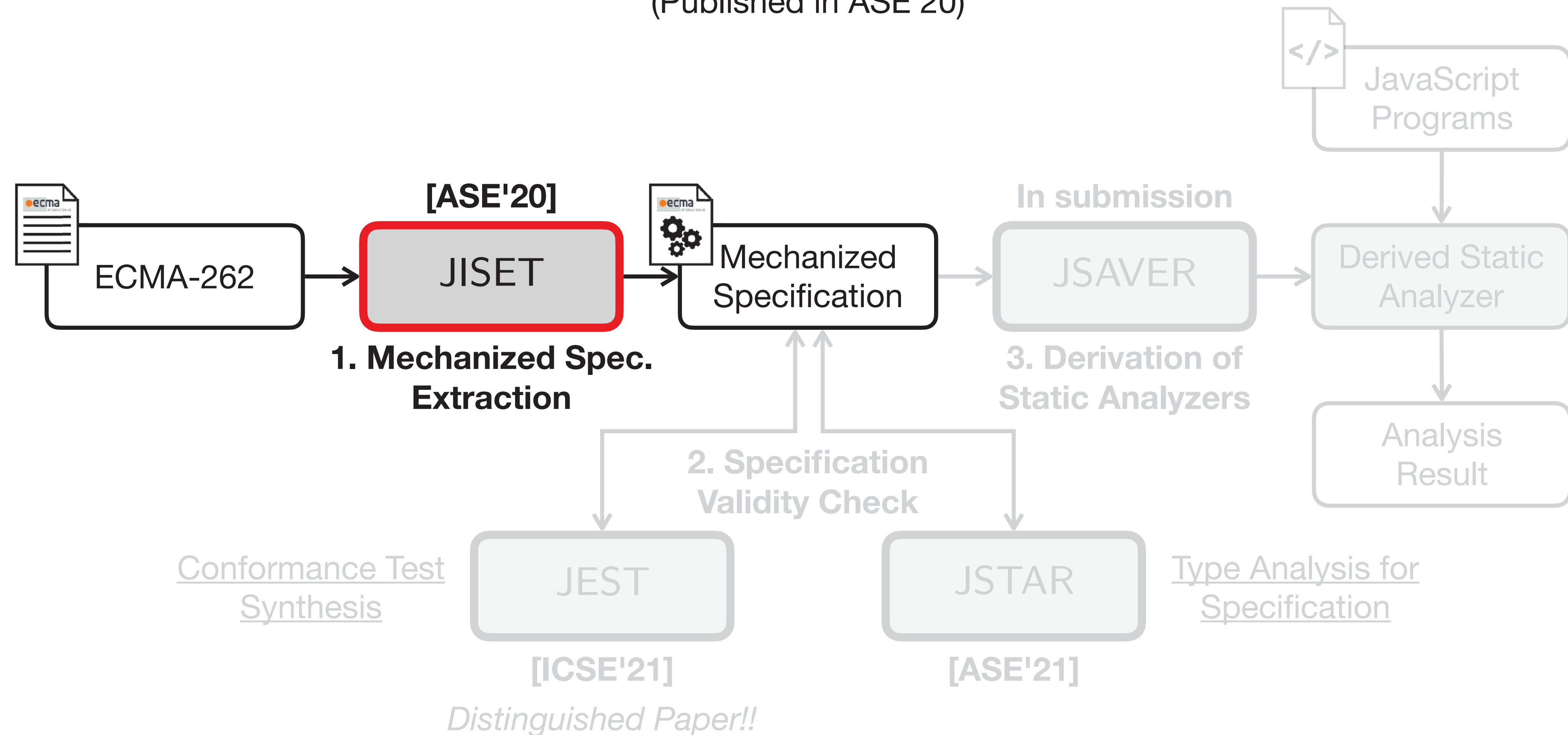


Overall Structure



JISSET: JavaScript IR-based Semantics Extraction Toolchain

Jihyeok Park, Jihee Park, Seungmin An, and Sukyoung Ryu
(Published in ASE'20)



Motivation: Patterns in Writing Style of ECMA-262

13.2.5.2 Runtime Semantics: Evaluation

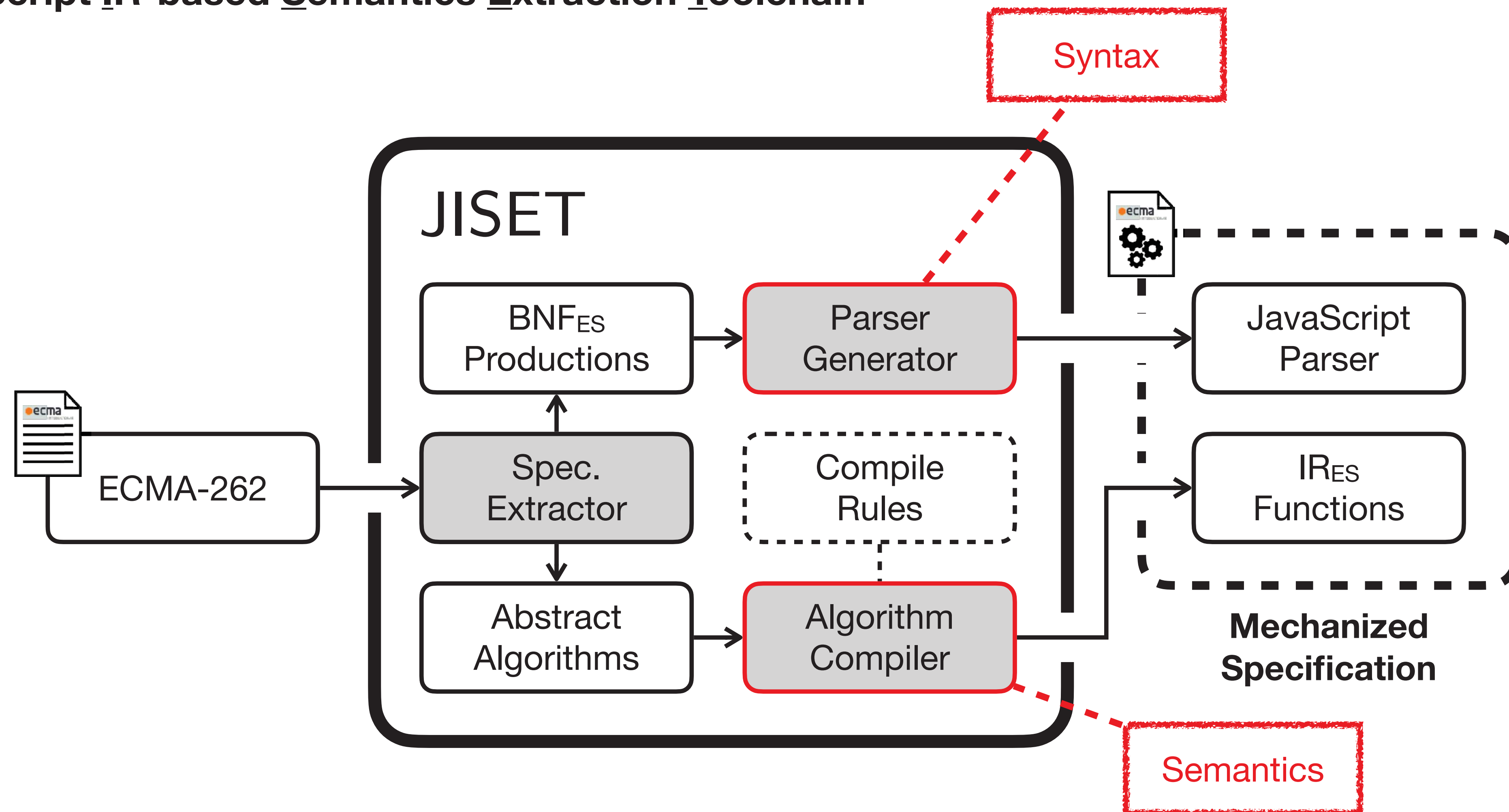
ArrayLiteral : [*ElementList* , *Elision*_{opt}]

1. Let *array* be ! *ArrayCreate*(0).
2. Let *nextIndex* be the result of performing *ArrayAccumulation* for *ElementList* with arguments *array* and 0.
3. *ReturnIfAbrupt*(*nextIndex*).
4. If *Elision* is present, then
 - a. Let *len* be the result of performing *ArrayAccumulation* for *Elision* with arguments *array* and *nextIndex*.
 - b. *ReturnIfAbrupt*(*len*).
5. Return *array*.

The Evaluation algorithm for
the third alternative of *ArrayLiteral* in ES12

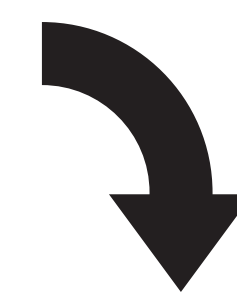
JISET [ASE'20]

JavaScript IR-based Semantics Extraction Toolchain



JISSET - Parser Generator (Syntax)

```
ArrayLiteral[Yield, Await] :  
  [ Elisionopt ]  
  [ ElementList[?Yield, ?Await] ]  
  [ ElementList[?Yield, ?Await] , Elisionopt ]
```



Parsing Expression Grammar
(+ Lookahead Parsing)

JavaScript Parser
in Scala

```
val ArrayLiteral: List[Boolean] => LParser[T] = memo {  
  case List(Yield, Await) =>  
    "[" ~ opt(Elision) ~ "]"          ^^ ArrayLiteral0 |  
    "[" ~ ElementList(Yield, Await) ~ "]" ^^ ArrayLiteral1 |  
    "[" ~ ElementList(Yield, Await) ~ ","  
      ~ opt(Elision) ~ "]"          ^^ ArrayLiteral2  
}
```

(POPL'04) Bryan Ford, "Parsing Expression Grammars: A Recognition-based Syntactic Foundation"

JISET - Algorithm Compiler (Semantics)

13.2.5.2 Runtime Semantics: Evaluation

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

1. Let *array* be ! *ArrayCreate*(0).
2. Let *nextIndex* be the result of performing *ArrayAccumulation* for *ElementList* with arguments *array* and 0.
3. *ReturnIfAbrupt*(*nextIndex*).
4. If *Elision* is present, then
 - a. Let *len* be the result of performing *ArrayAccumulation* for *Elision* with arguments *array* and *nextIndex*.
 - b. *ReturnIfAbrupt*(*len*).
5. Return *array*.

**118 Compile Rules for
Steps in Abstract Algorithms**

**IR_{ES}
Functions**

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

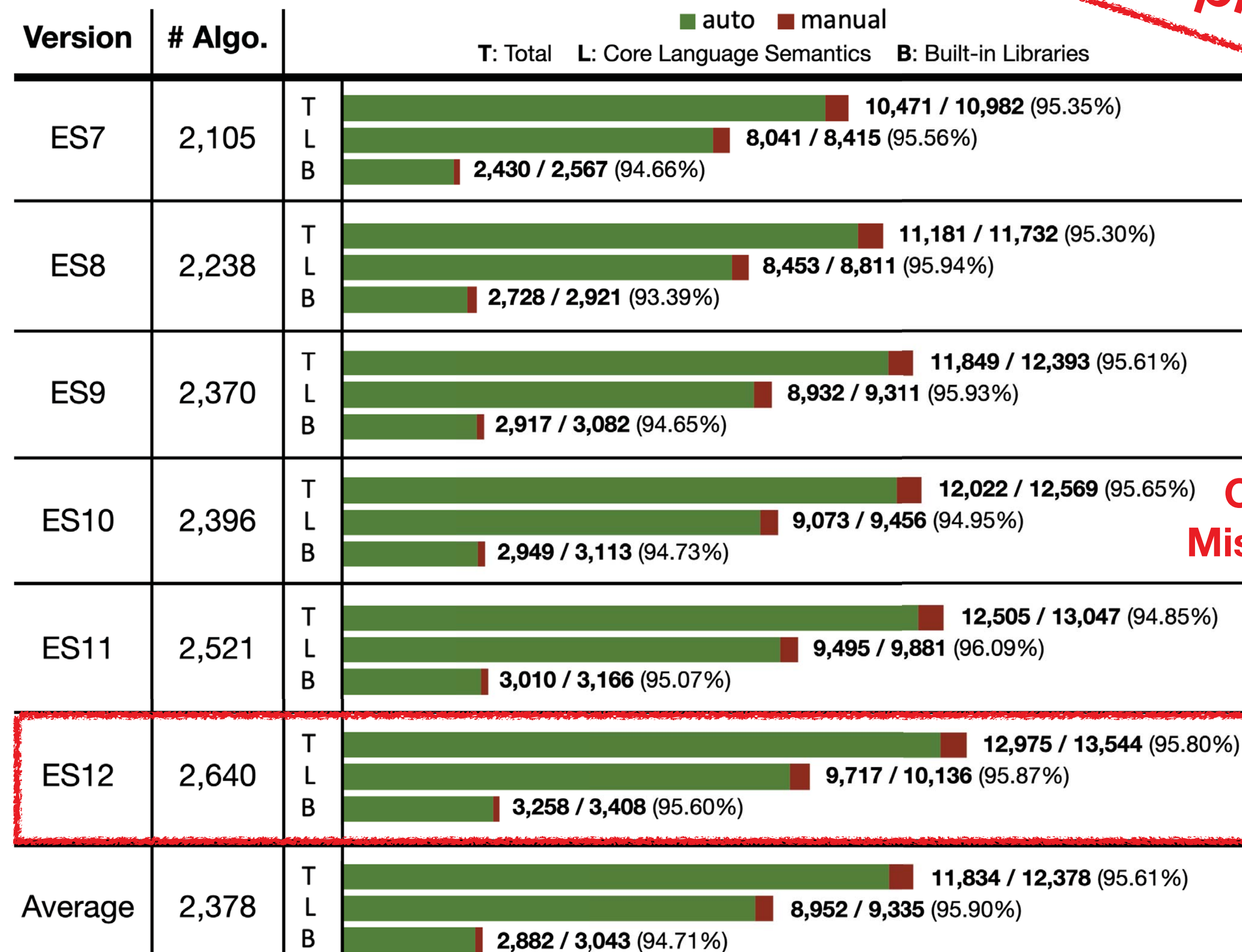
JISSET - Evaluation

**≈ 95%
Compiled**

Version	# Algo.		
		T: Total	L: Core Language Semantics B: Built-in Libraries
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

**≈ 95%
Compiled**



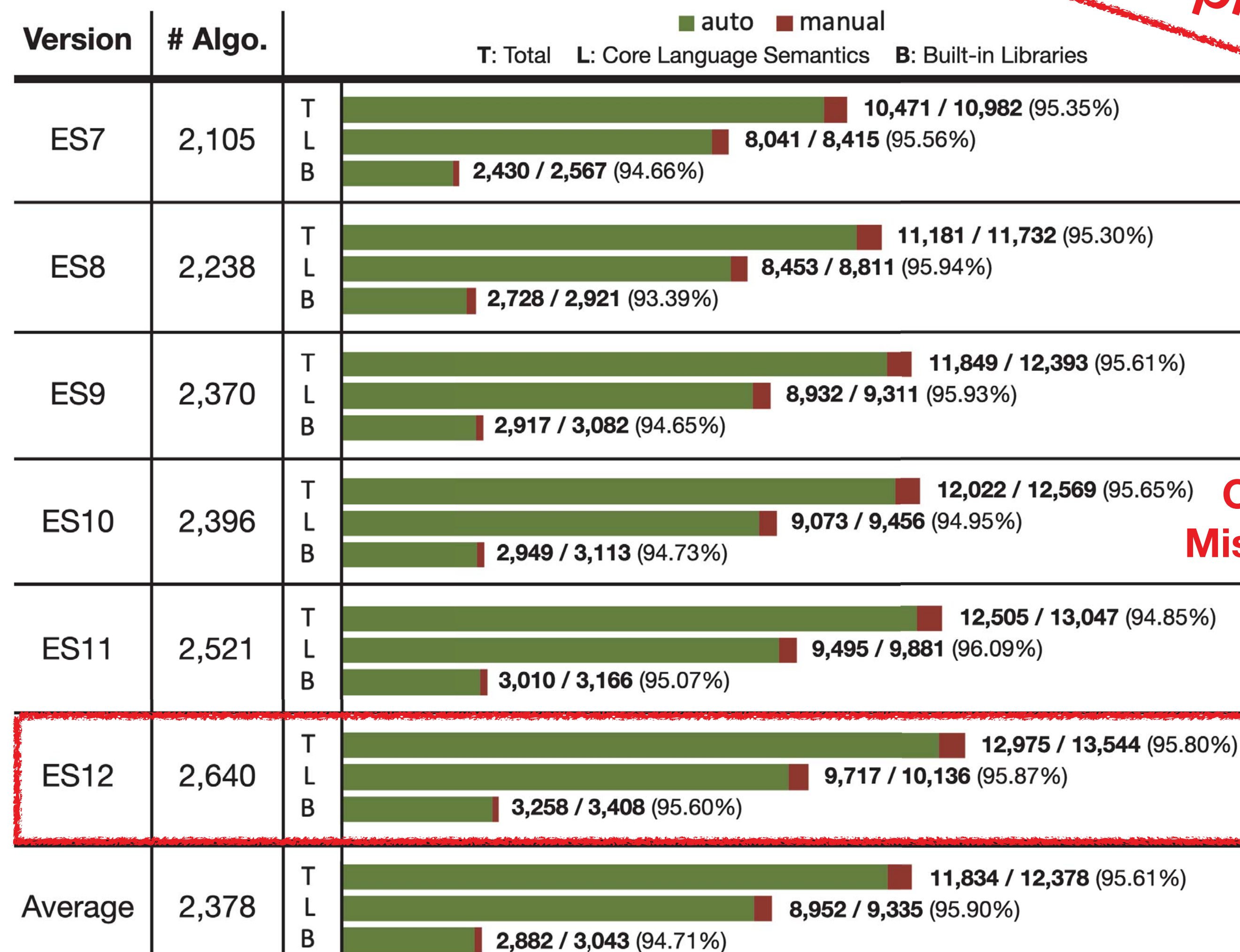
**Complete
Missing Parts**



JISSET - Evaluation

≈ 95%
Compiled

Passed
All Tests

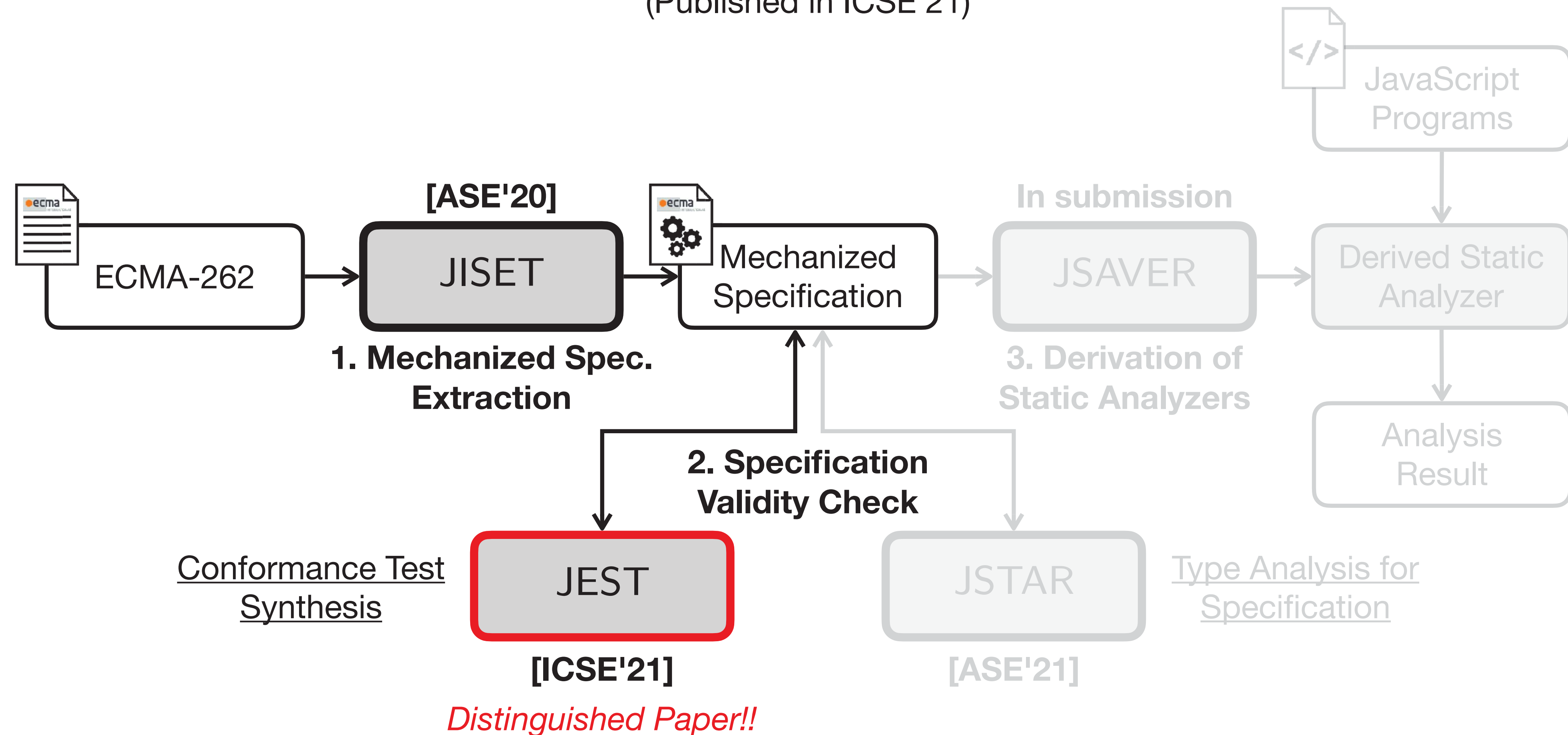


Complete
Missing Parts

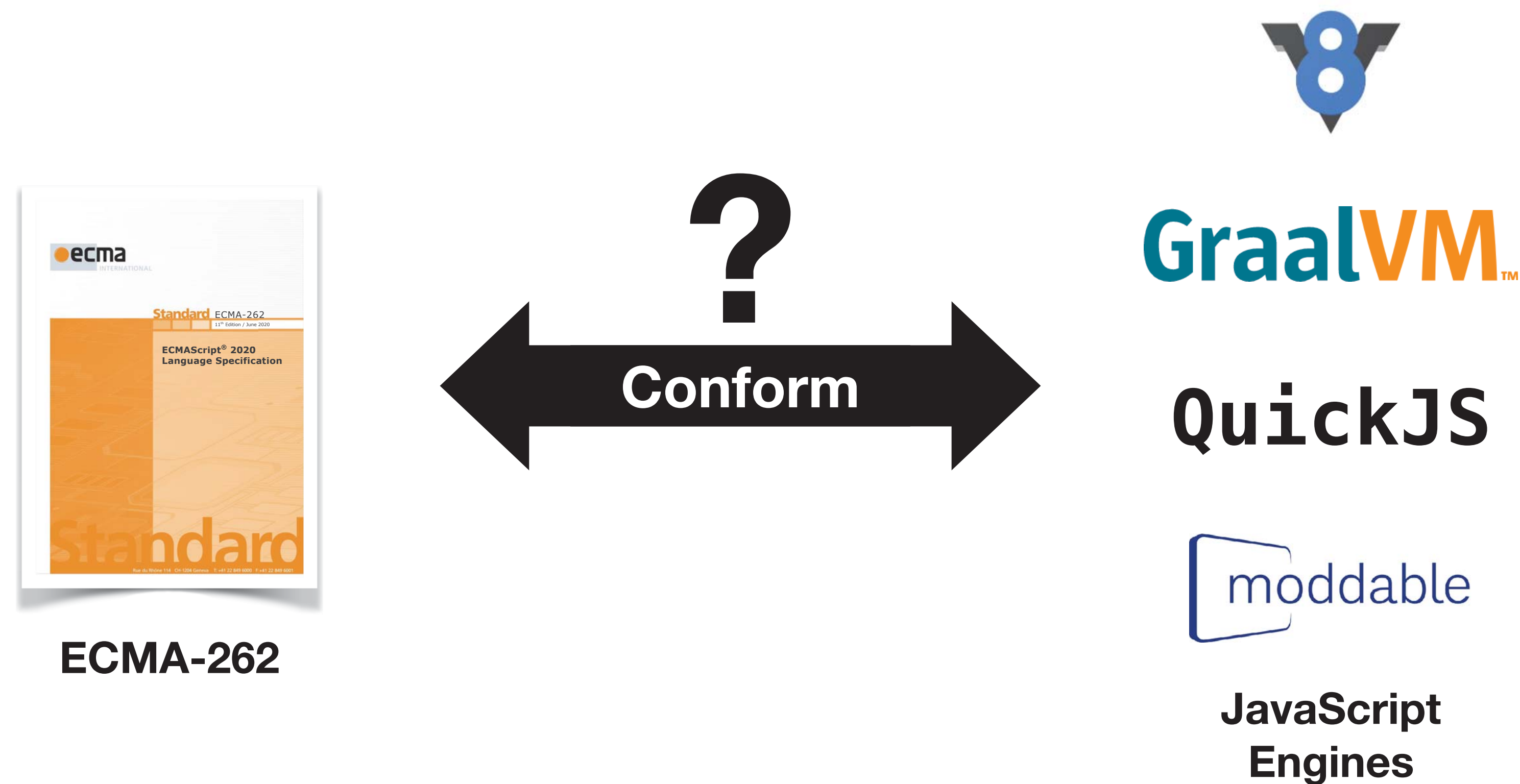
- **Test262**
(Official Conformance Tests)
 - 18,556 applicable tests
- **Parsing tests**
 - Passed all 18,556 tests
- **Evaluation Tests**
 - Passed all 18,556 tests

JEST: N+1-version Differential Testing of Both JavaScript Engines

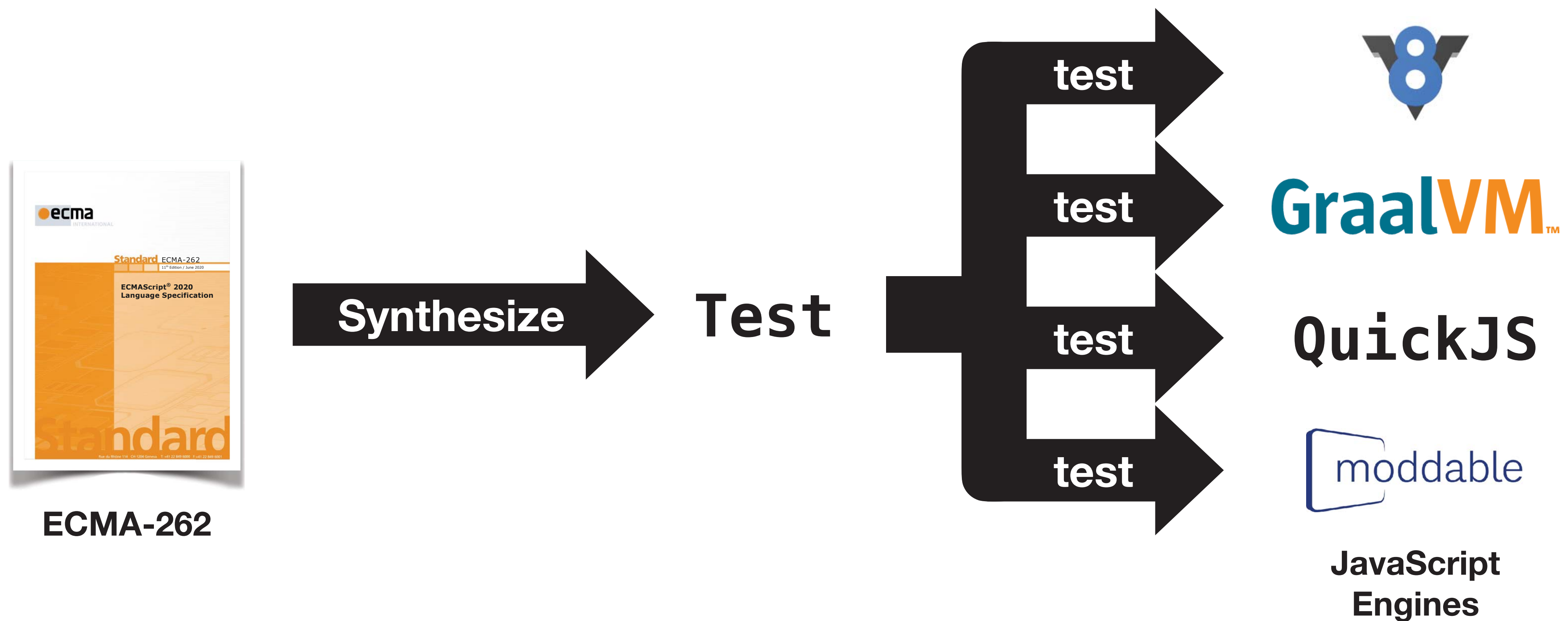
Jihyeok Park, Seungmin An, Dongjun Youn, Gyeongwon Kim, and Sukyoung Ryu
(Published in ICSE'21)



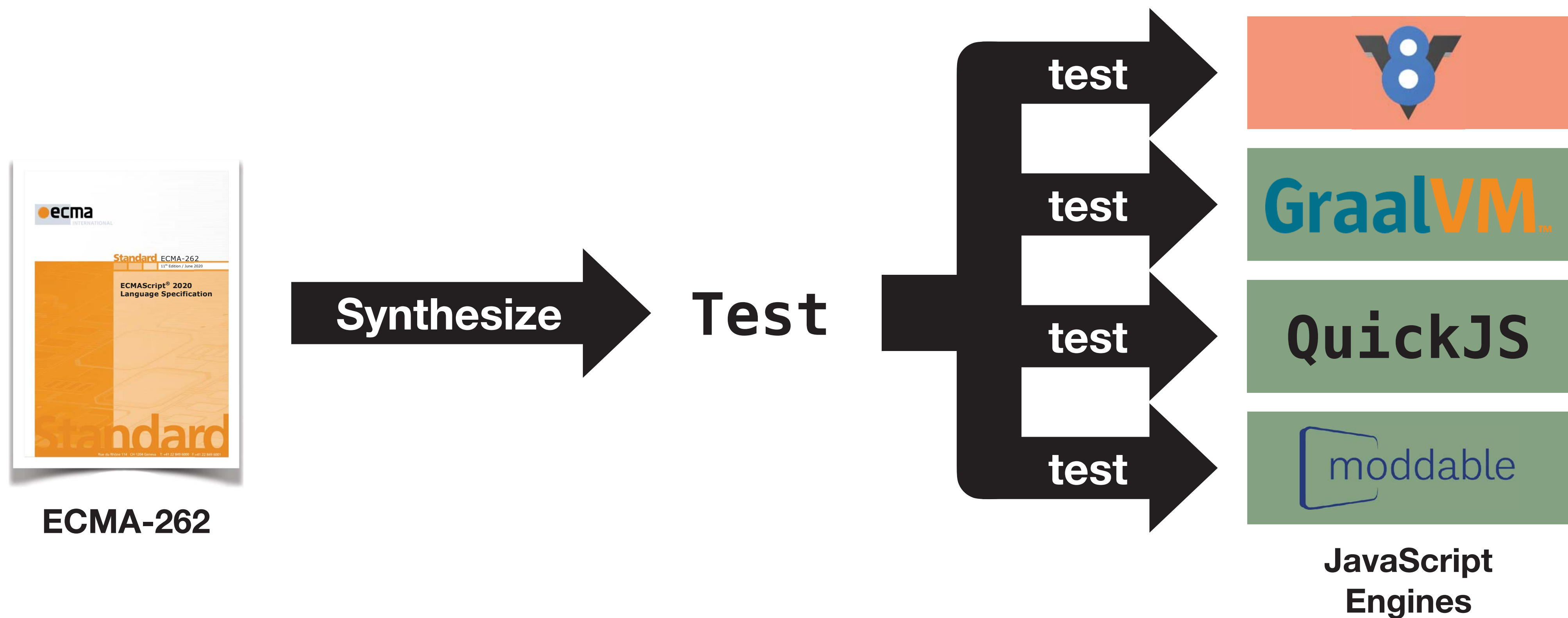
JEST - Conformance with Engines



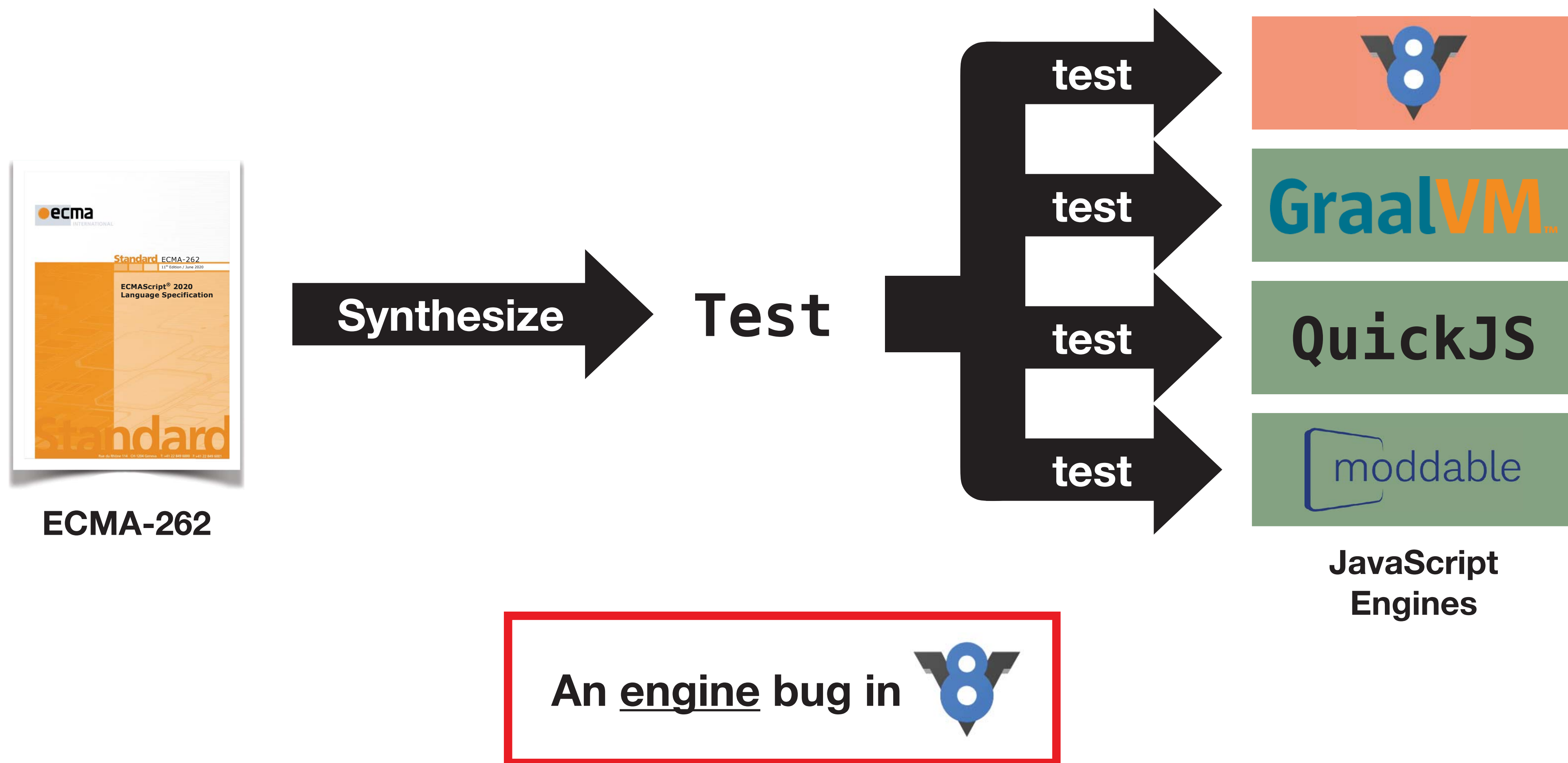
JEST - N+1-version Differential Testing



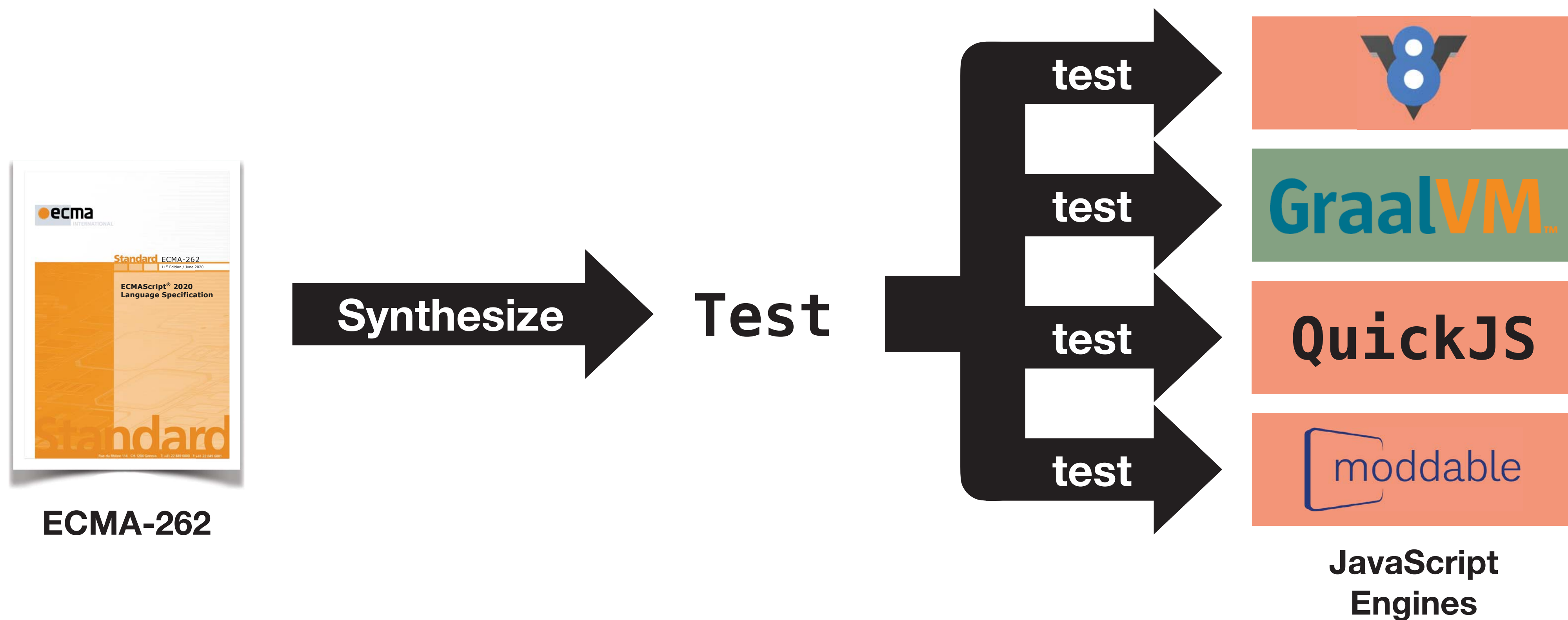
JEST - N+1-version Differential Testing



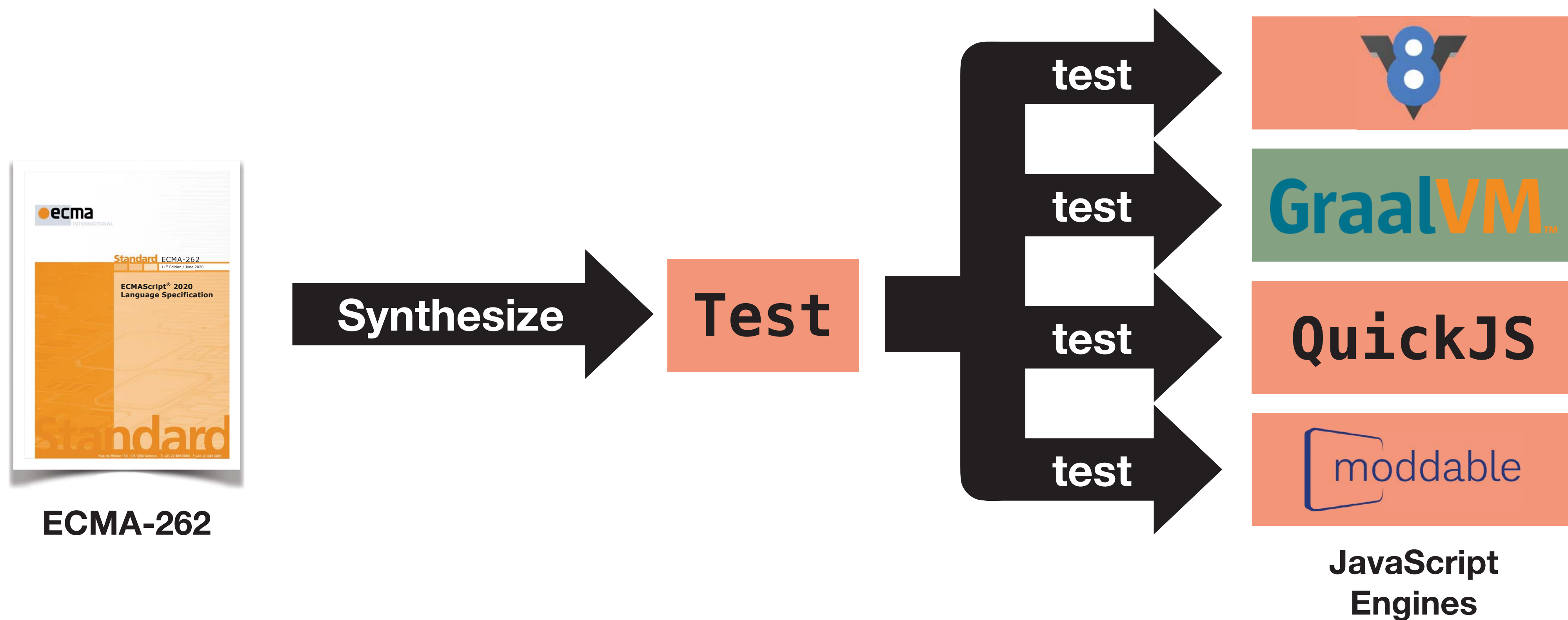
JEST - N+1-version Differential Testing



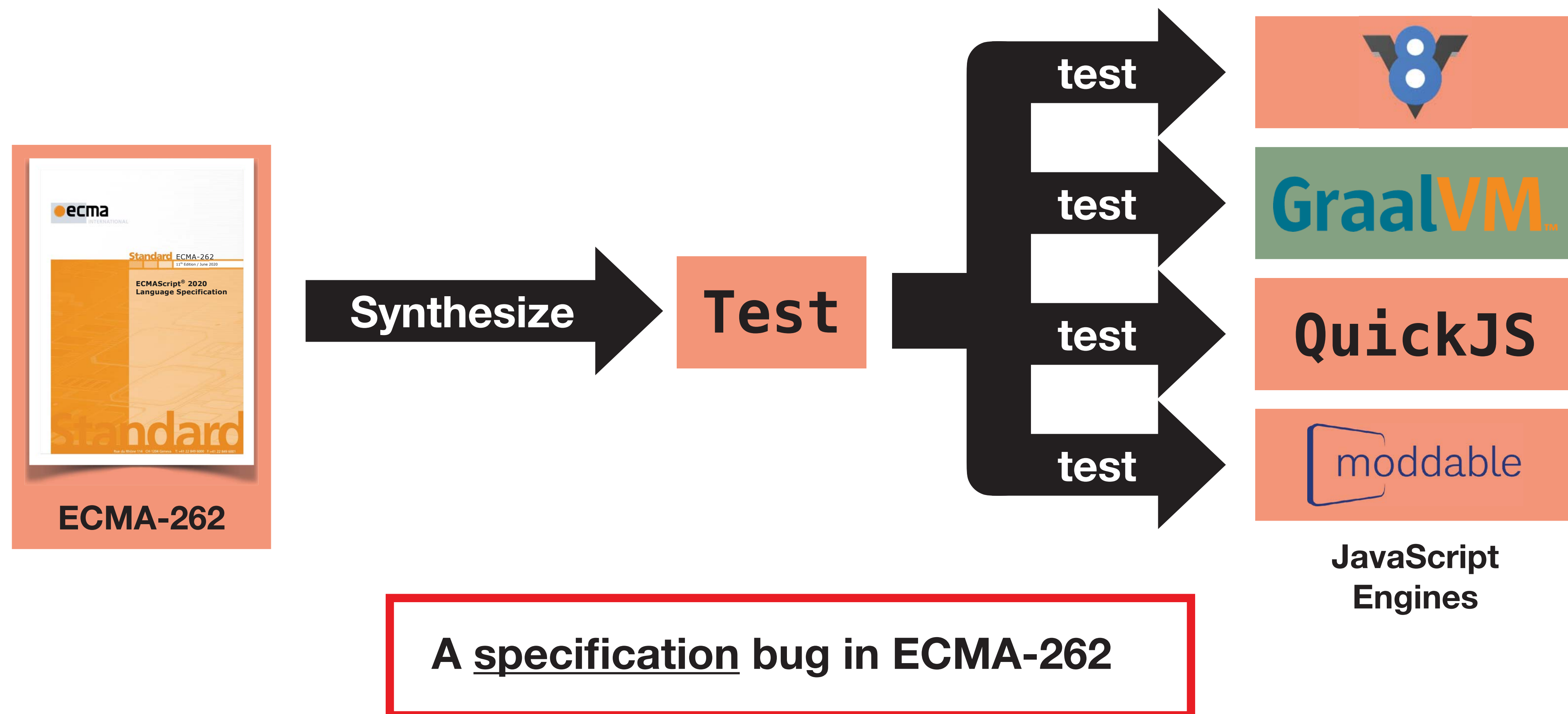
JEST - N+1-version Differential Testing



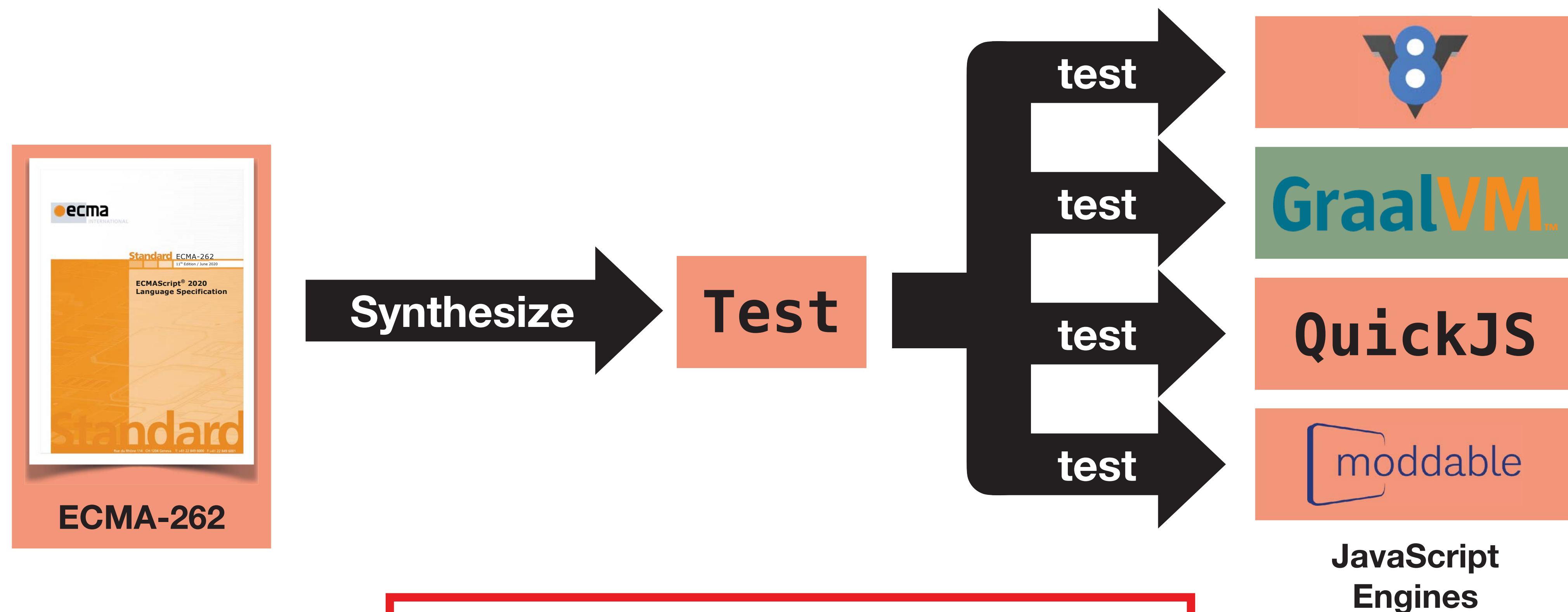
JEST - N+1-version Differential Testing



JEST - N+1-version Differential Testing



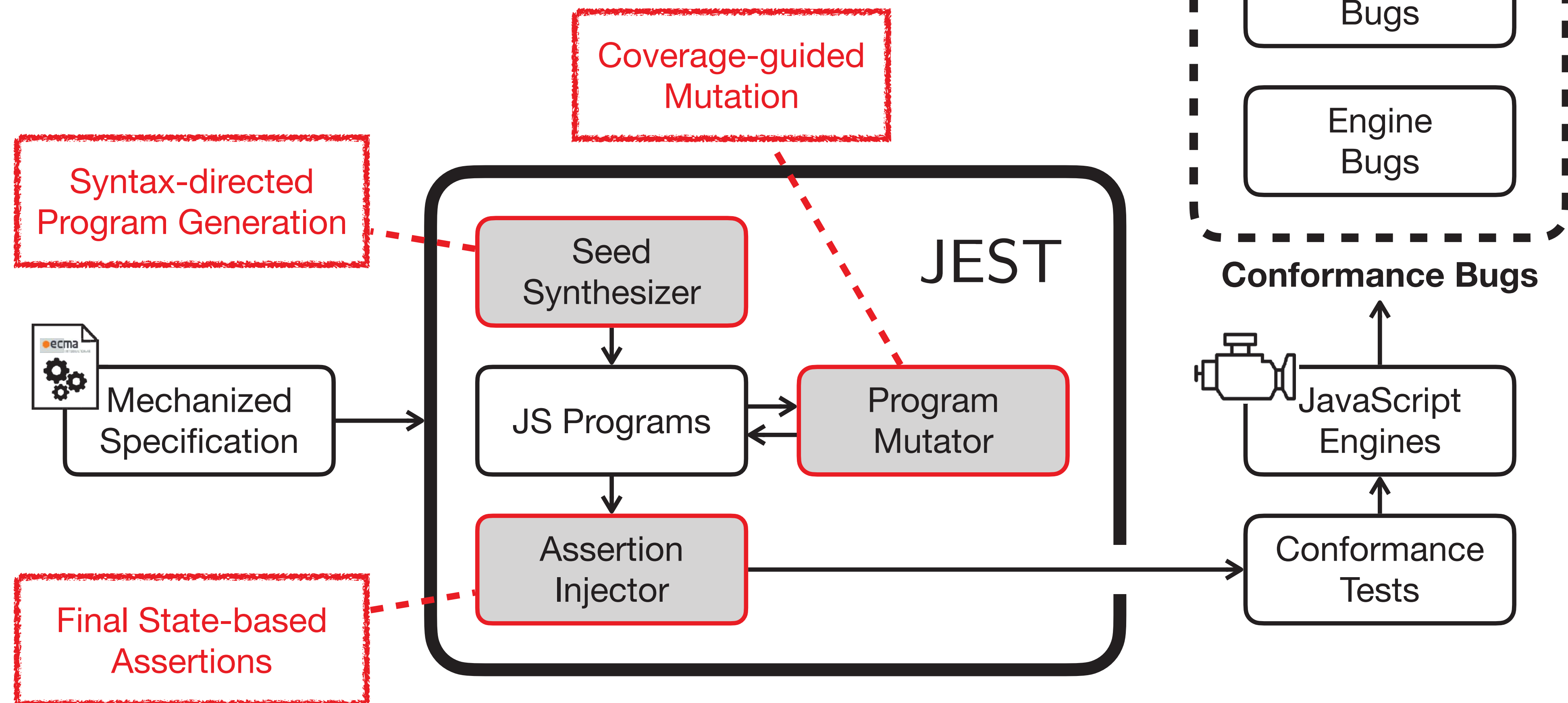
JEST - N+1-version Differential Testing



A specification bug in ECMA-262
An engine bug in **GraalVM**

JEST [ICSE'21]

JavaScript Engines and Specification Tester



JEST - Assertion Injector (7 Kinds)

```
var x = 1 + 2;
```

JEST - Assertion Injector (7 Kinds)

```
var x = 1 + 2;
```

```
+ $assert.sameValue(x, 3);
```

JEST - Assertion Injector (7 Kinds)

1. Exceptions (Exc)

```
+ // Throw  
let x = 42;  
function x() {};
```

2. Aborts (Abort)

```
+ // Abort  
var x = 42; x++;
```

3. Variable Values (Var)

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

4. Object Values (Obj)

```
var x = {}, y = {}, z = { p: x, q: y };  
+ $assert.sameValue(z.p, x);  
+ $assert.sameValue(z.q, y);
```

JEST - Assertion Injector (7 Kinds)

5. Object Properties (Desc)

```
var x = { p: 42 };  
+ $verifyProperty(x, "p", {  
+   value: 42.0, writable: true,  
+   enumerable: true, configurable: true  
+ });
```

6. Property Keys (Key)

```
var x = {[Symbol.match]: 0, p: 0, 3: 0, q: 0, 1: 0}  
+ $assert.compareArray(  
+   Reflect.ownKeys(x),  
+   ["1", "3", "p", "q", Symbol.match]  
+ );
```

7. Internal Methods and Slots (In)

```
function f() {}  
+ $assert.sameValue(Object.getPrototypeOf(f),  
+   Function.prototype);  
+ $assert.sameValue(Object.isExtensible(x), true);  
+ $assert.callable(f);  
+ $assert.constructable(f);
```

JEST - Evaluation

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
GraalJS	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

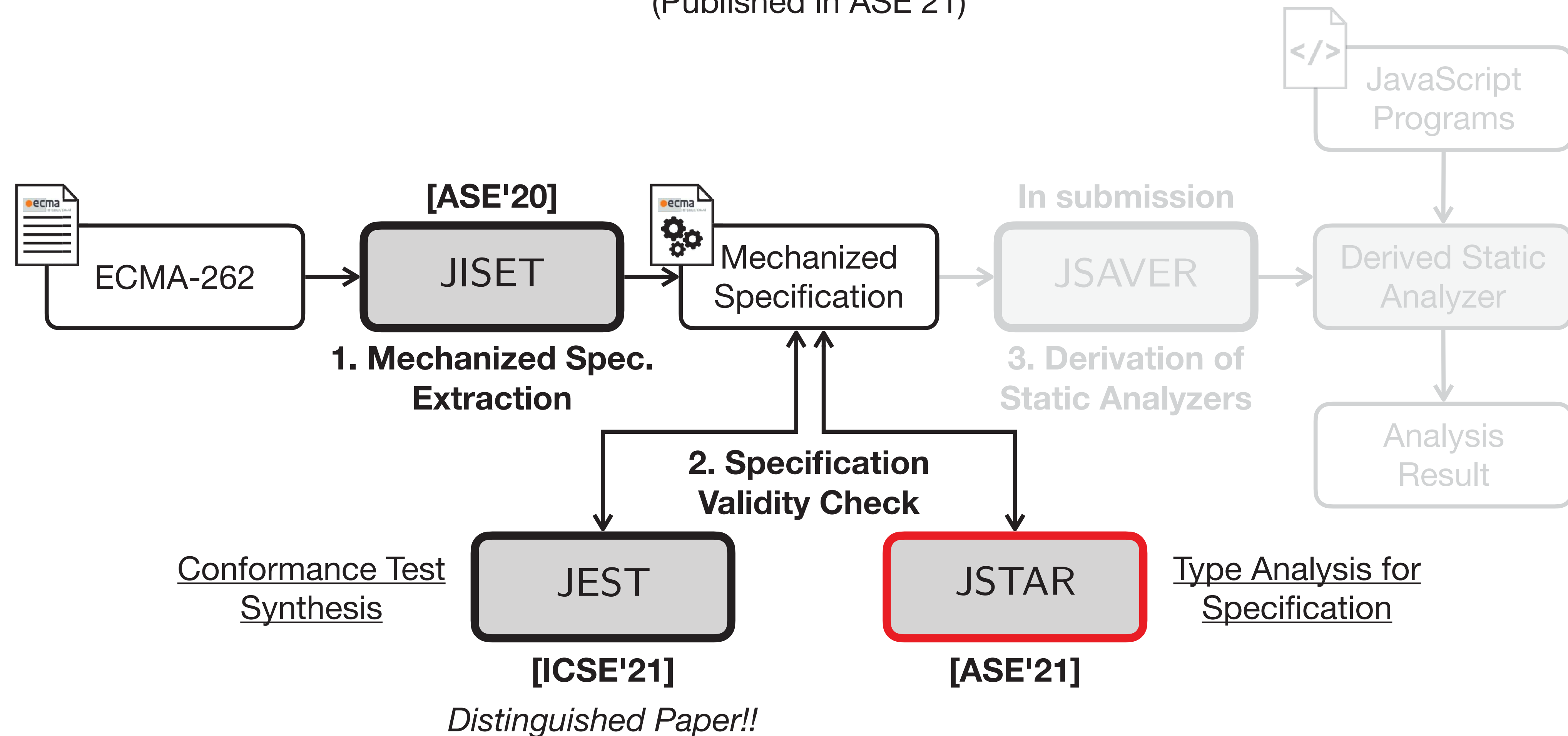
27 Bugs
in Spec.

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

JSTAR: JavaScript Specification Type Analyzer using Refinement

Jihyeok Park, Seungmin An, Wonho Shin, Yusung Sim, and Sukyoung Ryu
(Published in ASE'21)



JSTAR - Types in Specification

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 - Types in Specification

20.3.2.28 Math.round (x) x : (String v Boolean v Number v Object v ...)

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 - Types in Specification

20.3.2.28 Math.round (x) x : (String v Boolean v Number v Object v ...)

1. Let n be ? **ToNumber(x)**. ToNumber(x): (Number v Exception)
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 - Types in Specification

20.3.2.28 Math.round (x) x : (String v Boolean v Number v Object v ...)

1. Let n be ? **ToNumber(x)**. ToNumber(x): (Number v Exception) $\wedge$ n : (Number)
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 - Types in Specification

20.3.2.28 Math.round (x) x : (String v Boolean v Number v Object v ...)

1. Let n be ? **ToNumber(x)**. ToNumber(x): (Number v Exception) $\wedge$ n : (Number)
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 Mismatch for
numeric operator ` $>$ `

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

JSTAR - Types in Specification

20.3.2.28 Math.round (x) x : (String v Boolean v Number v Object v ...)

1. Let n be ? **ToNumber(x)**. ToNumber(x): (Number v Exception) $\wedge$ n : (Number)
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 Mismatch for
numeric operator ` $>$ `

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

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

JSTAR - Types in Specification

20.3.2.28 Math.round (x) x : (String v Boolean v Number v Object v ...)

1. Let n be ? **ToNumber(x)**. ToNumber(x): (Number v Exception) $\wedge$ n : (Number)
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 Mismatch for
numeric operator ` $>$ `

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

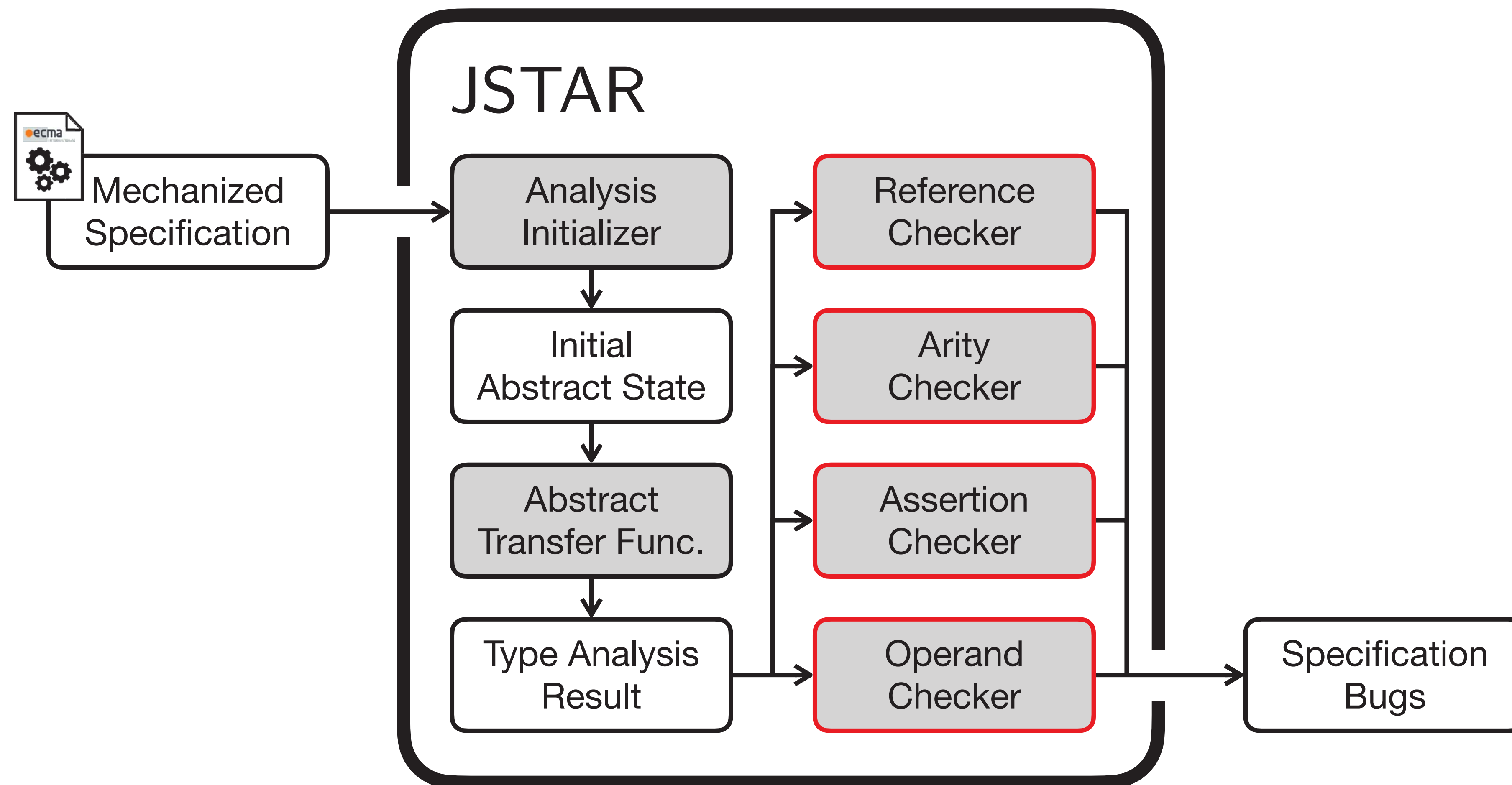
3. If $n < 0.5$ and $n > 0$, return +0.
4. If $n < 0$ and $n \geq -0.5$, return -0.

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

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

JSTAR [ASE'21]

JavaScript Specification Type Aalyzer using Refinement



JSTAR - Evaluation

- Type Analysis for 864 versions of ECMA-262

59.2%
Precision

93 Bugs
Detected

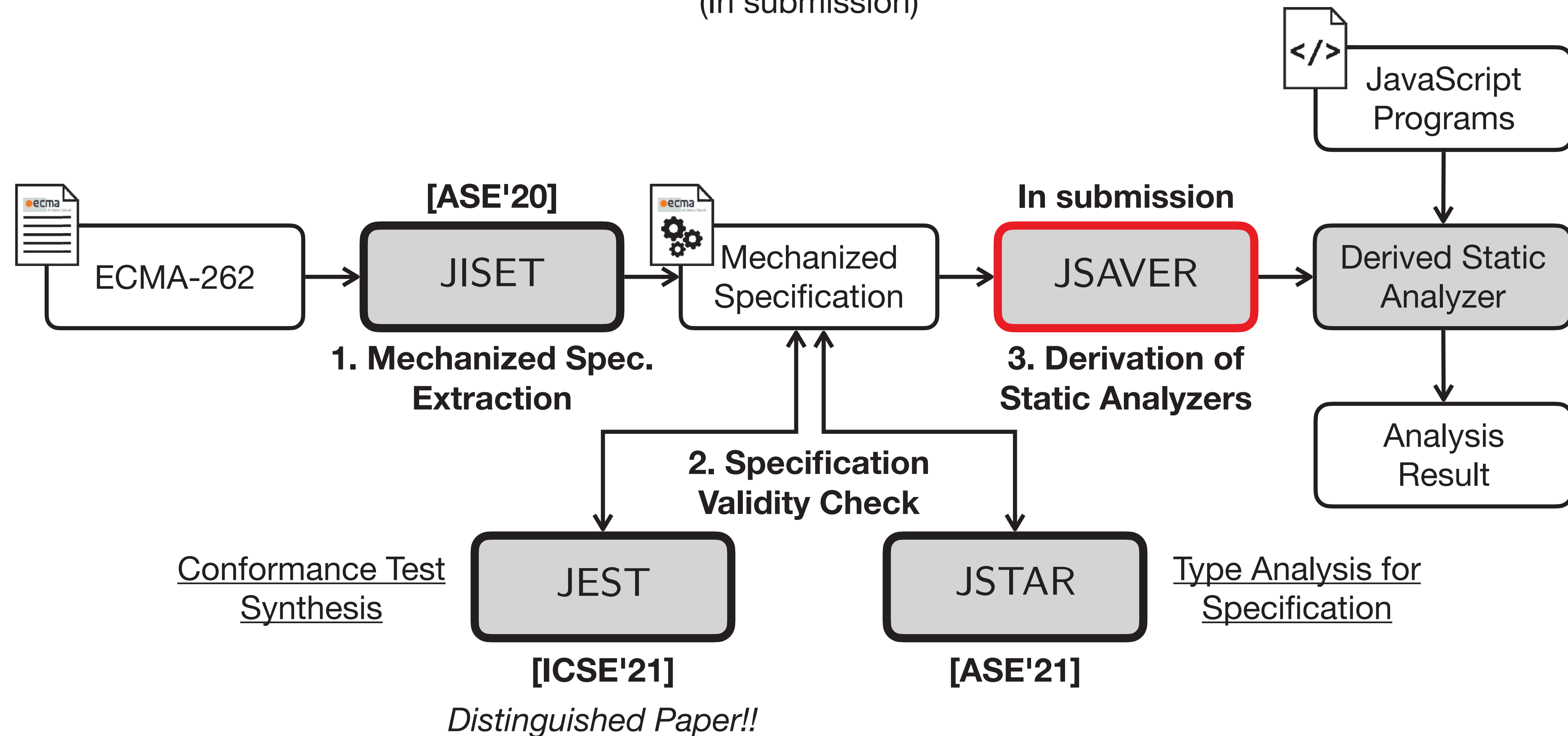
Checker	Bug Kind	Precision = (# True Bugs) / (# Detected Bugs)					
		no-refine		refine		Δ	
Reference	UnknownVar	62 / 106	17 / 60	63 / 78	17 / 31	+1 / -28	/ -29
	DuplicatedVar		45 / 46		46 / 47		+1 / +1
Arity	MissingParam	4 / 4	4 / 4	4 / 4	4 / 4	/	/
Assertion	Assertion	4 / 56	4 / 56	4 / 31	4 / 31	/ -25	/ -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

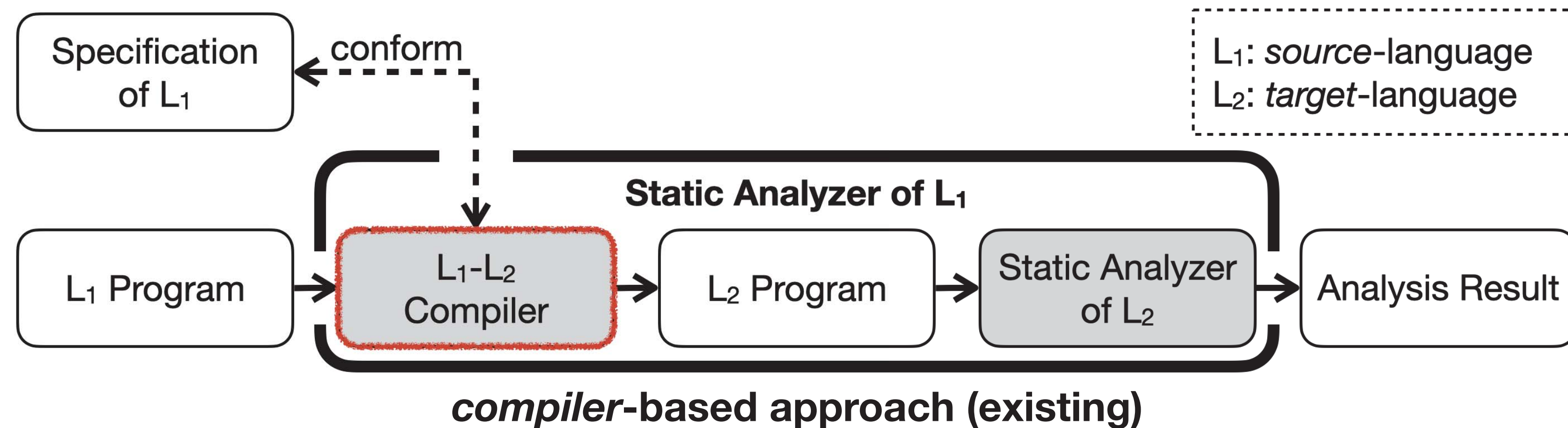
14 Bugs
in ES12

Automatically Deriving JavaScript Static Analyzers from Language Specifications

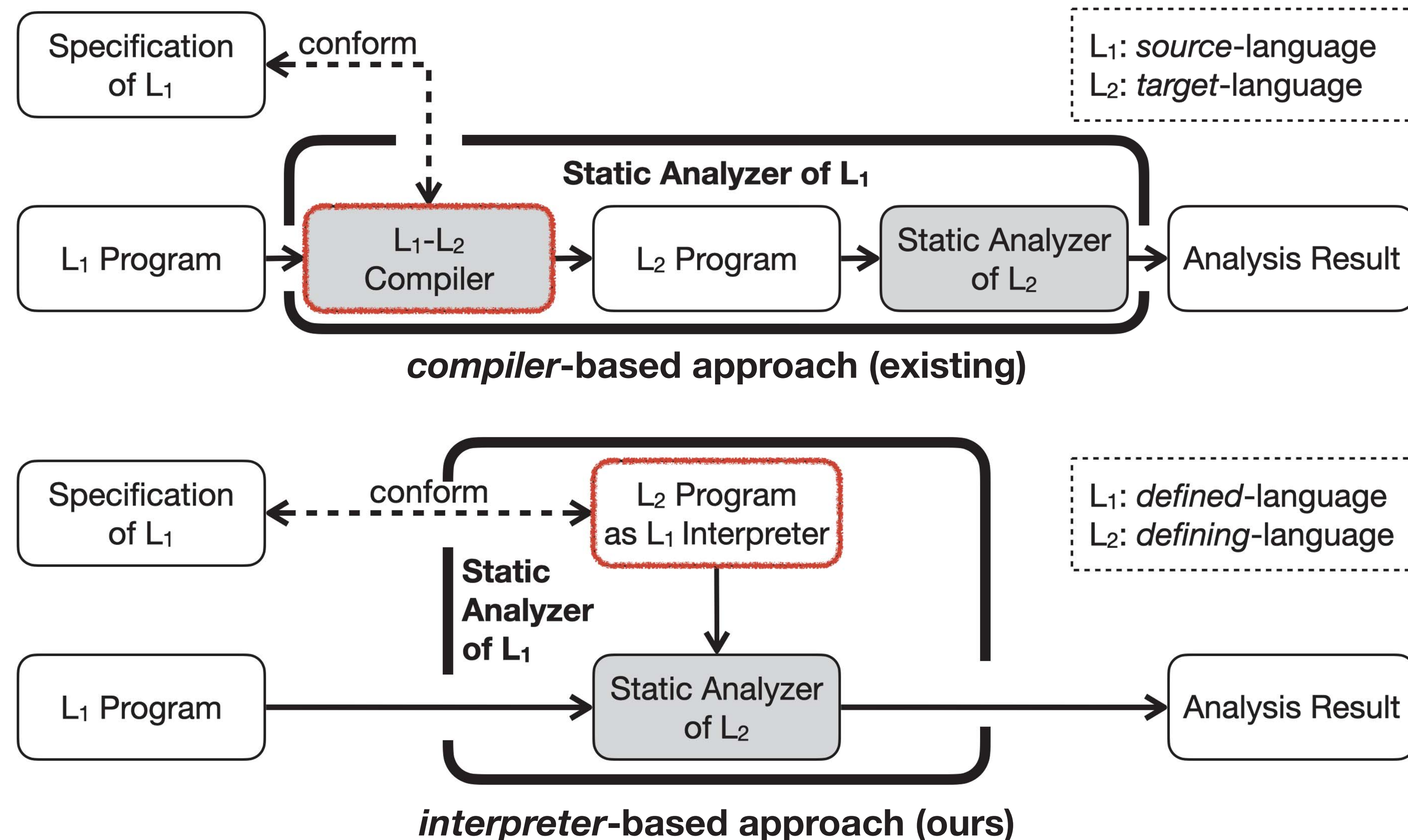
Jihyeok Park, Seungmin An, and Sukyoung Ryu
(In submission)



JSAVER - Meta-Level Static Analysis



JSAVER - Meta-Level Static Analysis



JSAVER - Meta-Level Static Analysis

defined-language
(JavaScript)

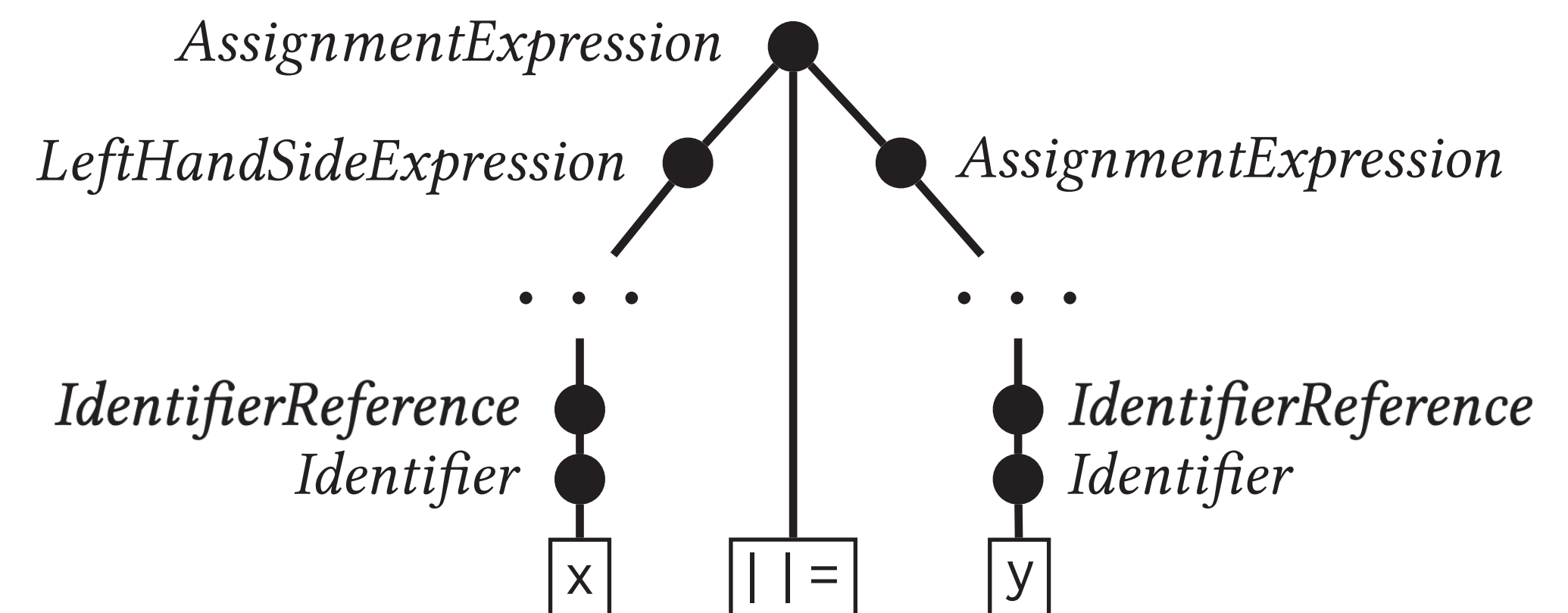
$x \mid \mid = y$

defining-language
(IR_{ES})

JSAVER - Meta-Level Static Analysis

defined-language
(JavaScript)

x || = y **parse** →

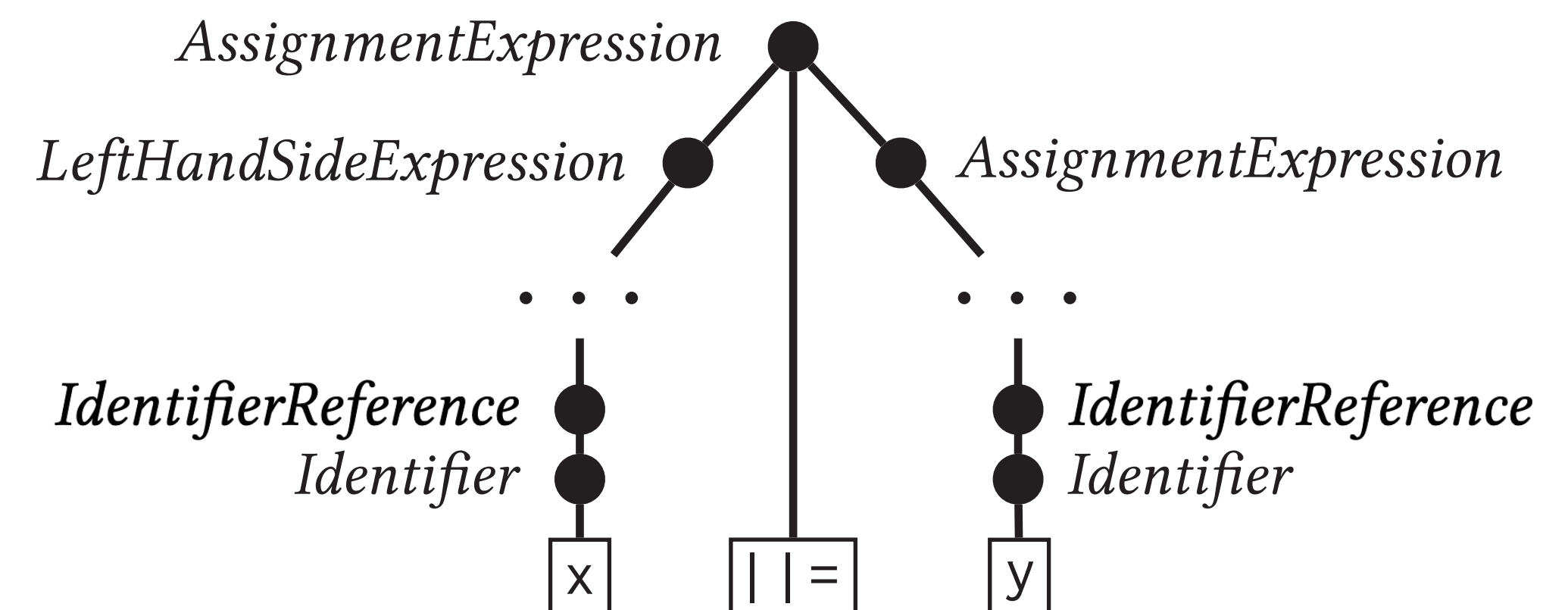


defining-language
(IR_{ES})

JSAVER - Meta-Level Static Analysis

defined-language
(JavaScript)

`x || = y` **parse** →



defining-language
(IR_{ES})

```

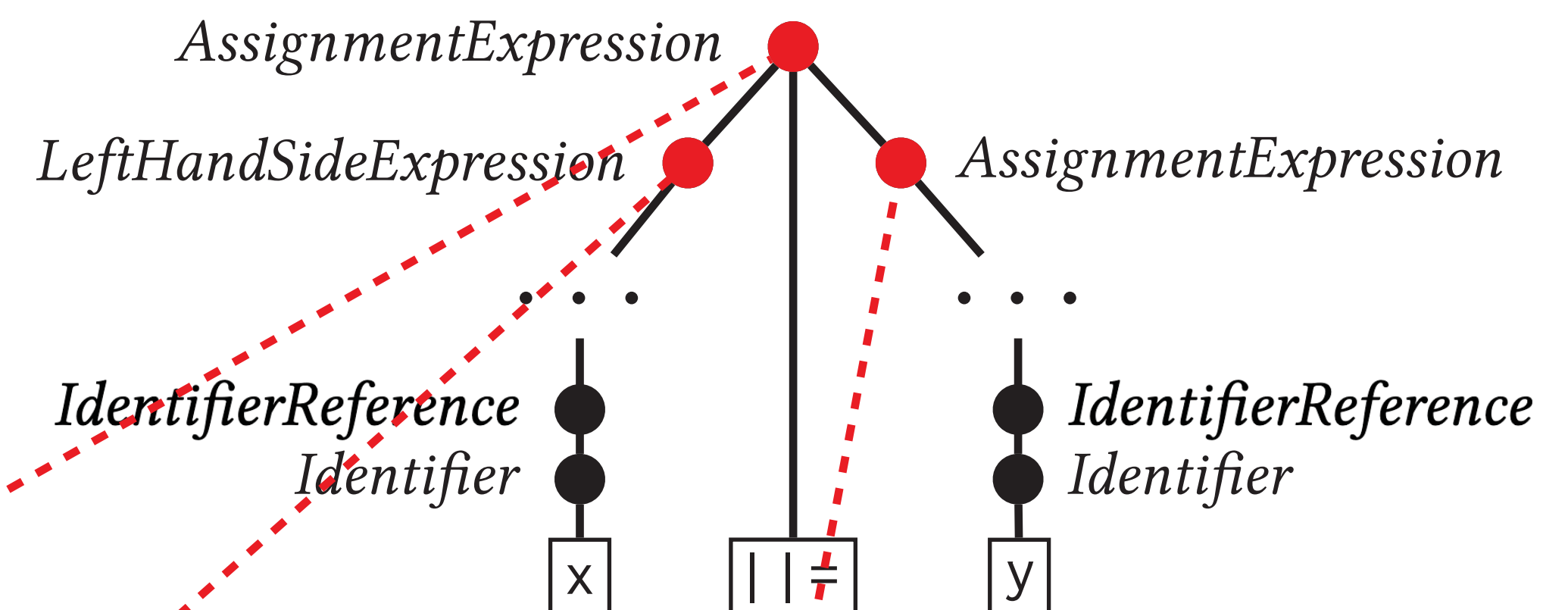
syntax def AssignmentExpression[8].Evaluation(
  this, LeftHandSideExpression, AssignmentExpression
) {
  let lref = (LeftHandSideExpression.Evaluation)
  let lval = [? (GetValue lref)]
  let lbool = [! (ToBoolean lval)]
  if (= lbool true) return lval
  ...
}
  
```

A mechanized specification from ES12
= A JavaScript interpreter
= An IR_{ES} program

JSAVER - Meta-Level Static Analysis

defined-language
(JavaScript)

$x \ || = y$ **parse** 



defining-language
(IR_{ES})

```

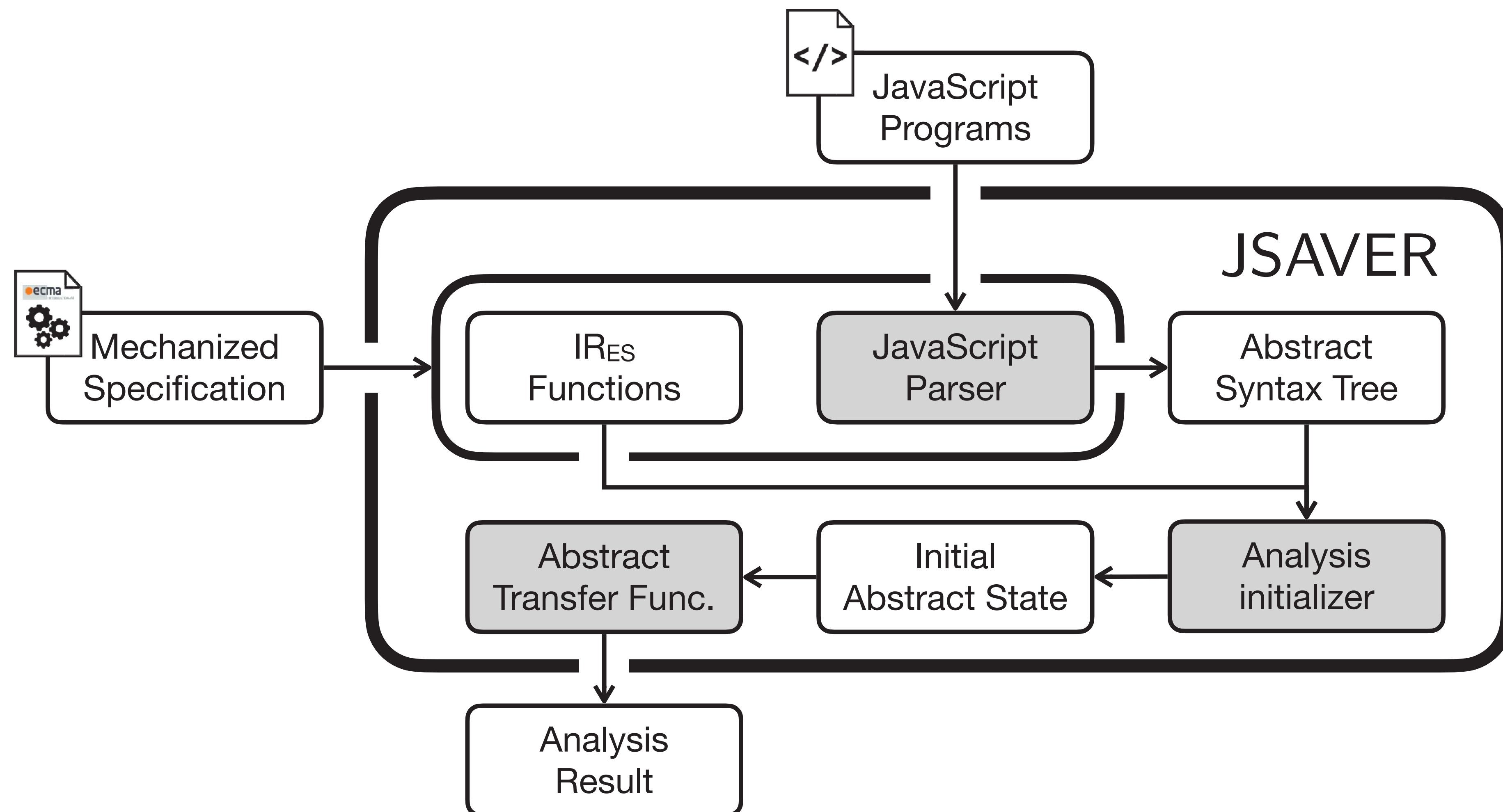
syntax def AssignmentExpression[8].Evaluation(
  this, LeftHandSideExpression, AssignmentExpression
) {
  let lref = (LeftHandSideExpression.Evaluation)
  let lval = [? (GetValue lref)]
  let lbool = [! (ToBoolean lval)]
  if (= lbool true) return lval
  ...
}

```

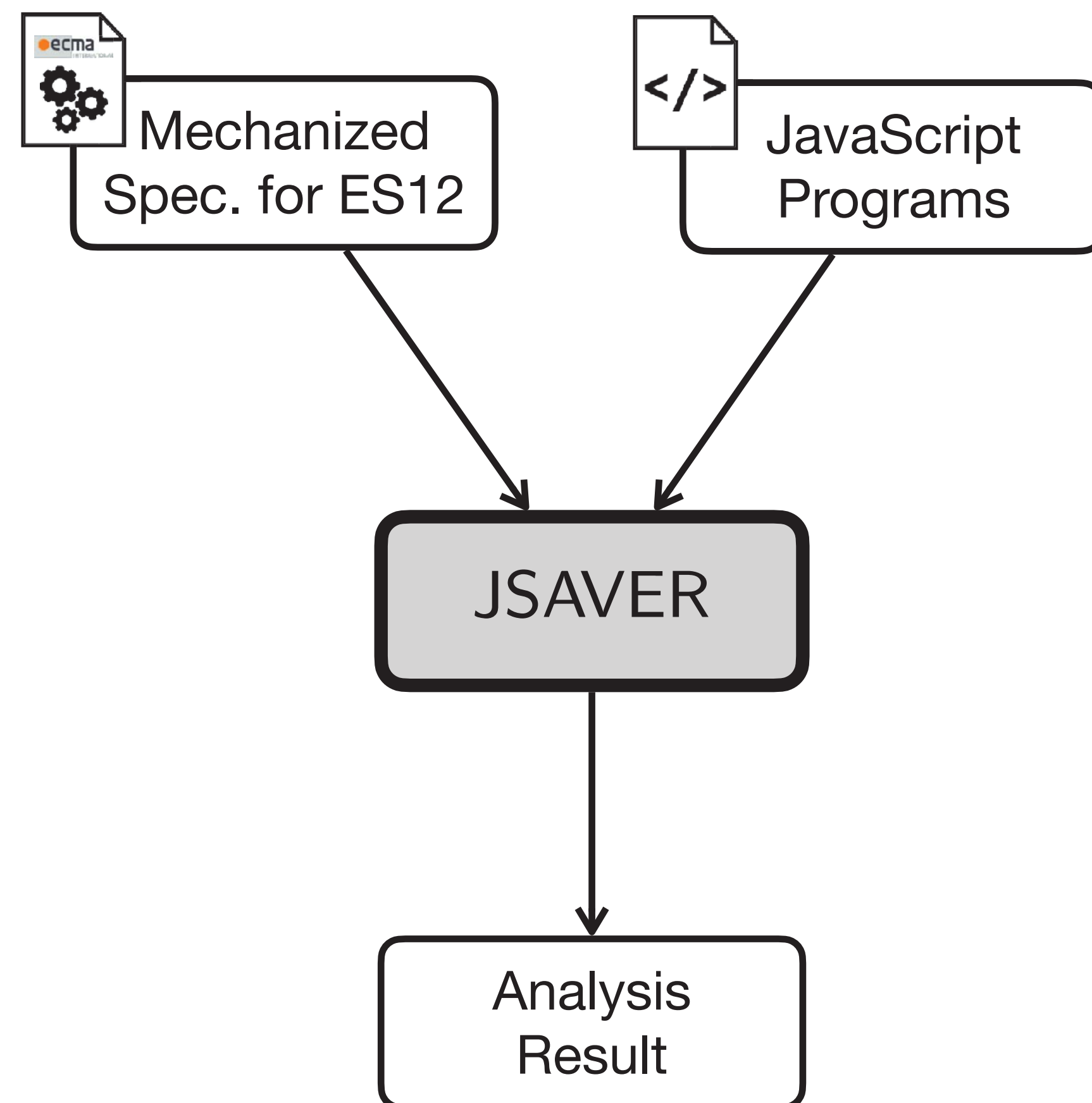
A mechanized specification from ES12
= A JavaScript interpreter
= An IR_{ES} program

JSAVER In submission

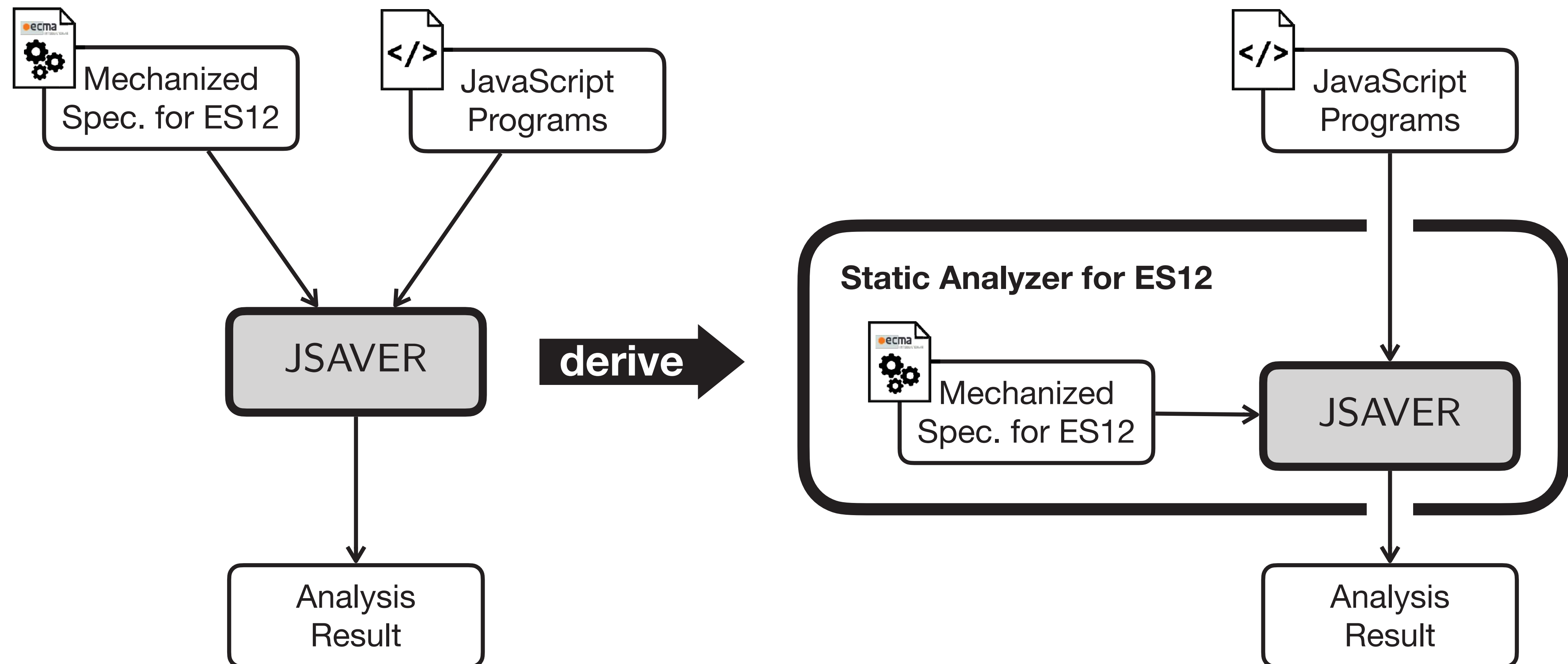
JavaScript Static Analyzer via ECMAScript Representation



JSAVER - Static Analyzer Derivation

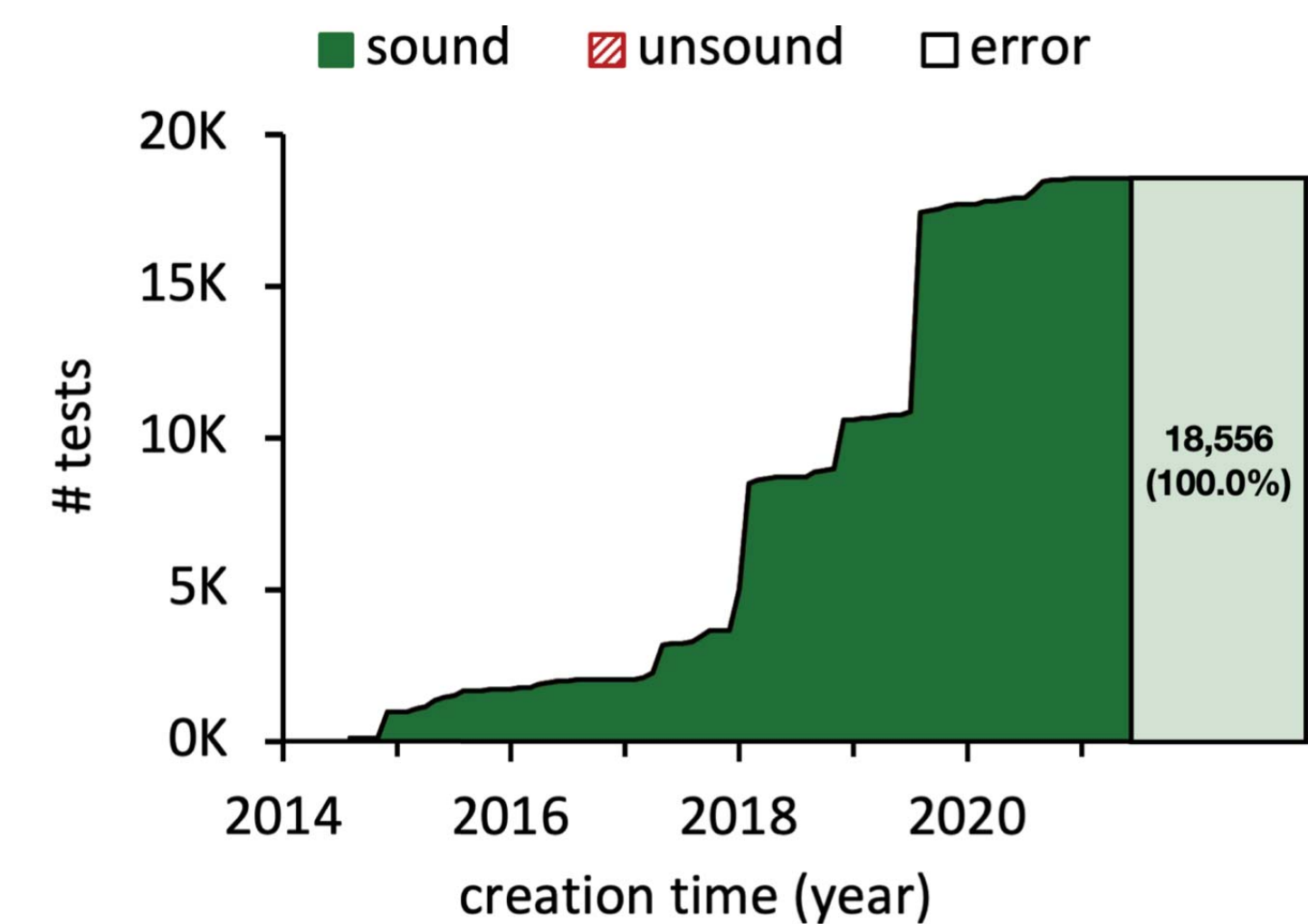
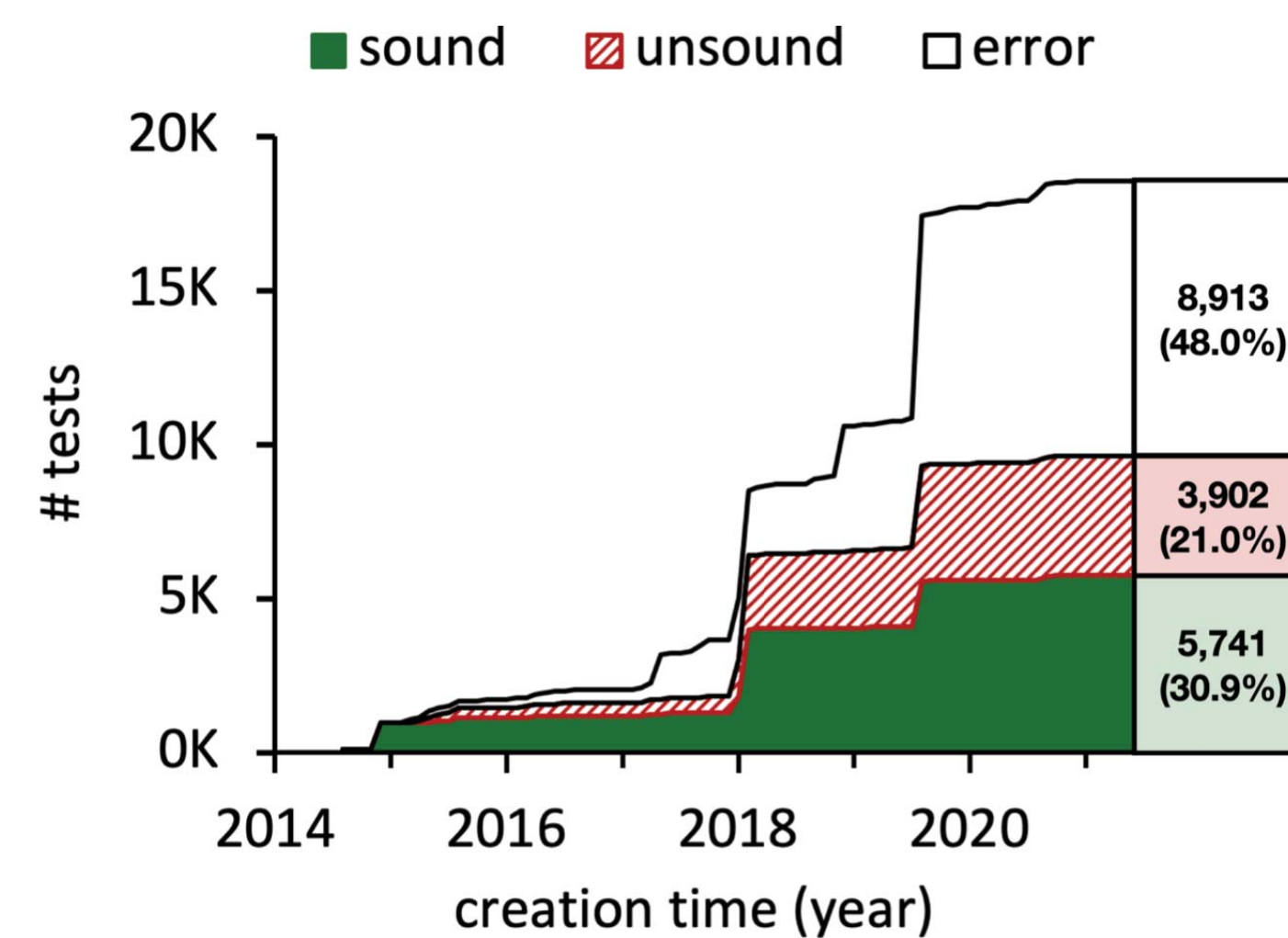
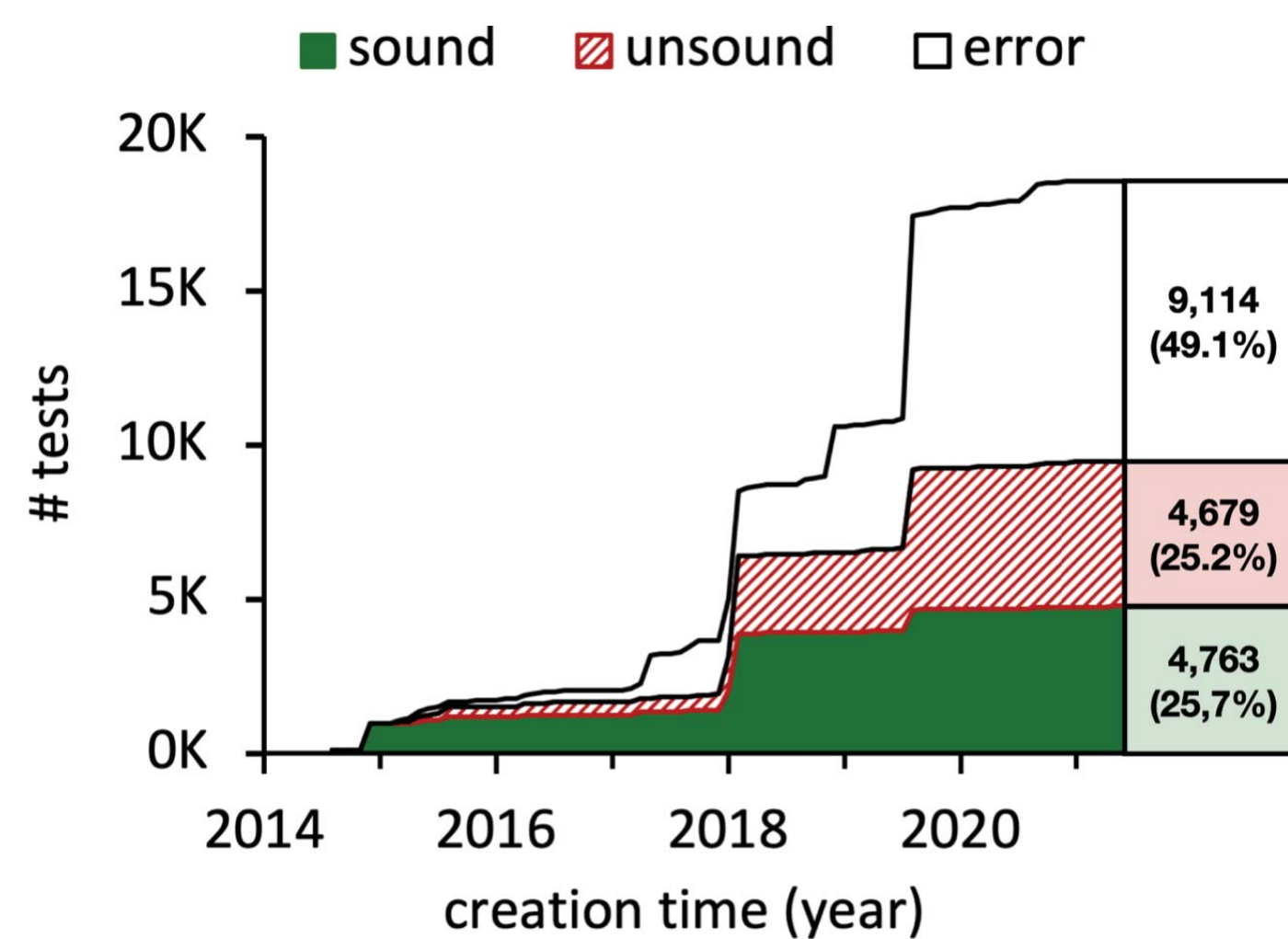
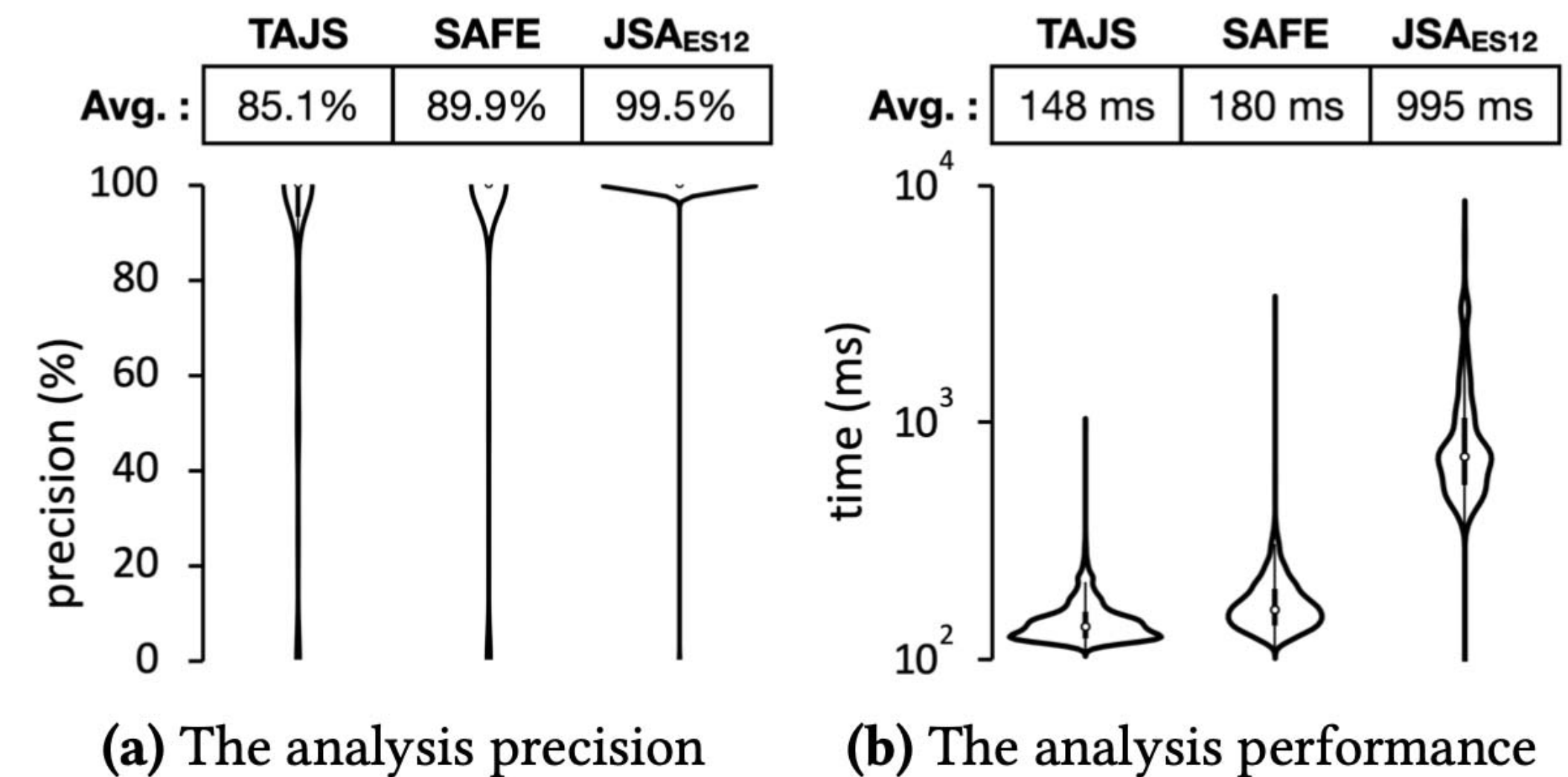


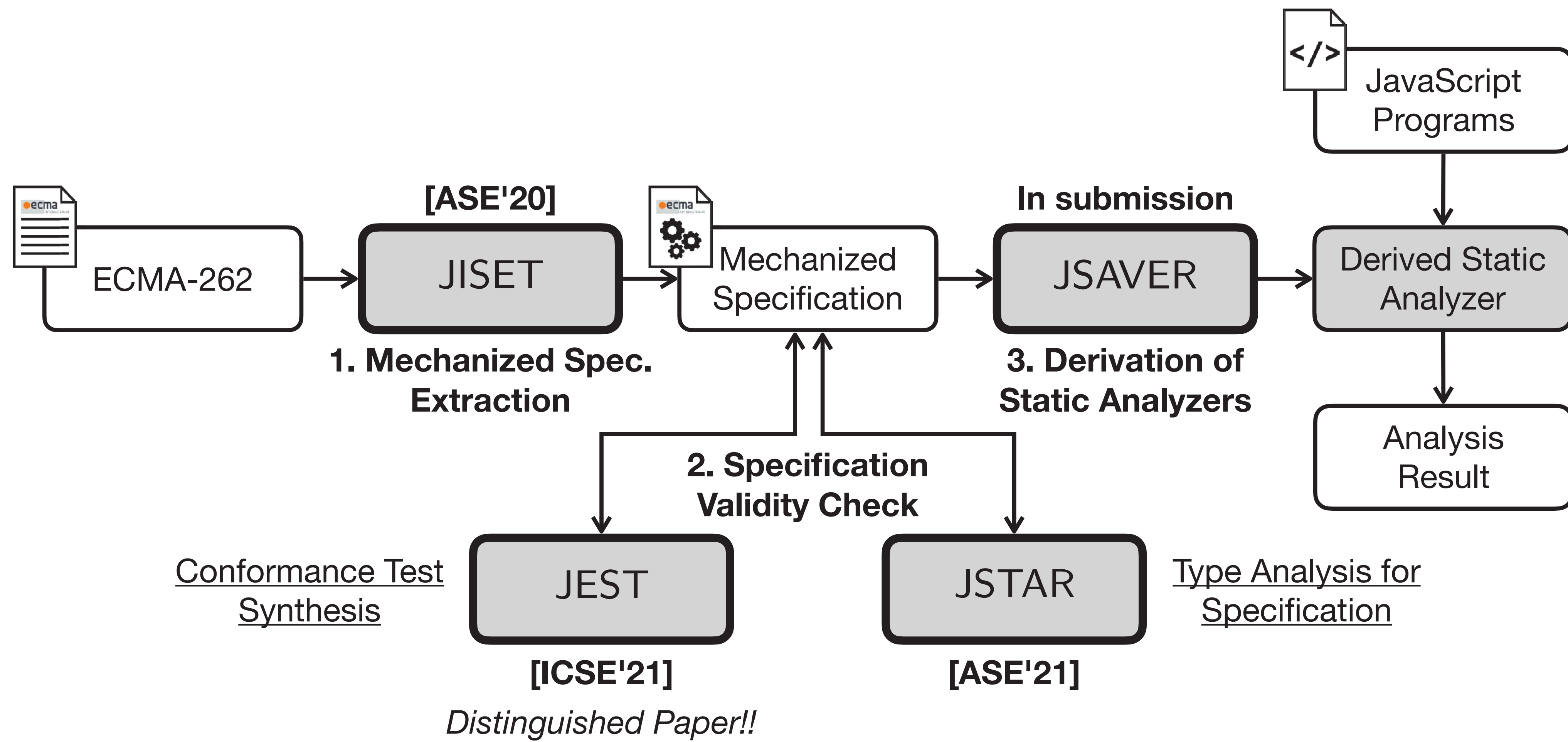
JSAVER - Static Analyzer Derivation

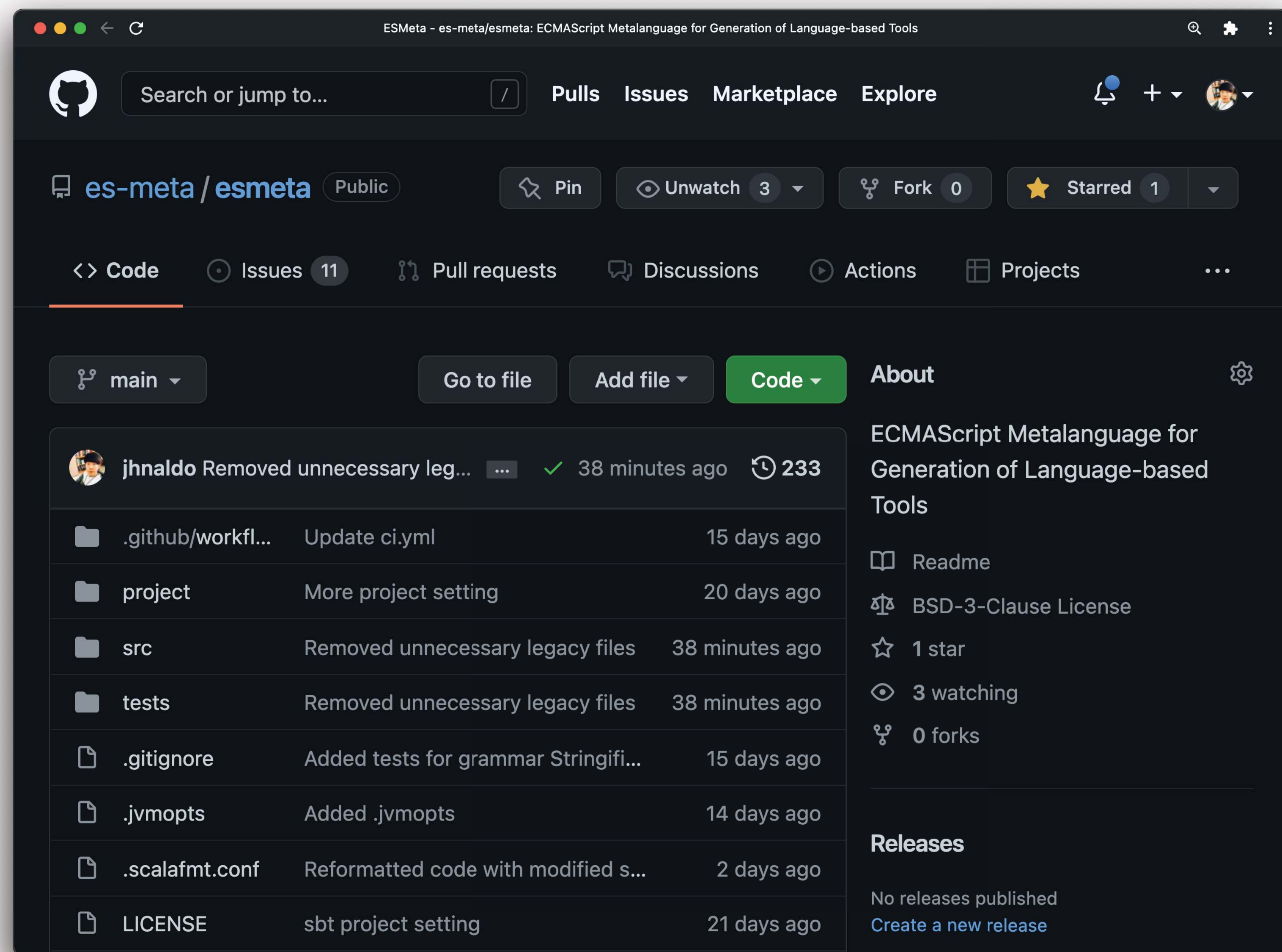


JSAVER - Evaluation

- **Soundness / Precision / Performance**
 - 18,556 applicable tests in Test262
 - 3,903 tests analyzable by all the three analyzers



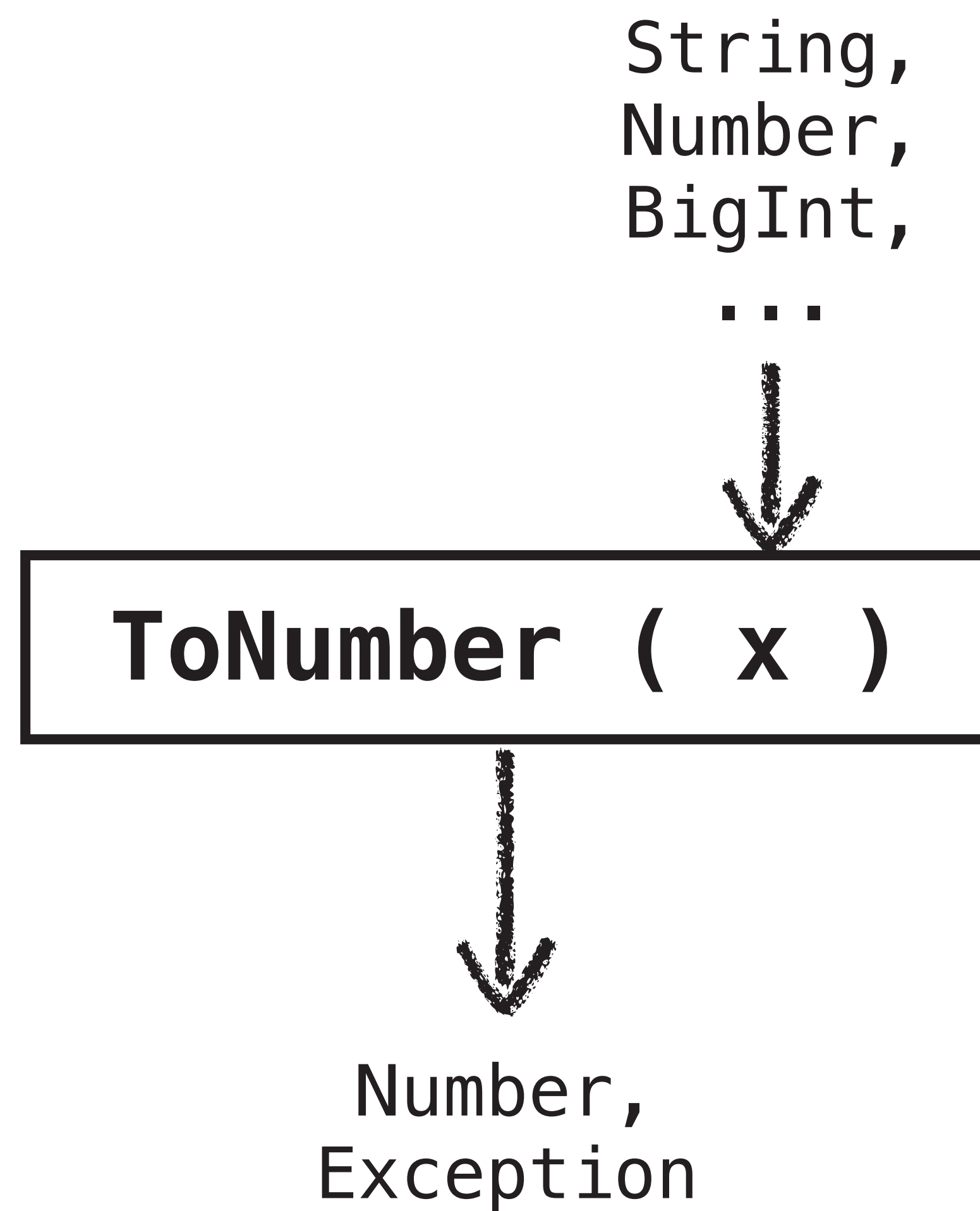




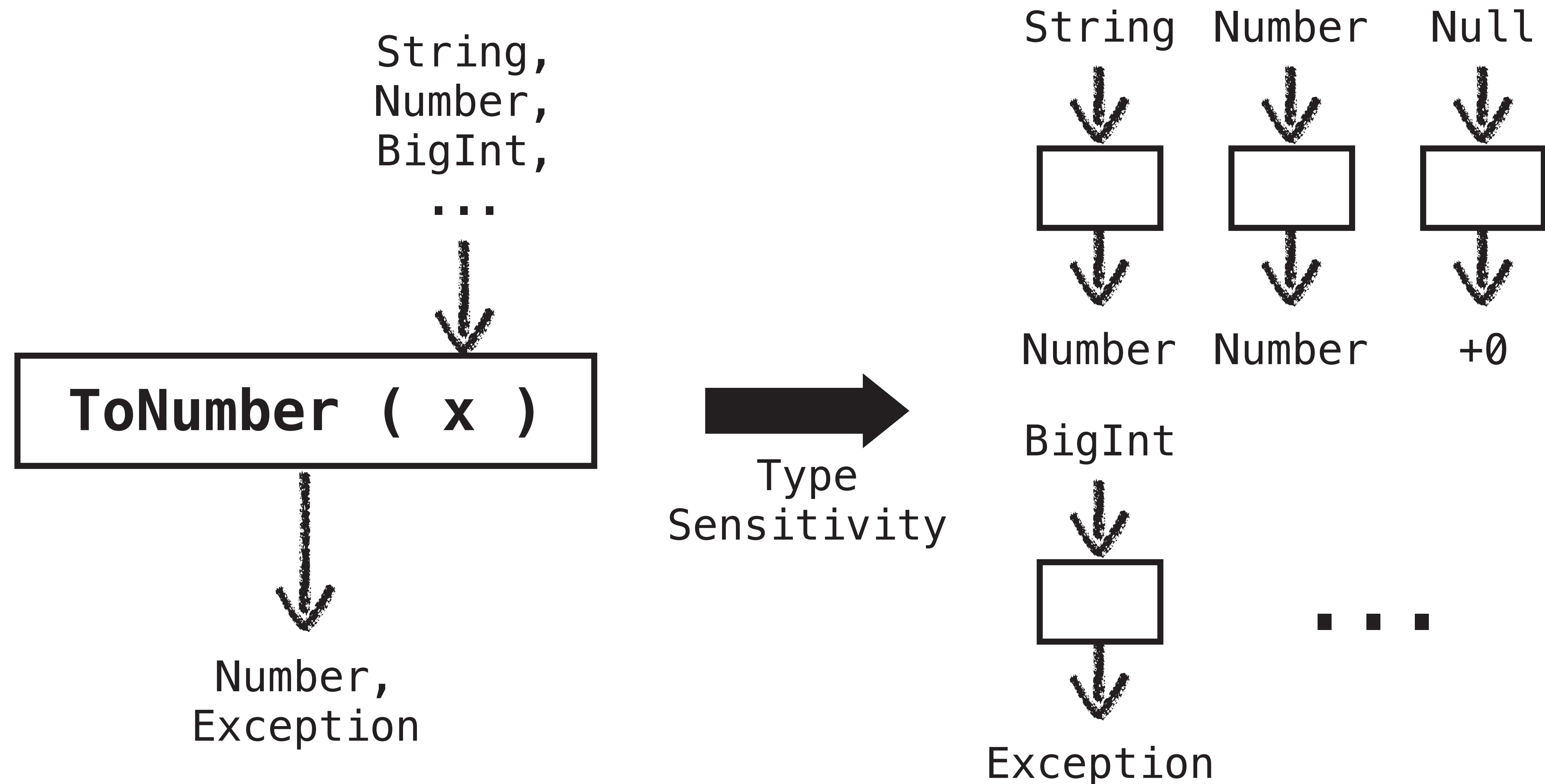
<https://github.com/es-meta/esmeta>

Backup Slides

JSTAR - Precision ↗ - 1) Type Sensitivity



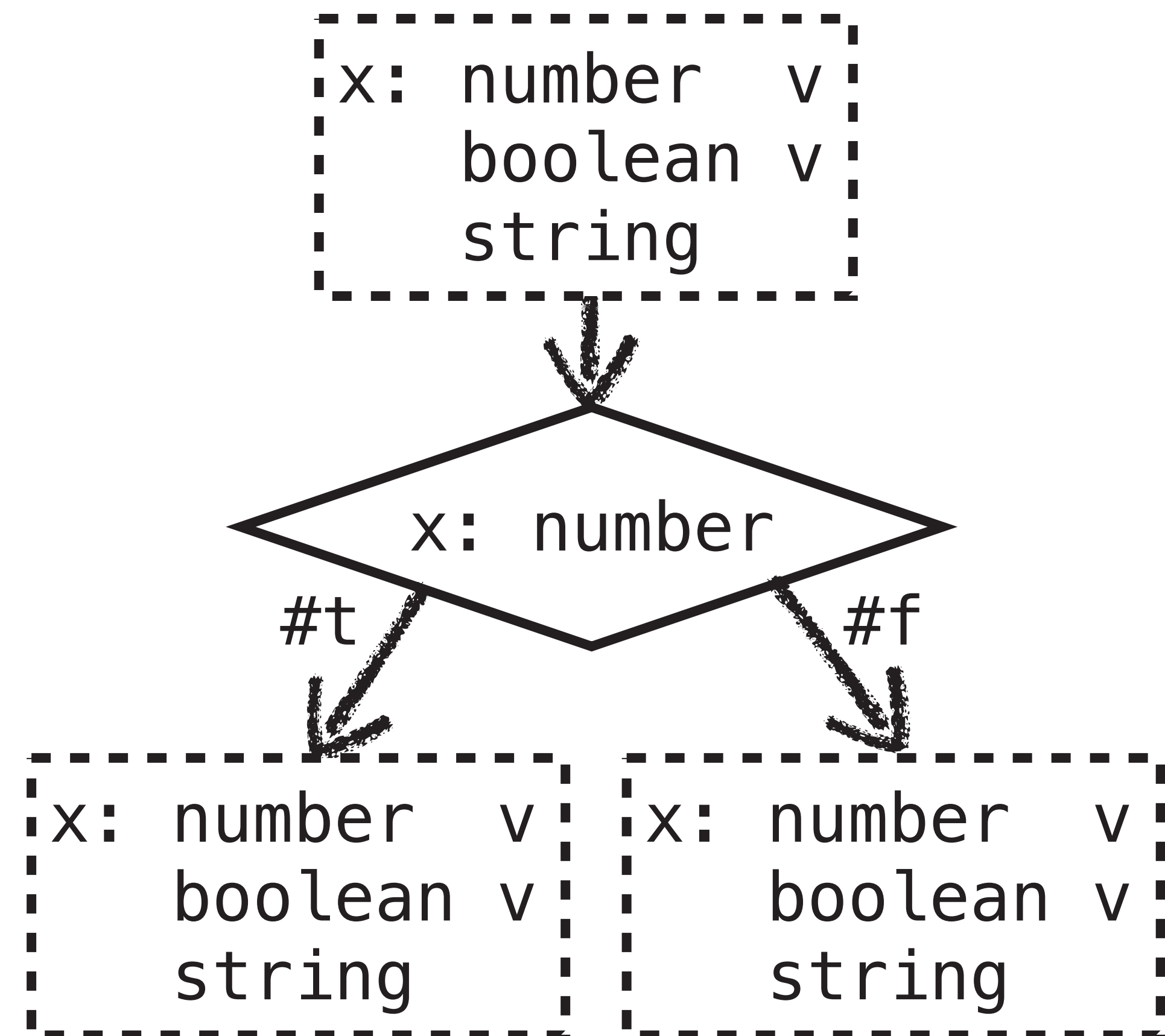
JSTAR - Precision ↗ - 1) Type Sensitivity



JSTAR - Precision $\Uparrow$ - 2) Type Refinement

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

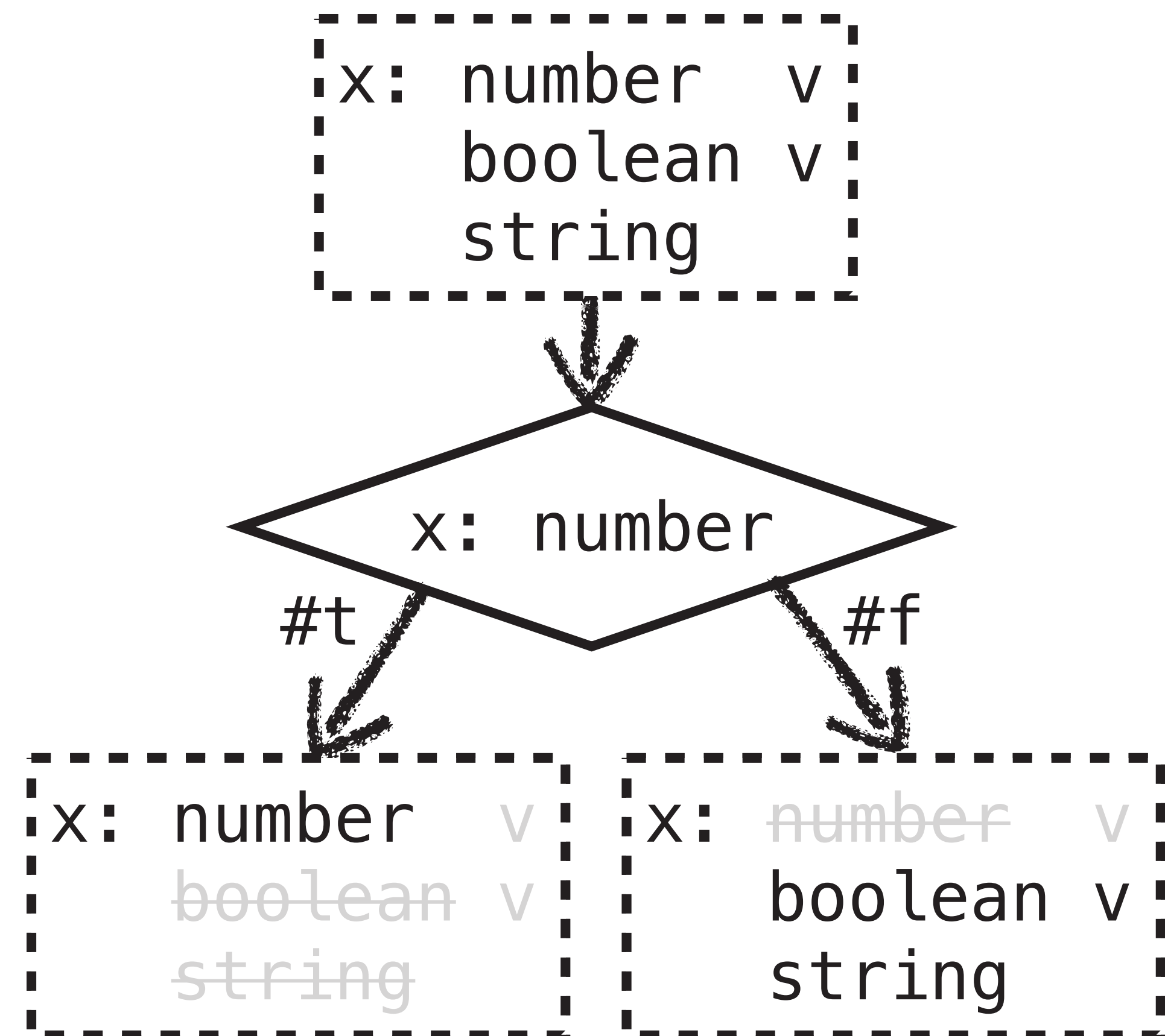
where $\sigma_j^\# = \text{refine}(e_j, b)(\sigma^\#)$ for $j = 0, 1$, $\tau_e^\# = \llbracket e \rrbracket_e^\#(\sigma^\#)$, and $\lfloor \tau^\# \rfloor$ returns $\{\tau\}$ if $\tau^\#$ denotes a singleton type τ , or returns $\emptyset$, otherwise.



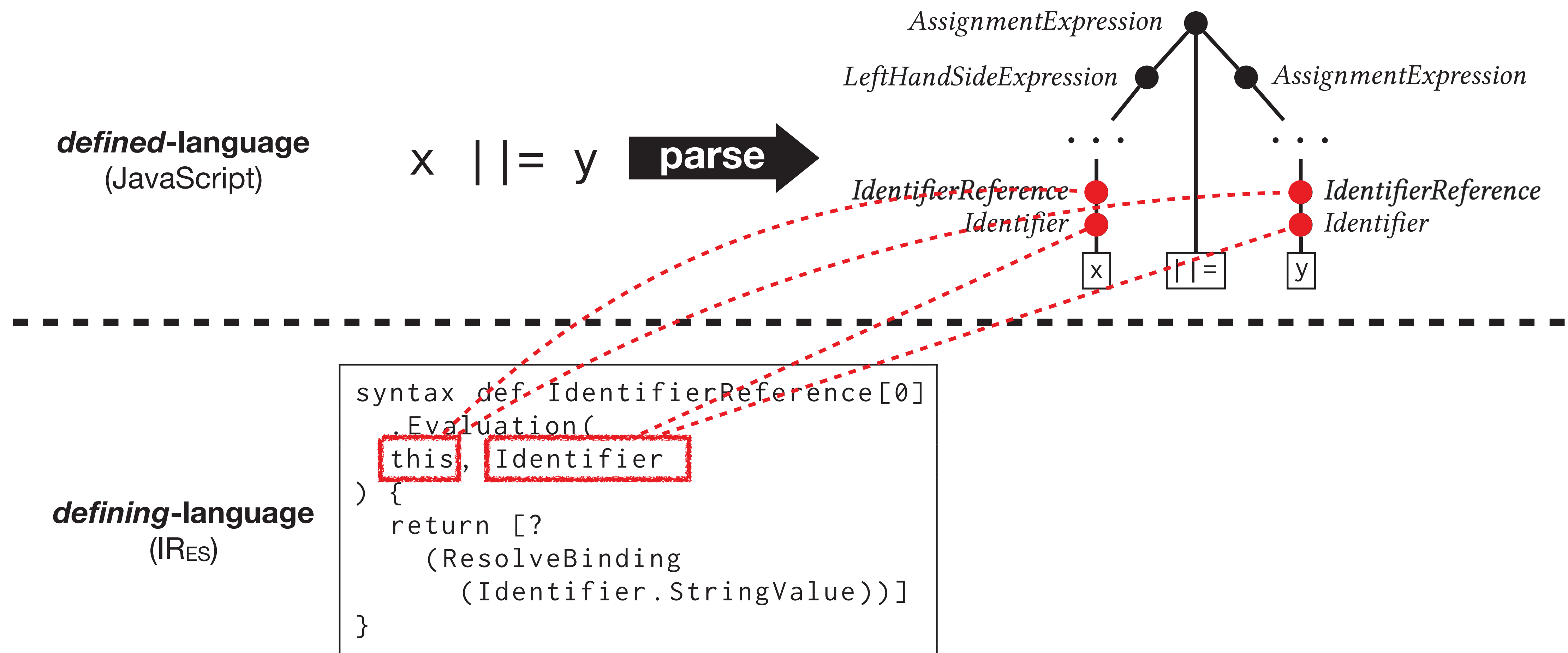
JSTAR - Precision $\Uparrow$ - 2) Type Refinement

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

where $\sigma_j^\# = \text{refine}(e_j, b)(\sigma^\#)$ for $j = 0, 1$, $\tau_e^\# = \llbracket e \rrbracket_e^\#(\sigma^\#)$, and $\lfloor \tau^\# \rfloor$ returns $\{\tau\}$ if $\tau^\#$ denotes a singleton type τ , or returns $\emptyset$, otherwise.



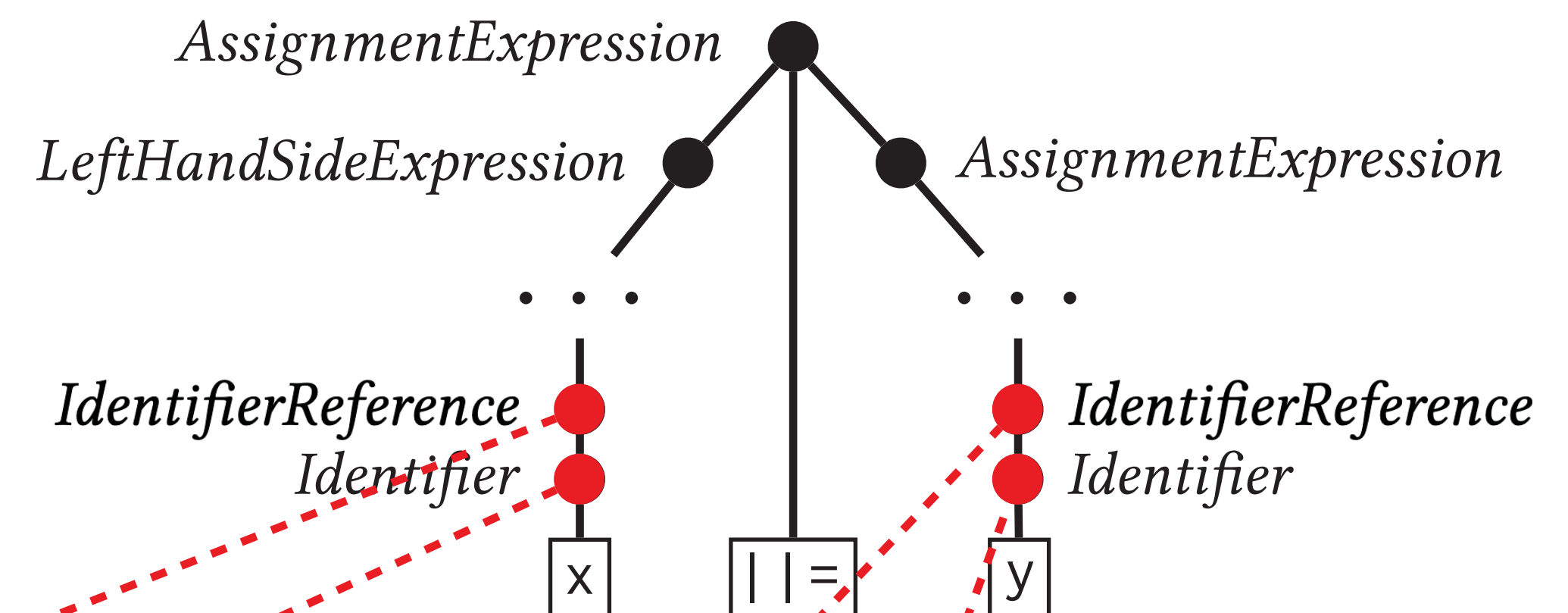
JSAVER - AST Sensitivity



JSAVER - AST Sensitivity

defined-language
(JavaScript)

`x || = y` **parse** →



defining-language
(IR_{ES})

```

syntax def IdentifierReference[0]
  .Evaluation(
    this, Identifier
  ) {
    return [?
      (ResolveBinding
        (Identifier.StringValue))]
  }
    
```

this = AST of `x`

```

syntax def IdentifierReference[0]
  .Evaluation(
    this, Identifier
  ) {
    return [?
      (ResolveBinding
        (Identifier.StringValue))]
  }
    
```

this = AST of `y`

JSAVER - AST Sensitivity

defined-language (JavaScript)	defining-language (IR _{ES})
flow-sensitivity	$\delta^{\text{js-flow}}(t_{\perp}) = \{\sigma = (_, _, \bar{c}, _) \in \mathbb{S} \mid \text{ast}(\bar{c}) = t_{\perp}\}$
k-callsite sensitivity	$\delta^{\text{js-k-cfa}}([t_1, \dots, t_n]) = \{\sigma = (_, _, \bar{c}, _) \in \mathbb{S} \mid$ $n \leq k \wedge (n = k \vee \text{js-ctxt}^{n+1}(\bar{c}) = \perp) \wedge$ $\forall 1 \leq i \leq n. \text{ast} \circ \text{js-ctxt}^i(\bar{c}) = t_i\}$