JavaScript Static Analysis for Evolving Language Specifications

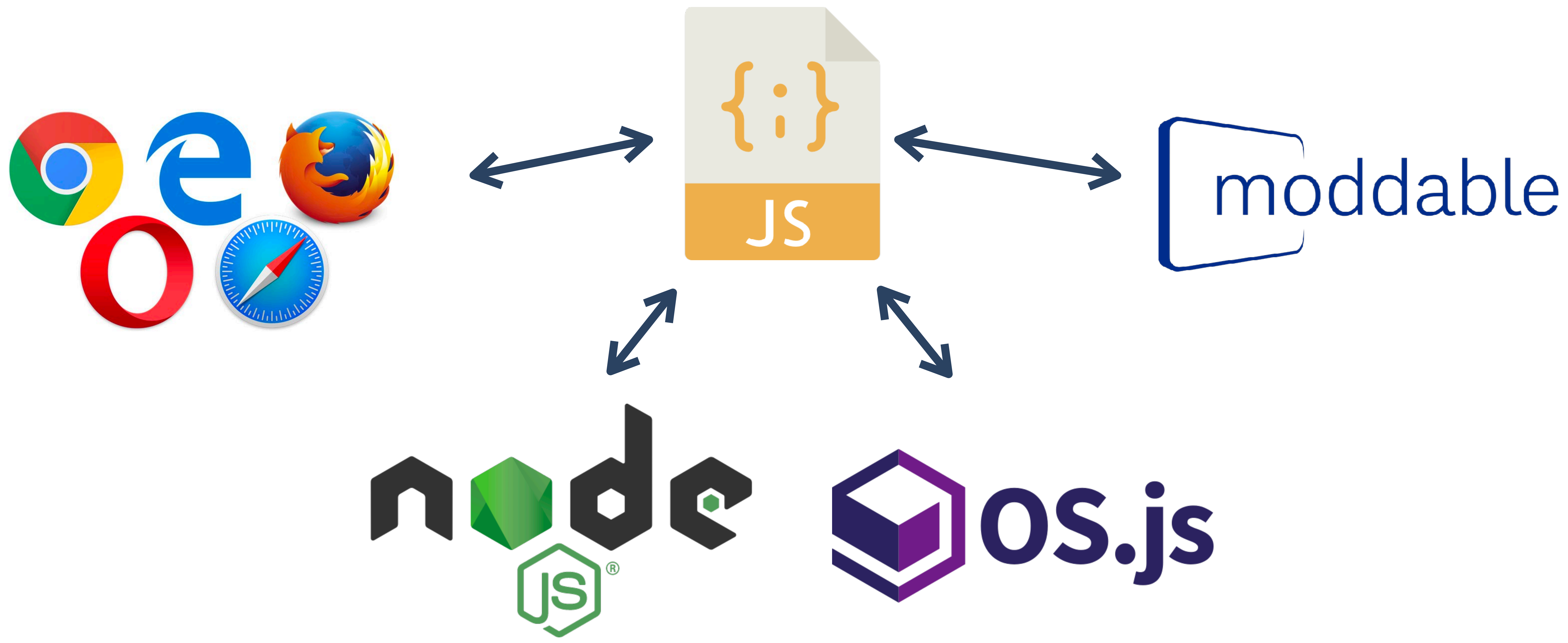
Jihyeok Park

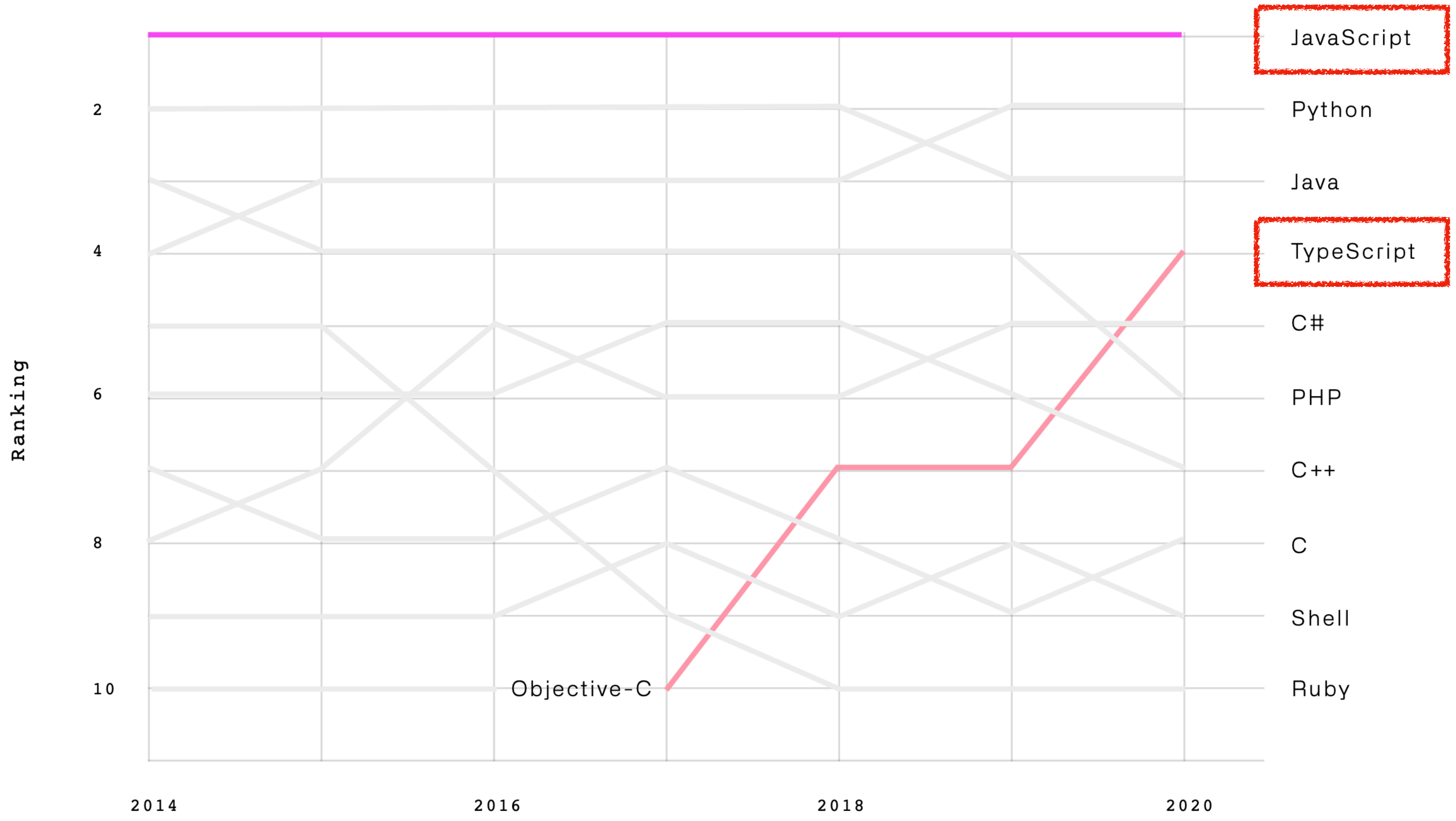
PLRG @ KAIST

KAIST PL

September 17, 2021

JavaScript is Everywhere





https://octoverse.github.com/

JavaScript Complex Semantics

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

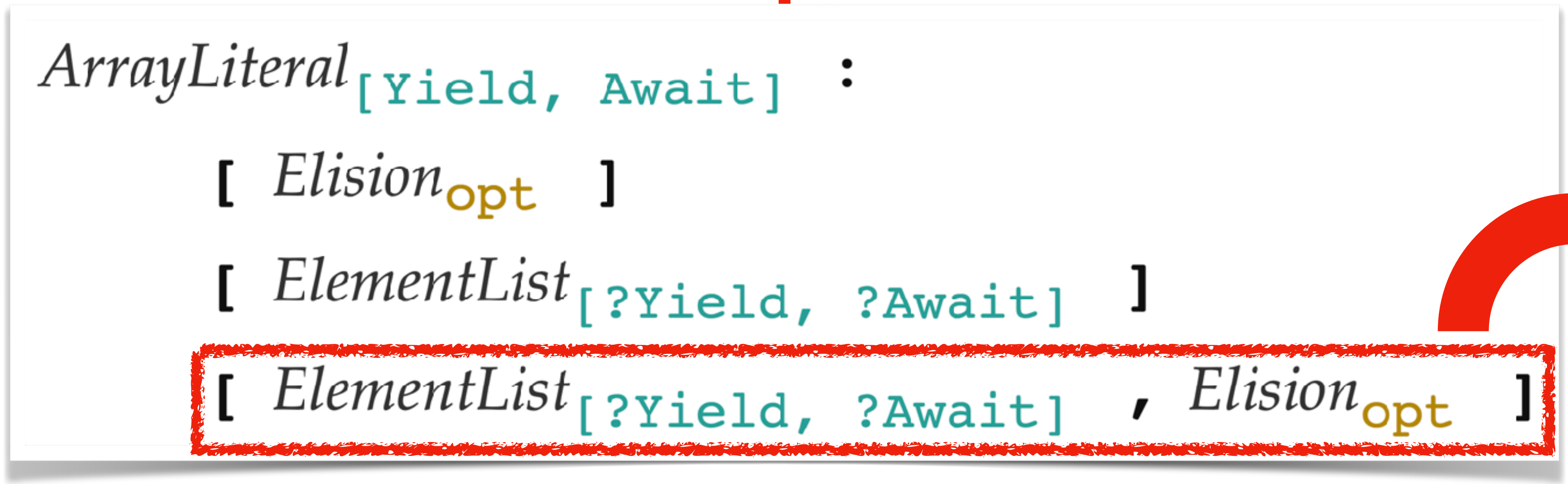
Always return `false`?

NO!!

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

ECMAScript: JavaScript Specification

Syntax



ArrayLiteral`[Yield, Await]` :

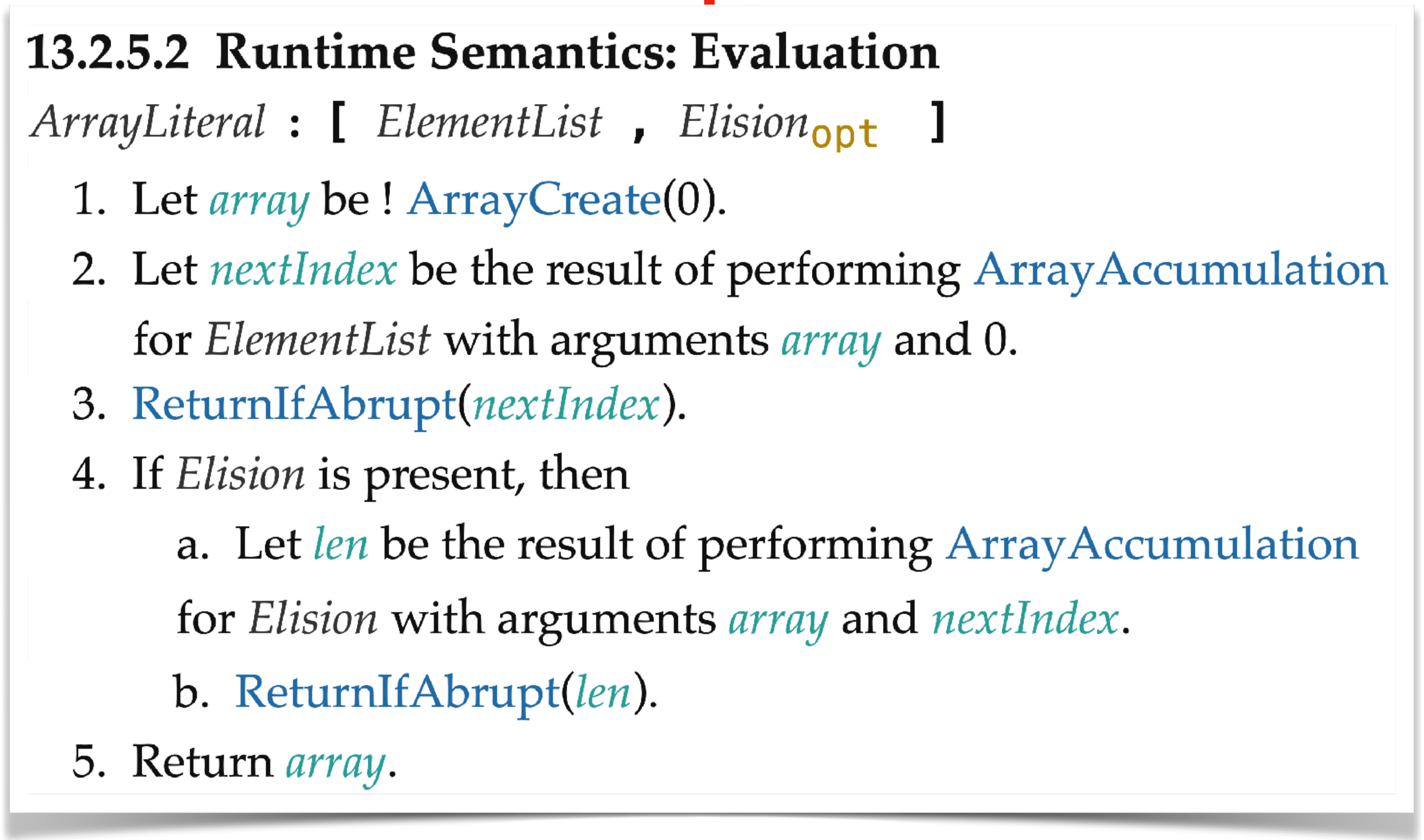
  `[` *Elision*`opt` `]`

  `[` *ElementList*`[?Yield, ?Await]` `]`

  `[` *ElementList*`[?Yield, ?Await]` `,` *Elision*`opt` `]`

The production of *ArrayLiteral* in ES12

Semantics



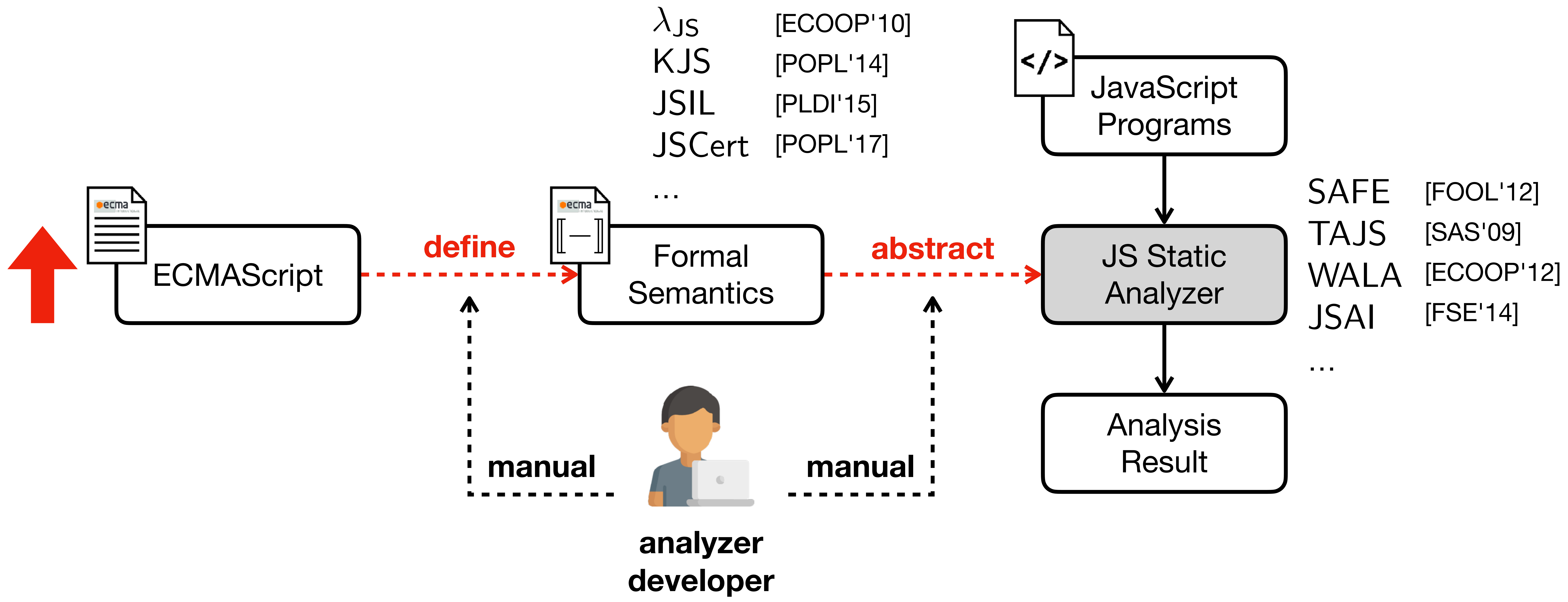
13.2.5.2 Runtime Semantics: Evaluation

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

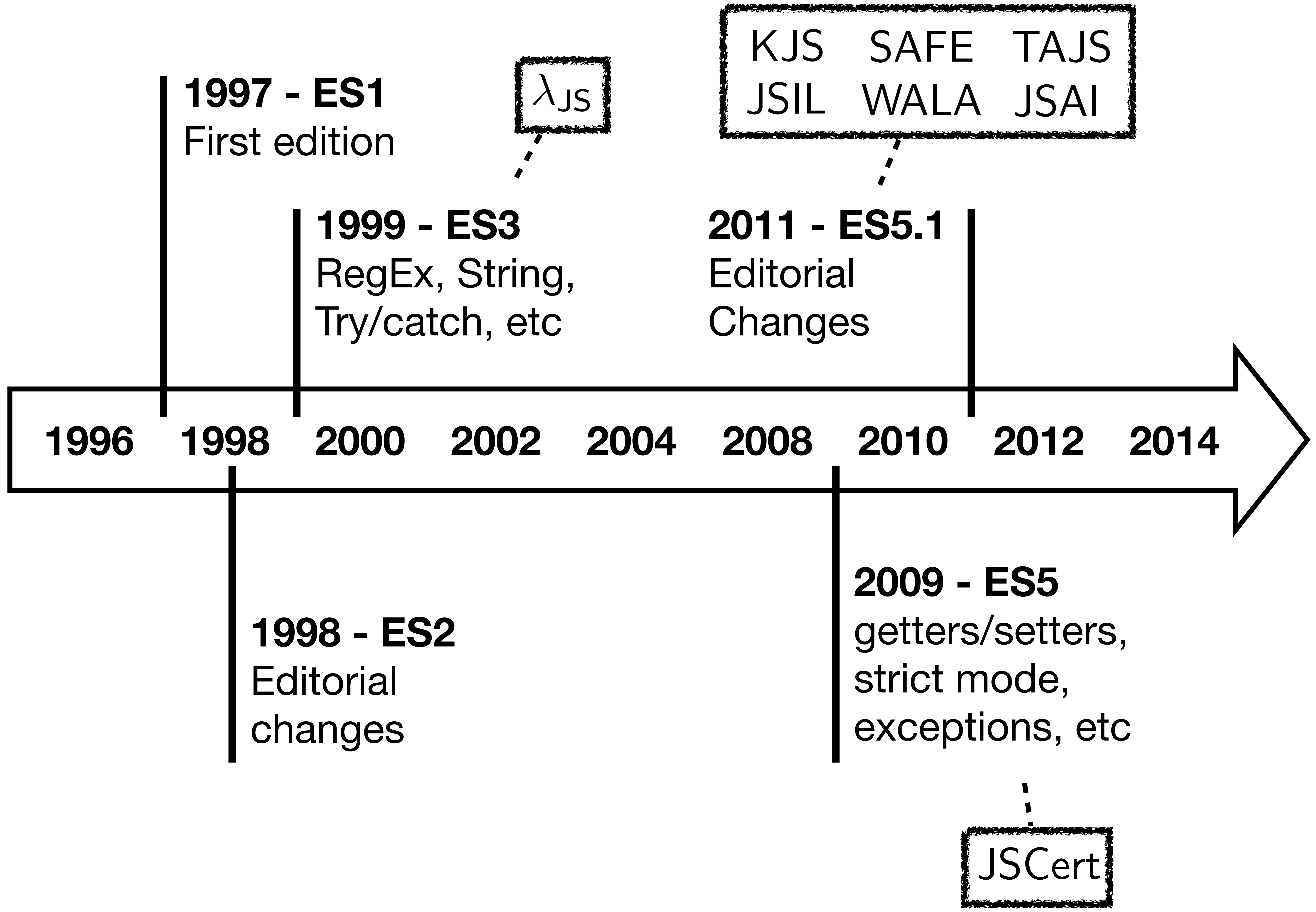
```
1. Let array be ! ArrayCreate(0).
2. Let nextIndex be the result of performing ArrayAccumulation
   for ElementList with arguments array and 0.
3. ReturnIfAbrupt(nextIndex).
4. If Elision is present, then
    a. Let len be the result of performing ArrayAccumulation
       for Elision with arguments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

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

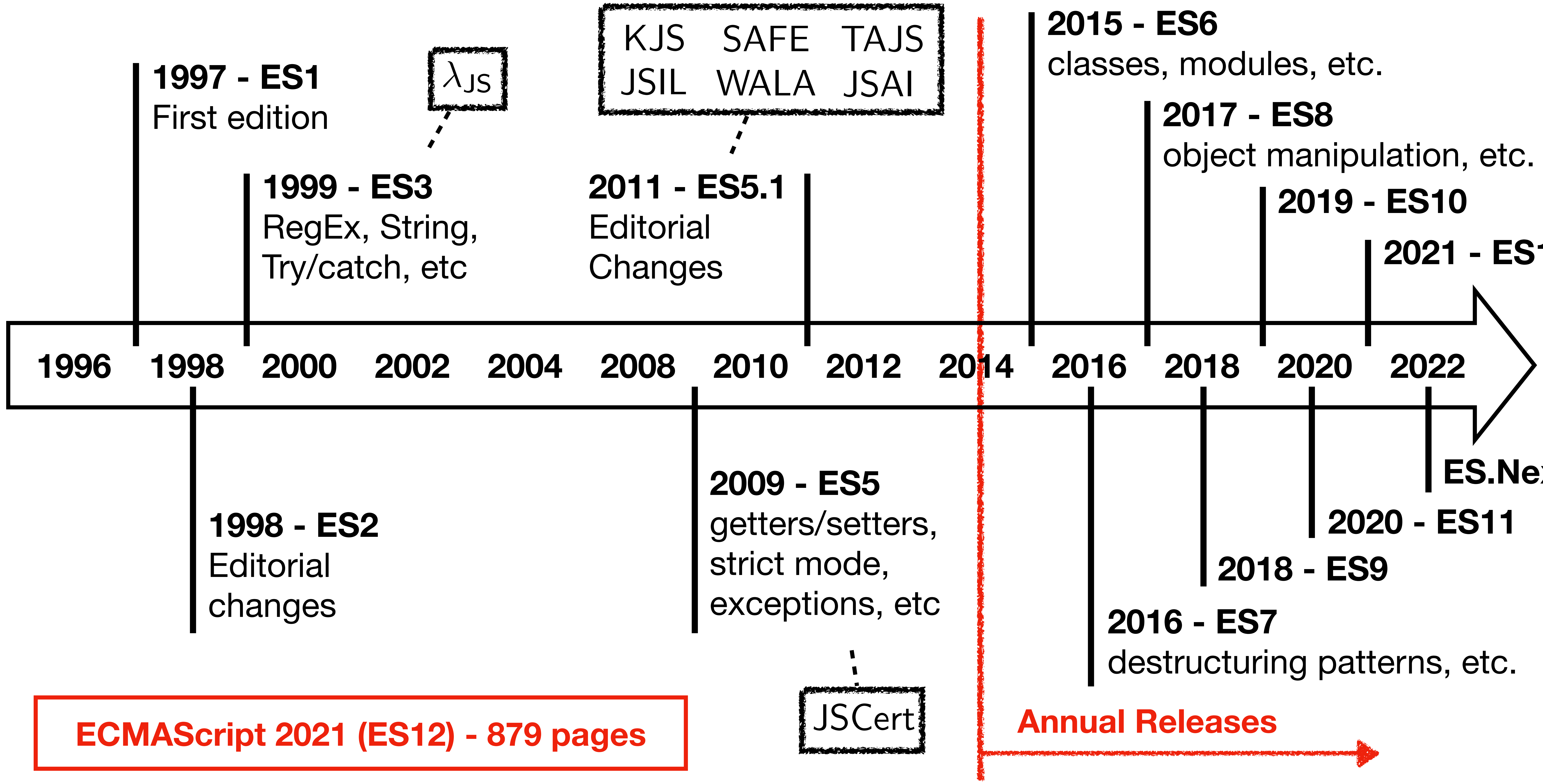
Problem: Manual JavaScript Static Analyzer



Problem: Fast Evolving JavaScript

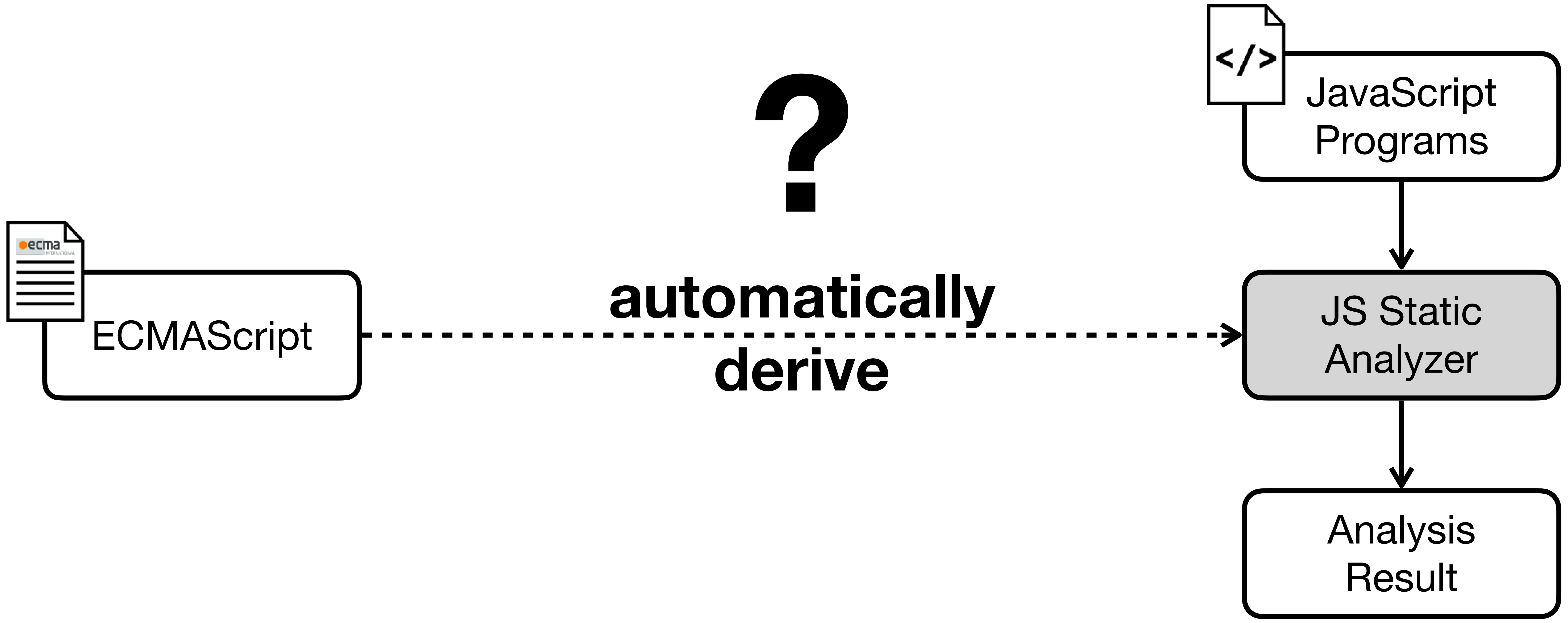


Problem: Fast Evolving JavaScript

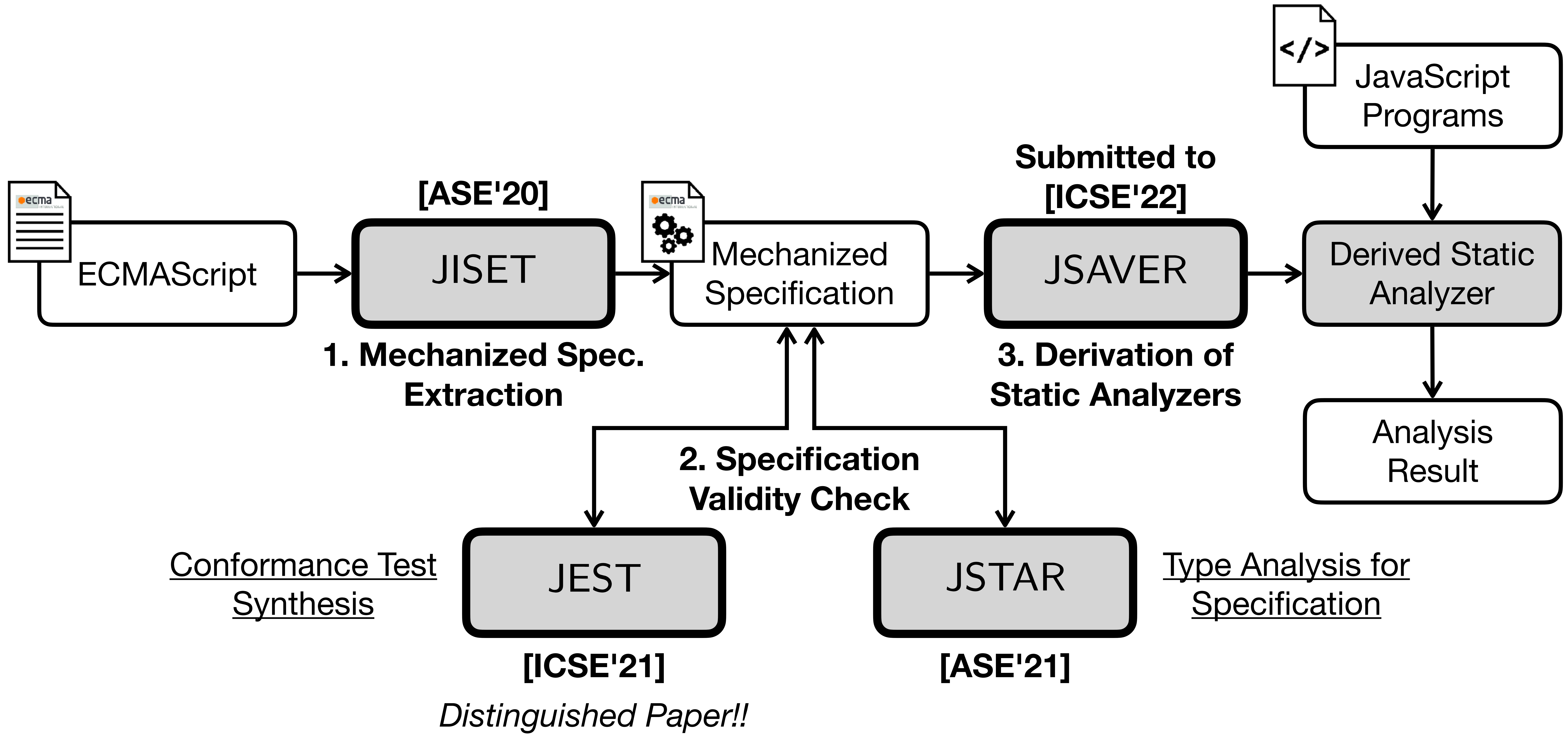


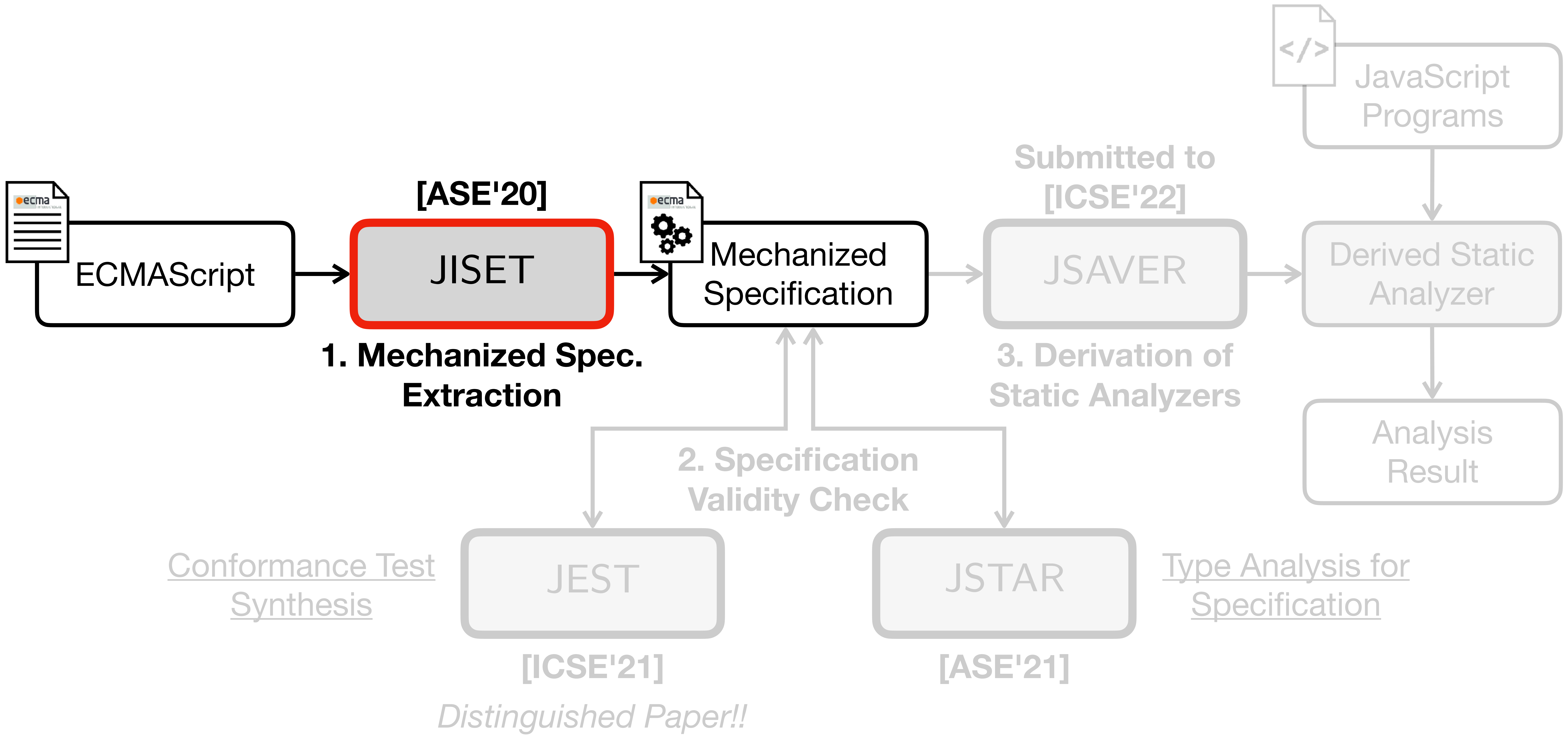
ECMAScript 2021 (ES12) - 879 pages

Main Idea: Deriving Static Analyzer from Spec.



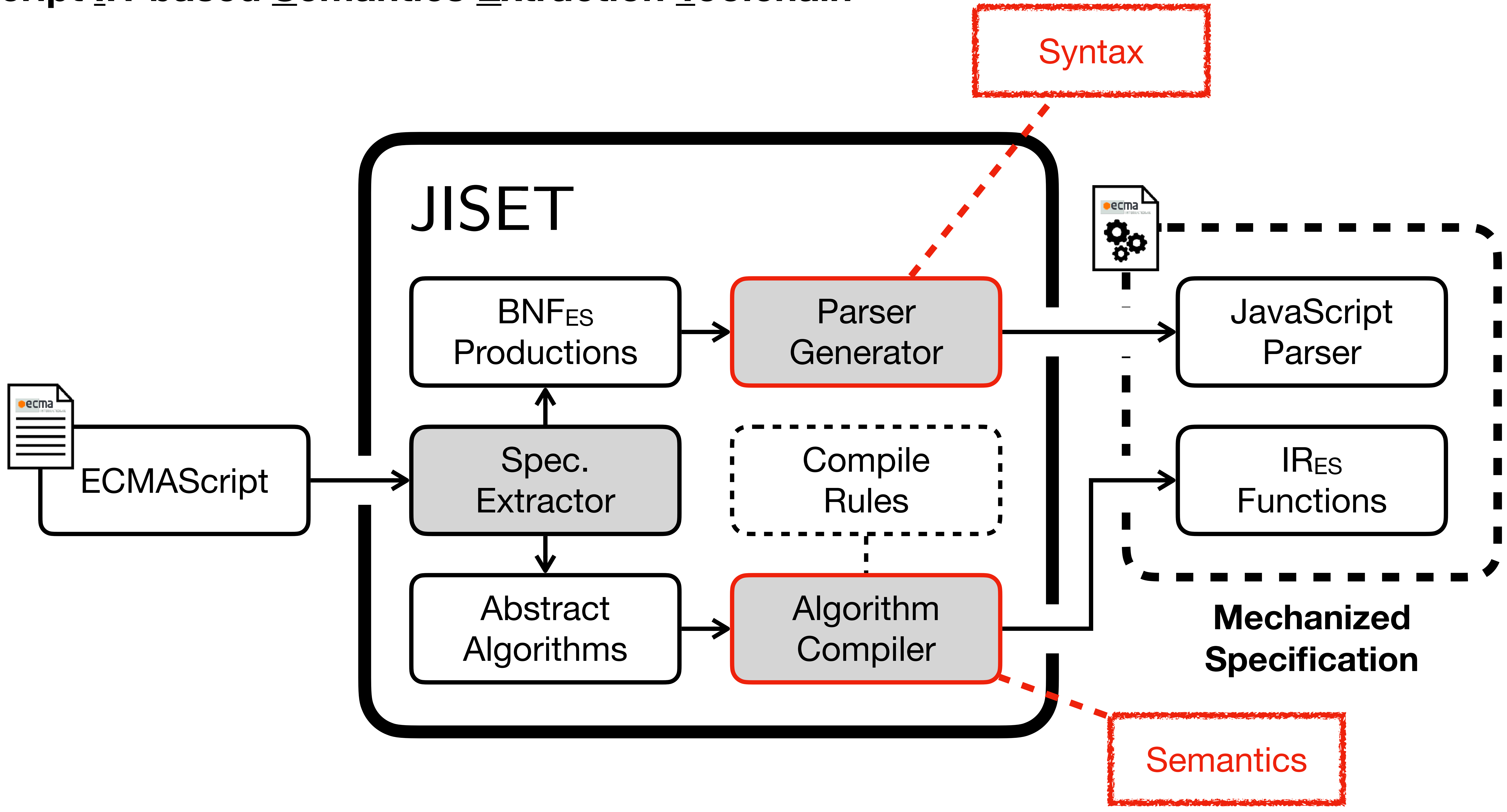
Overall Structure



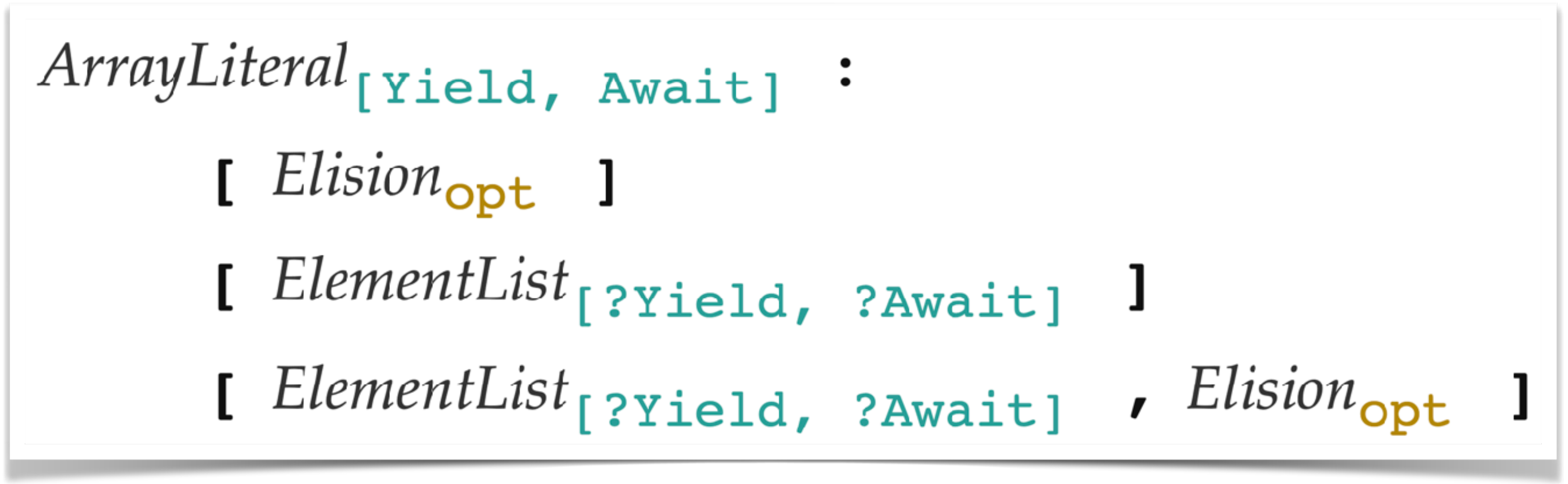

ECMAScript
[ASE'20]
JISET
1. Mechanized Spec. Extraction
Mechanized Specification
Submitted to [ICSE'22]
JSAVER
3. Derivation of Static Analyzers
JavaScript Programs
Derived Static Analyzer
Analysis Result
2. Specification Validity Check
Conformance Test Synthesis
JEST
[ICSE'21]
Distinguished Paper!!
JSTAR
Type Analysis for Specification
[ASE'21]

JISET [ASE'20]

JavaScript IR-based Semantics Extraction Toolchain



JISET - Parser Generator (Syntax)



ArrayLiteral[Yield, Await] :

[*Elision*opt]

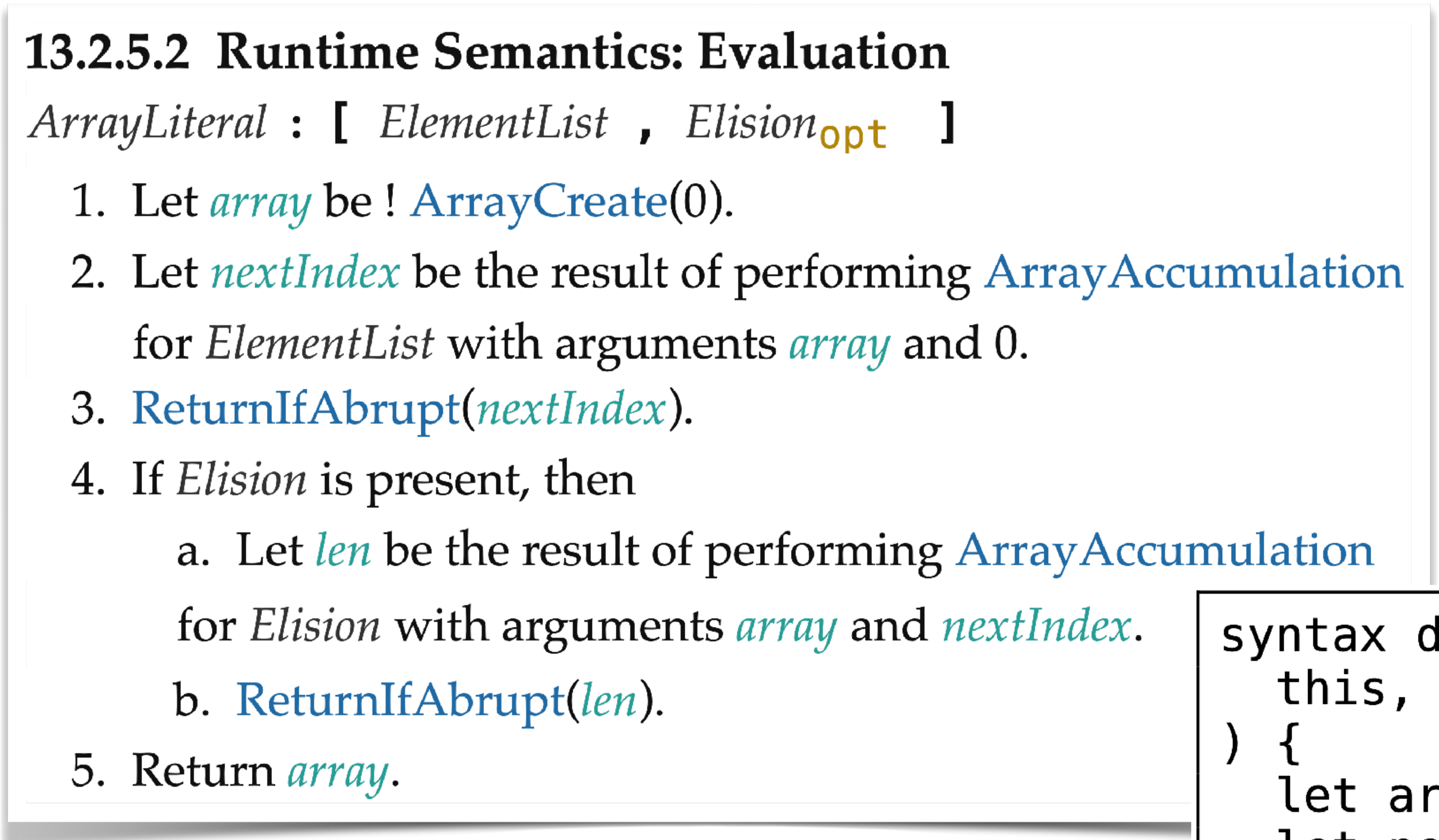
[*ElementList*[?Yield, ?Await]]

[*ElementList*[?Yield, ?Await] , *Elision*opt]

<u>Parsing Expression Grammar</u>
(+ Lookahed Parsing)

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

JISET - Algorithm Compiler (Semantics)



13.2.5.2 Runtime Semantics: Evaluation

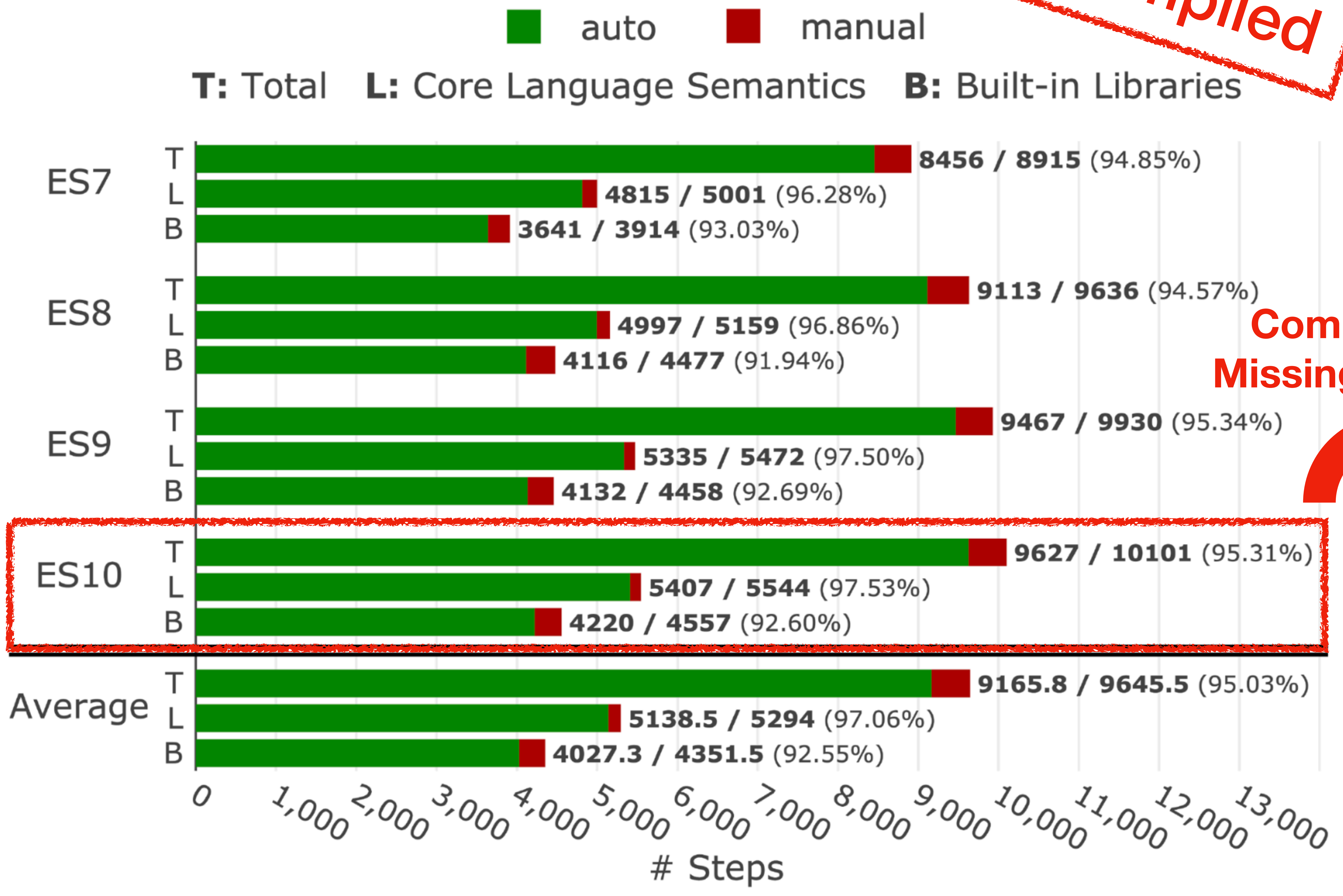
ArrayLiteral **:** **[** *ElementList* **,** *Elision*opt **]**

```
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

```
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
}
```

JISET - Evaluation

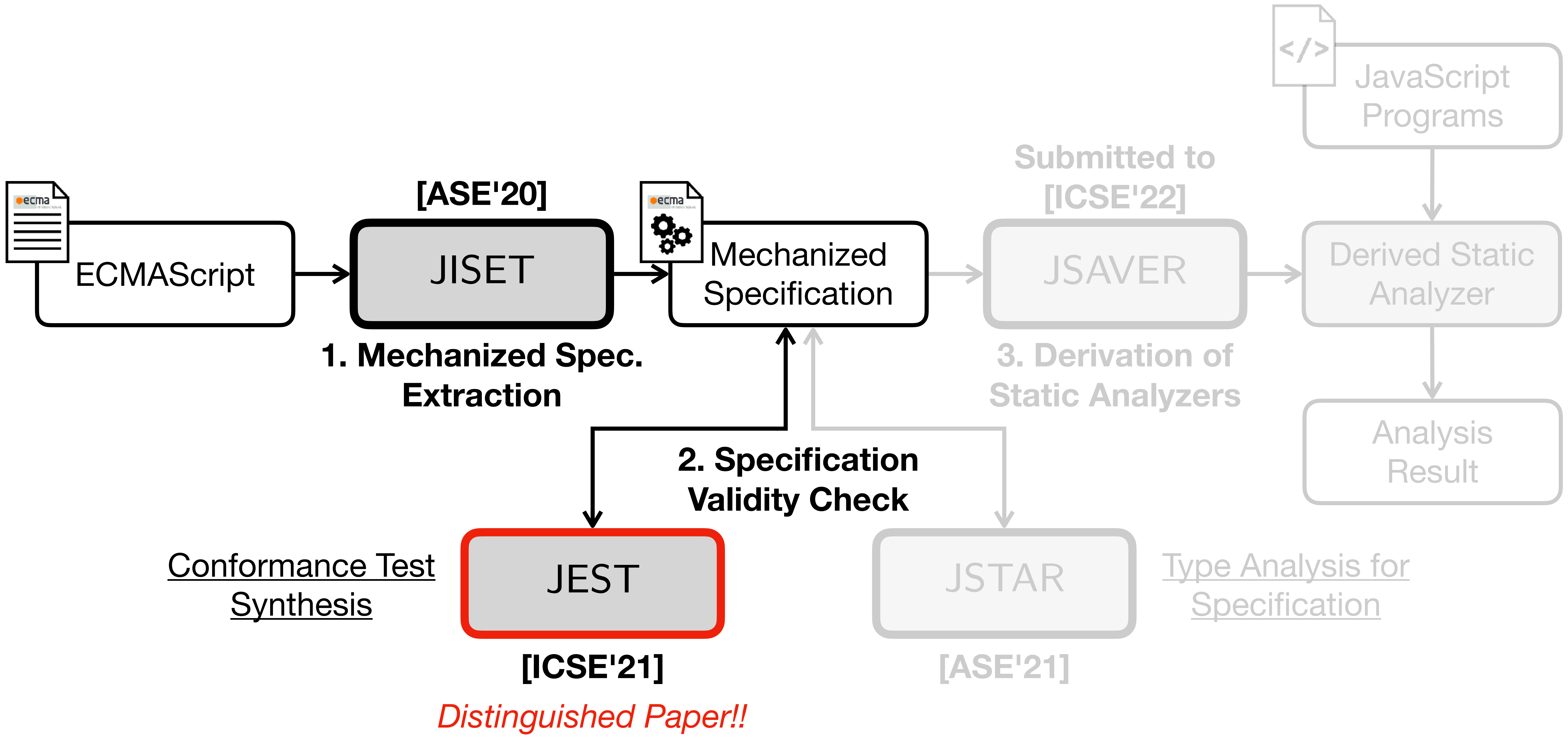


≈ 95% Compiled

Complete Missing Parts

Passed All Tests

- **Test262**
  (Official Conformance Tests)
  - 18,064 applicable tests
- **Parsing tests**
  - Passed all 18,064 tests
- **Evaluation Tests**
  - Passed all 18,064 tests


[ASE'20]
ECMAScript
JISET
Mechanized Specification
Submitted to [ICSE'22]
JSAVER
JavaScript Programs
Derived Static Analyzer
Analysis Result
1. Mechanized Spec. Extraction
3. Derivation of Static Analyzers
2. Specification Validity Check
Conformance Test Synthesis
JEST
JSTAR
Type Analysis for Specification
[ICSE'21]
Distinguished Paper!!
[ASE'21]

JEST - Conformance with Engines

ecma INTERNATIONAL

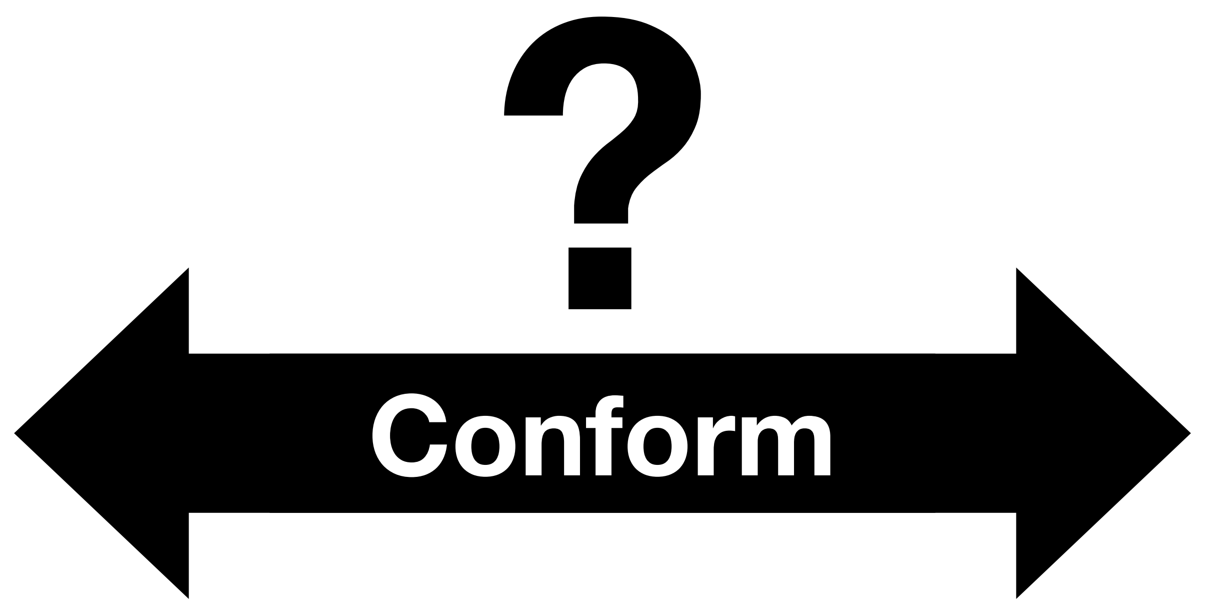
Standard ECMA-262

11th Edition / June 2020

ECMAScript® 2020 Language Specification

Standard

ECMAScript



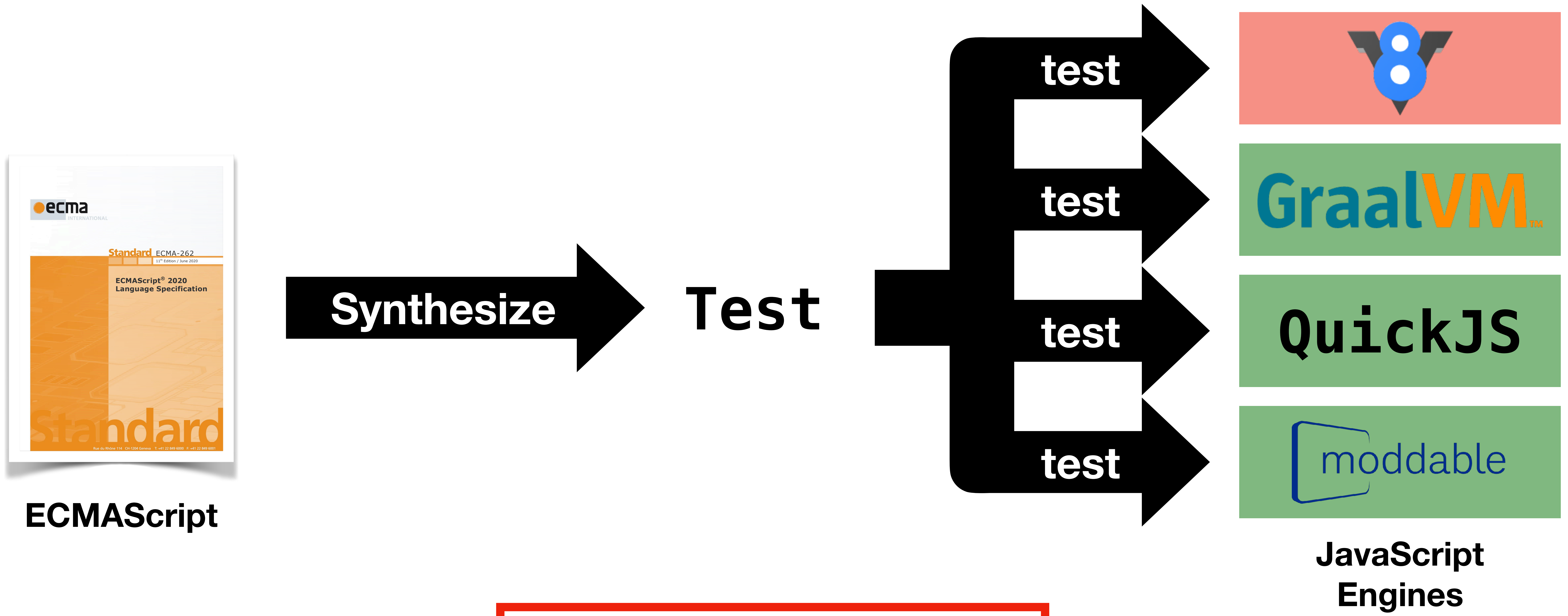
GraalVM™

QuickJS

moddable

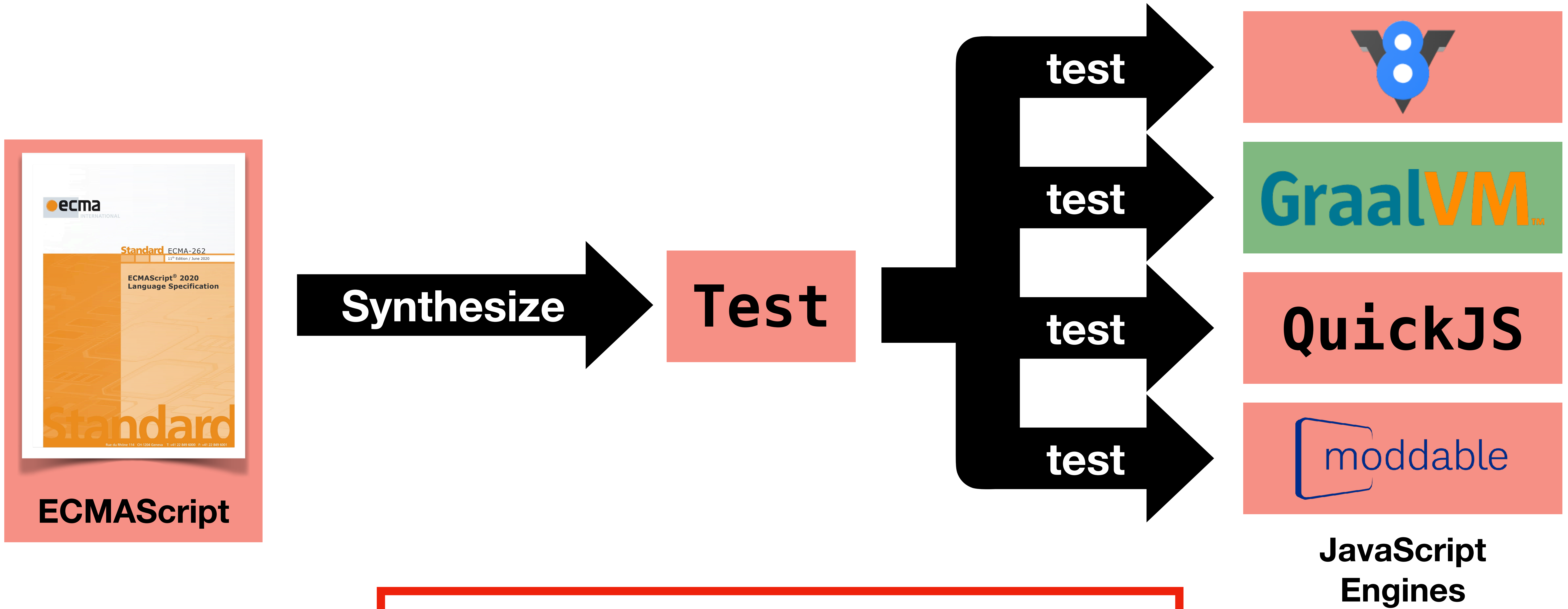
JavaScript Engines

JEST - N+1-version Differential Testing



An engine bug in

JEST - N+1-version Differential Testing

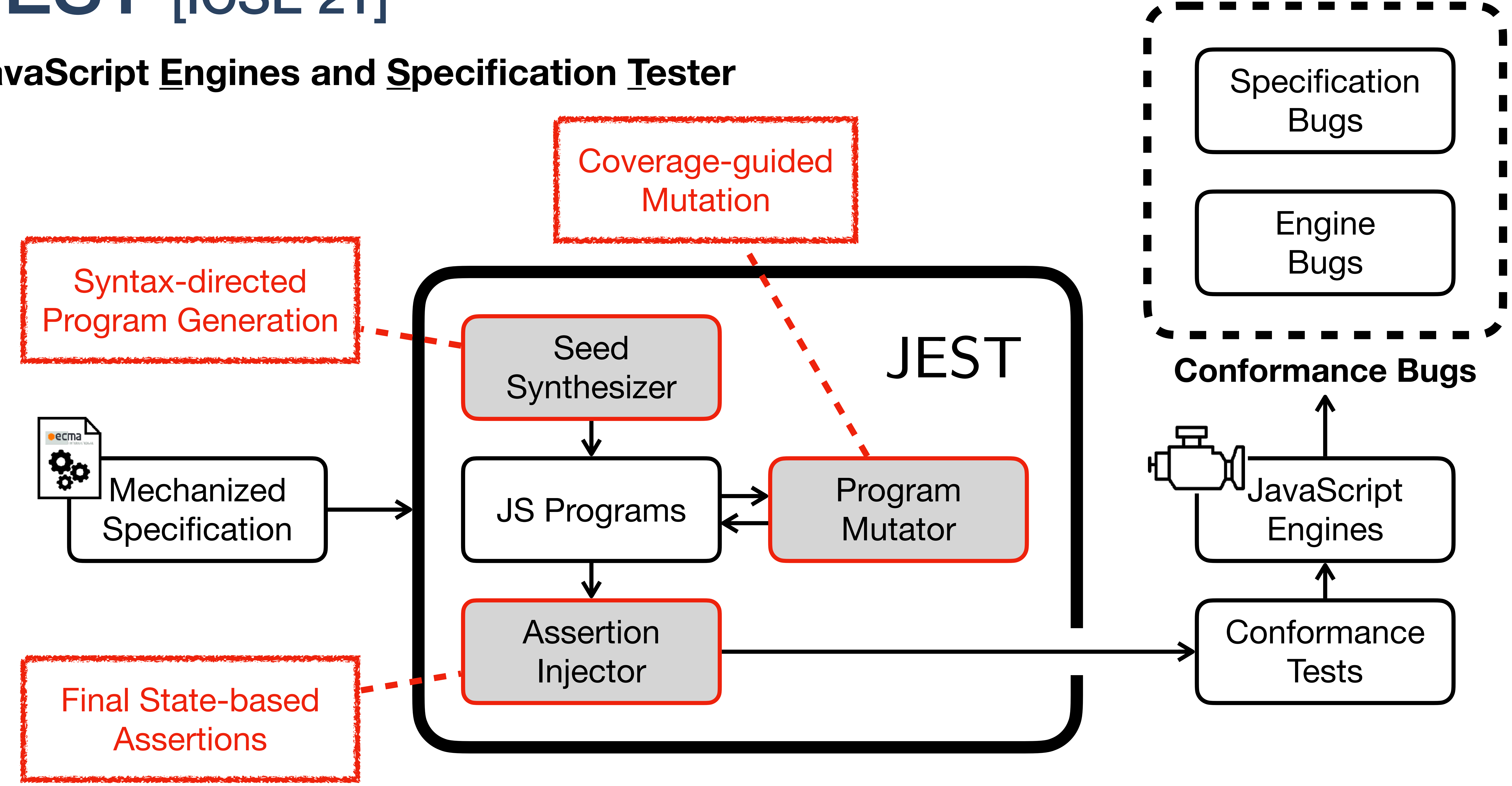


A specification bug in ECMAScript

An engine bug in GraalVM

JEST [ICSE'21]

JavaScript Engines and Specification Tester



JEST - Evaluation

44 Bugs in Engines

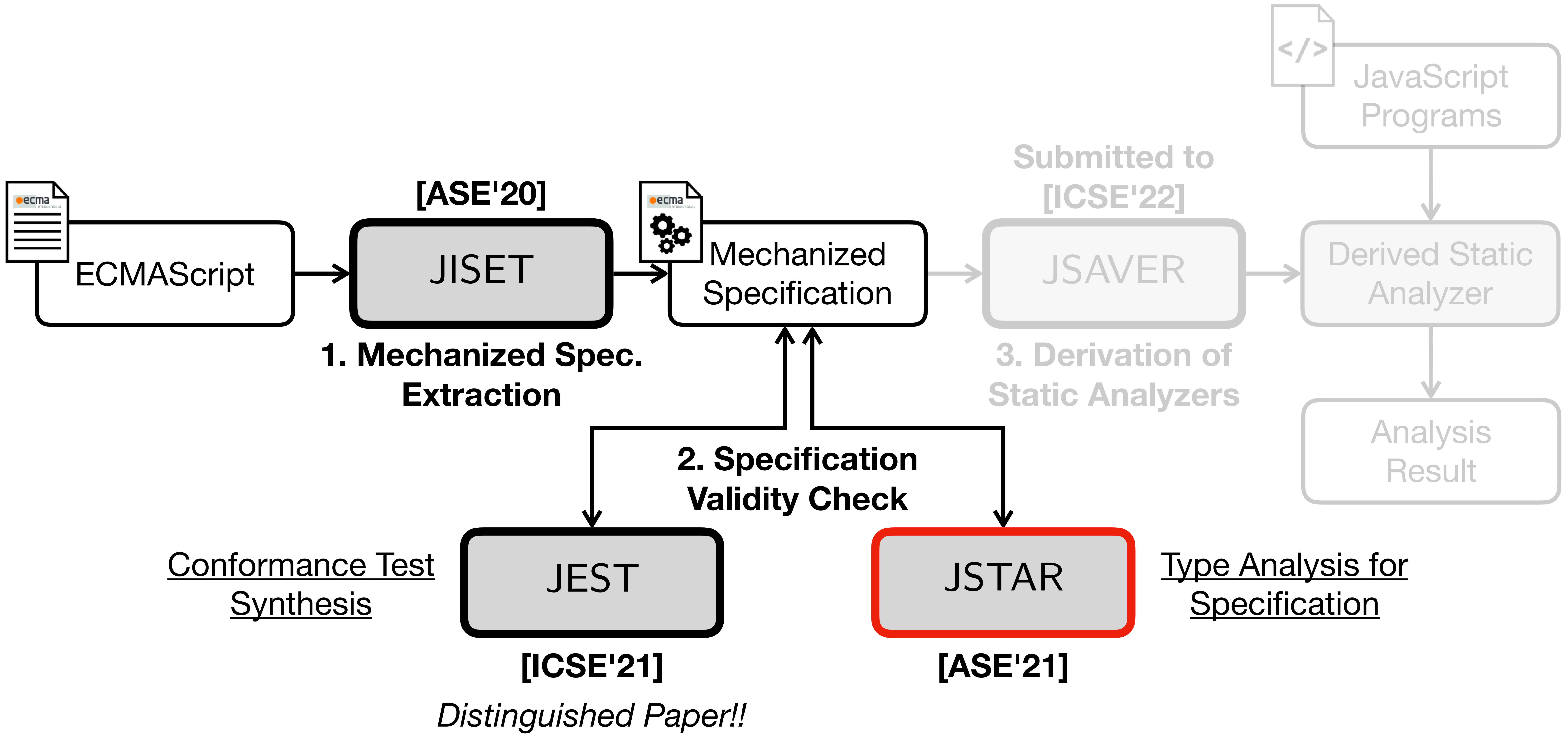
TABLE II: The number of engine bugs detected by JEST

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

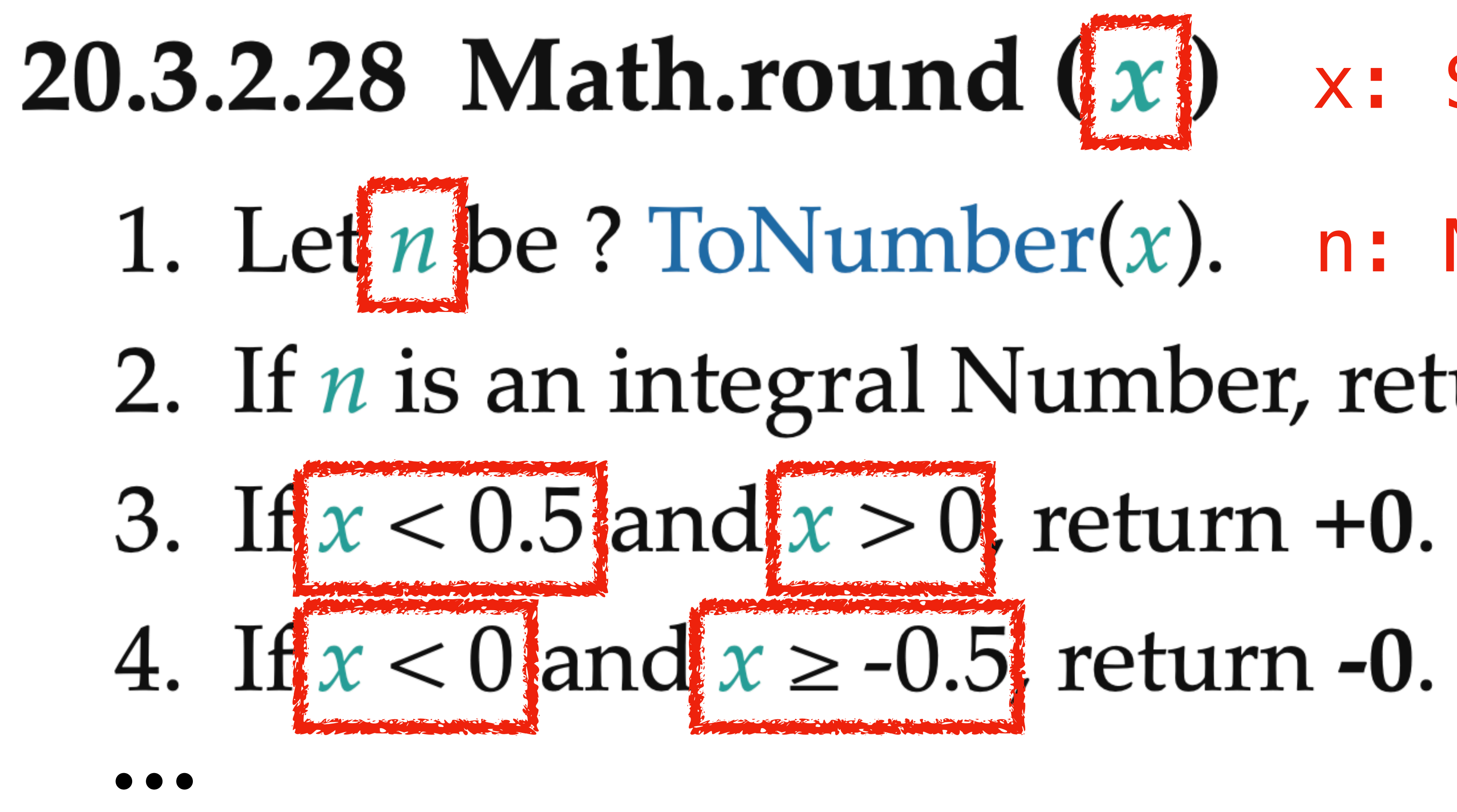
27 Bugs in Spec.

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


ECMAScript
[ASE'20]
JISET
Mechanized Specification
1. Mechanized Spec. Extraction
Submitted to [ICSE'22]
JSAVER
3. Derivation of Static Analyzers
JavaScript Programs
Derived Static Analyzer
Analysis Result
2. Specification Validity Check
Conformance Test Synthesis
JEST
[ICSE'21]
Distinguished Paper!!
JSTAR
Type Analysis for Specification
[ASE'21]

JSTAR - Types in Specification



20.3.2.28 Math.round ($\boxed{x}$)

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

x: String, Boolean, Number, Object, ...

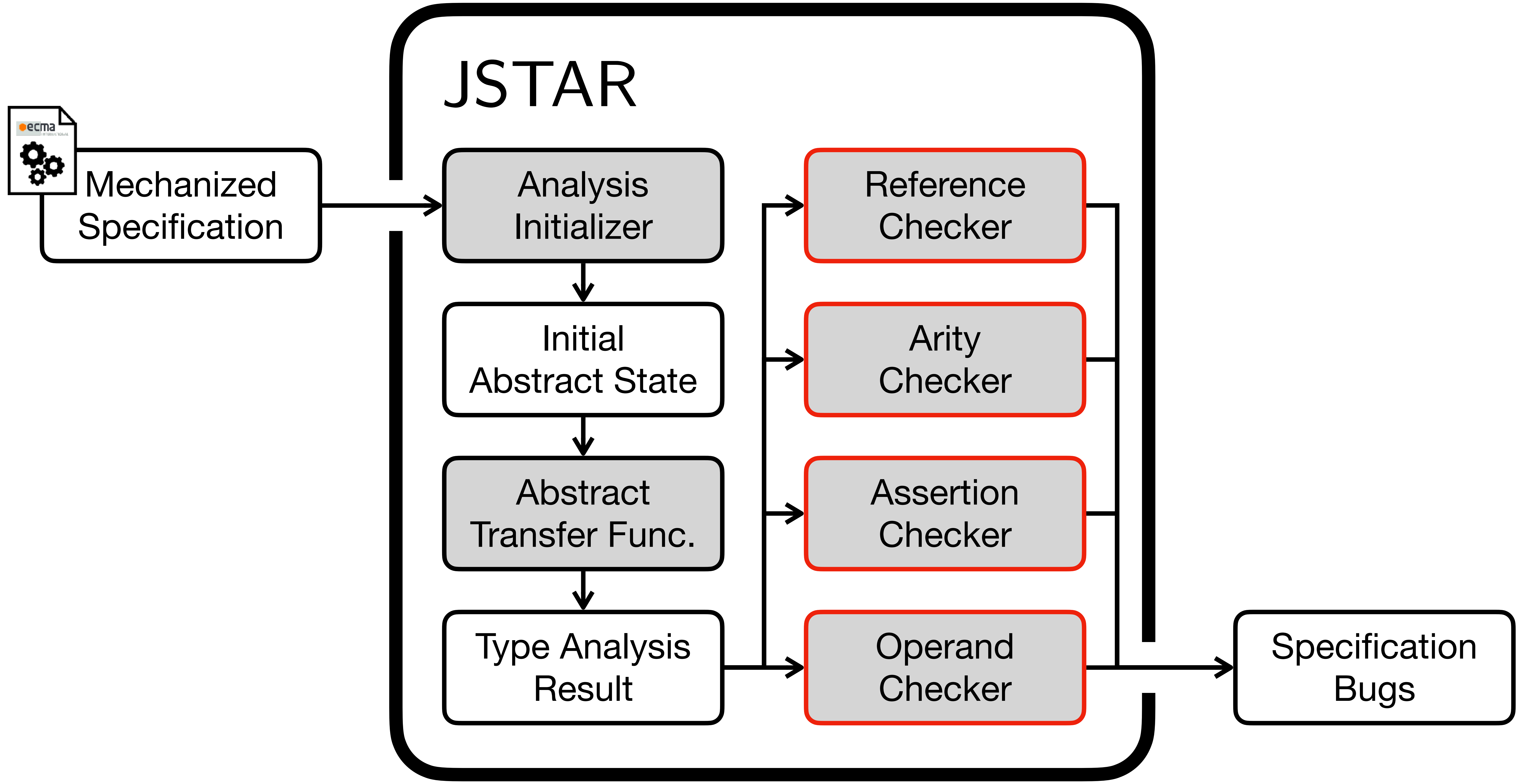
n: Number

Type Mismatch for numeric operator `>`

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

JSTAR [ASE'21]

JavaScript Specification Type Analyzer using Refinement



JSTAR - Evaluation

14 Bugs in Spec.

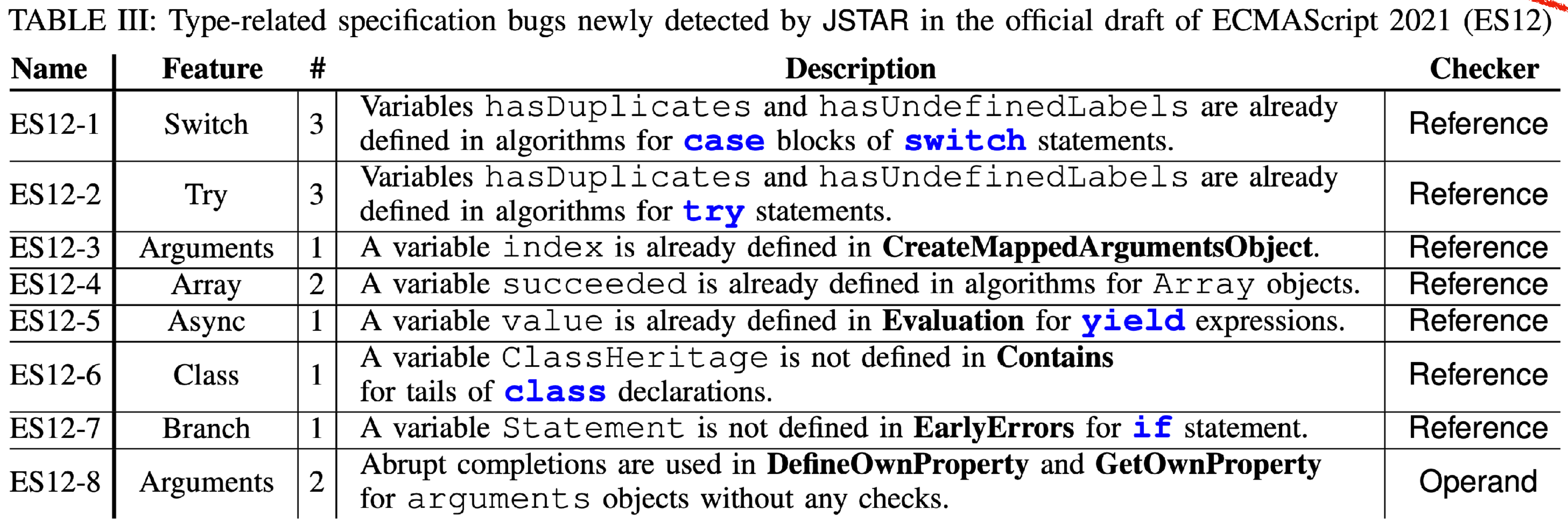
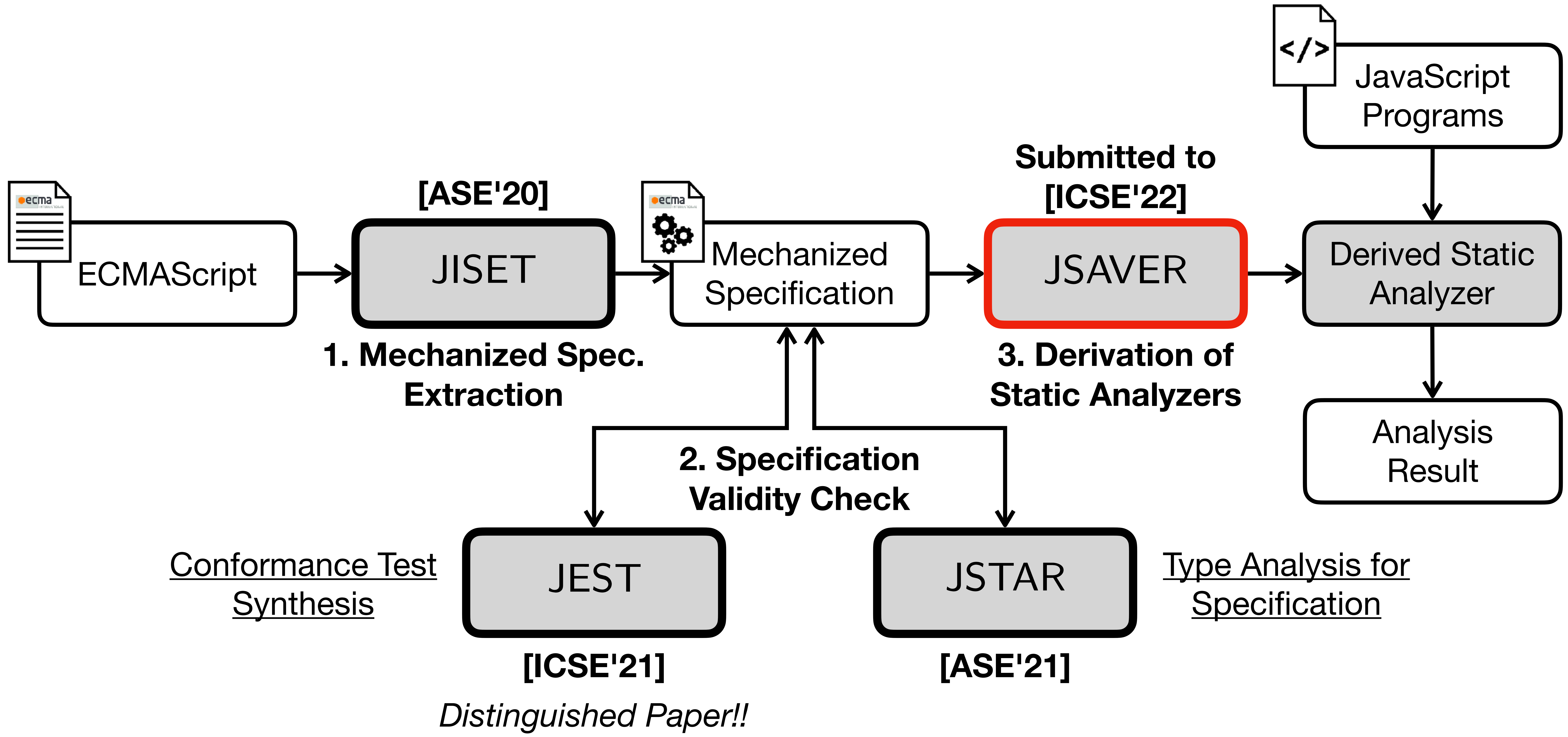
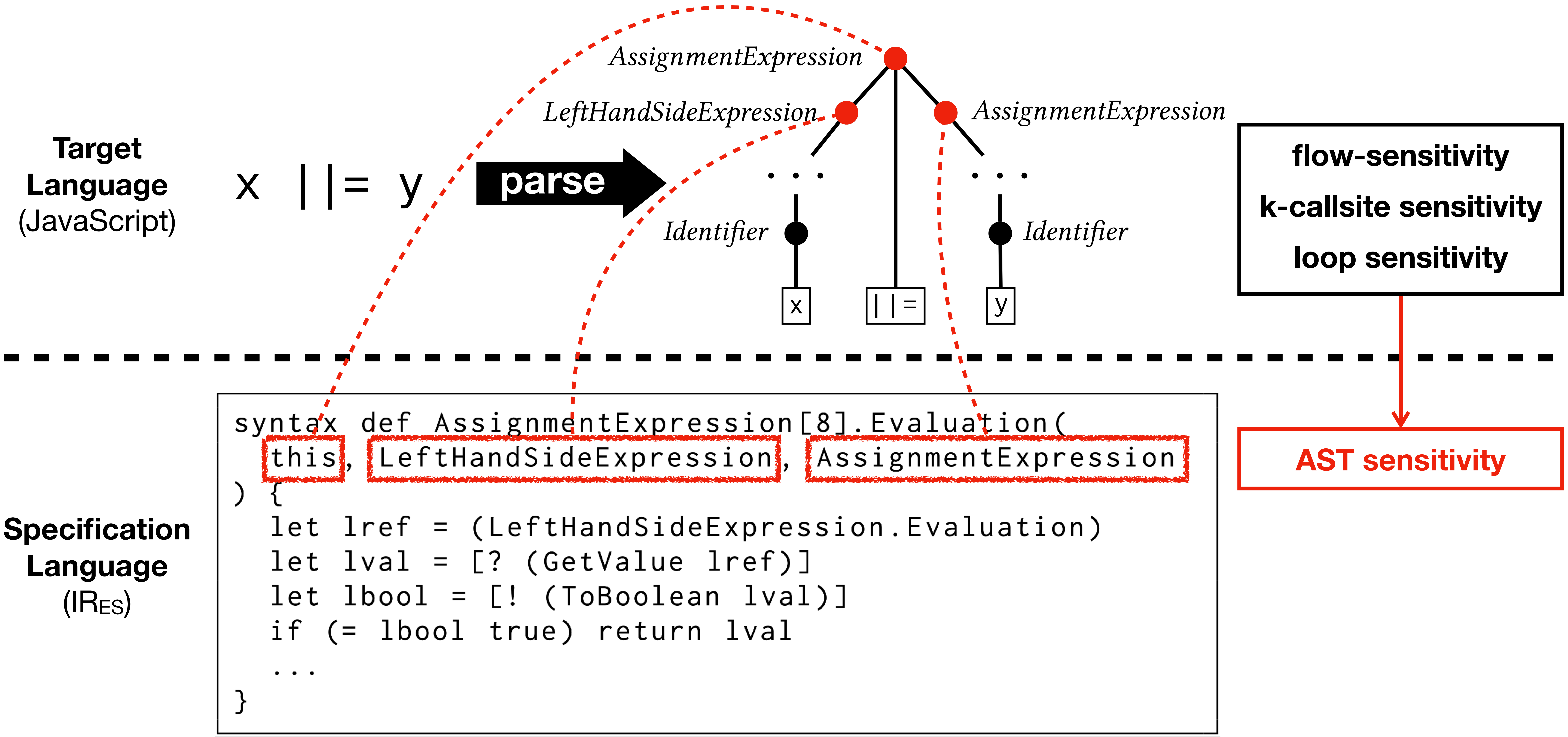


TABLE III: Type-related specification bugs newly detected by JSTAR in the official draft of ECMAScript 2021 (ES12)

Name	Feature	#	Description	Checker
ES12-1	Switch	3	Variables `hasDuplicates` and `hasUndefinedLabels` are already defined in algorithms for **`case`** blocks of **`switch`** statements.	Reference
ES12-2	Try	3	Variables `hasDuplicates` and `hasUndefinedLabels` are already defined in algorithms for **`try`** statements.	Reference
ES12-3	Arguments	1	A variable `index` is already defined in **CreateMappedArgumentsObject**.	Reference
ES12-4	Array	2	A variable `succeeded` is already defined in algorithms for `Array` objects.	Reference
ES12-5	Async	1	A variable `value` is already defined in **Evaluation** for **`yield`** expressions.	Reference
ES12-6	Class	1	A variable `ClassHeritage` is not defined in **Contains** for tails of **`class`** declarations.	Reference
ES12-7	Branch	1	A variable `Statement` is not defined in **EarlyErrors** for **`if`** statement.	Reference
ES12-8	Arguments	2	Abrupt completions are used in **DefineOwnProperty** and **GetOwnProperty** for `arguments` objects without any checks.	Operand


ECMAScript
[ASE'20]
JISET
1. Mechanized Spec. Extraction
Mechanized Specification
Submitted to [ICSE'22]
JSAVER
3. Derivation of Static Analyzers
JavaScript Programs
Derived Static Analyzer
Analysis Result
2. Specification Validity Check
Conformance Test Synthesis
JEST
[ICSE'21]
Distinguished Paper!!
JSTAR
Type Analysis for Specification
[ASE'21]

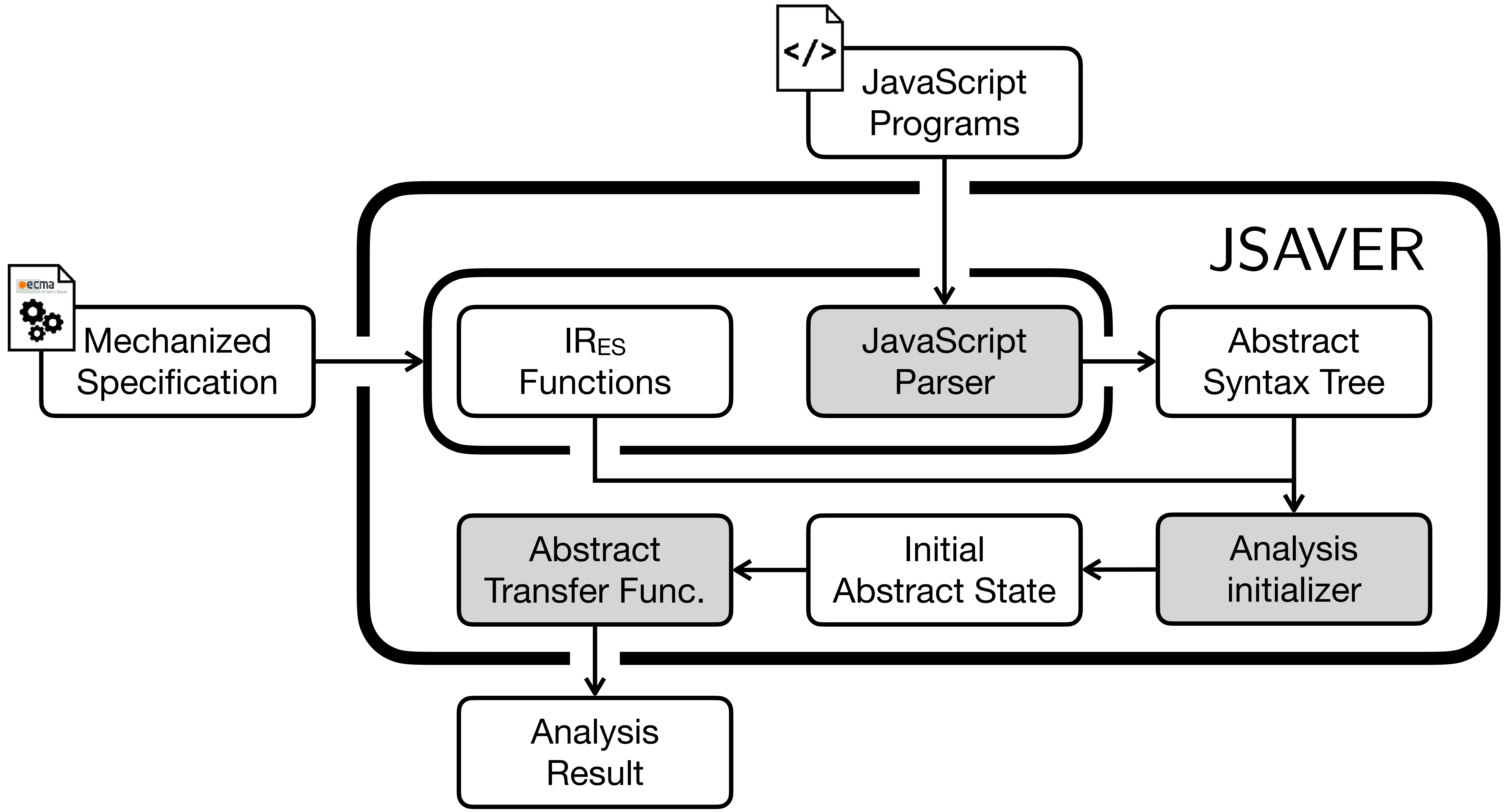
JSAVER - Meta-Level Static Analysis



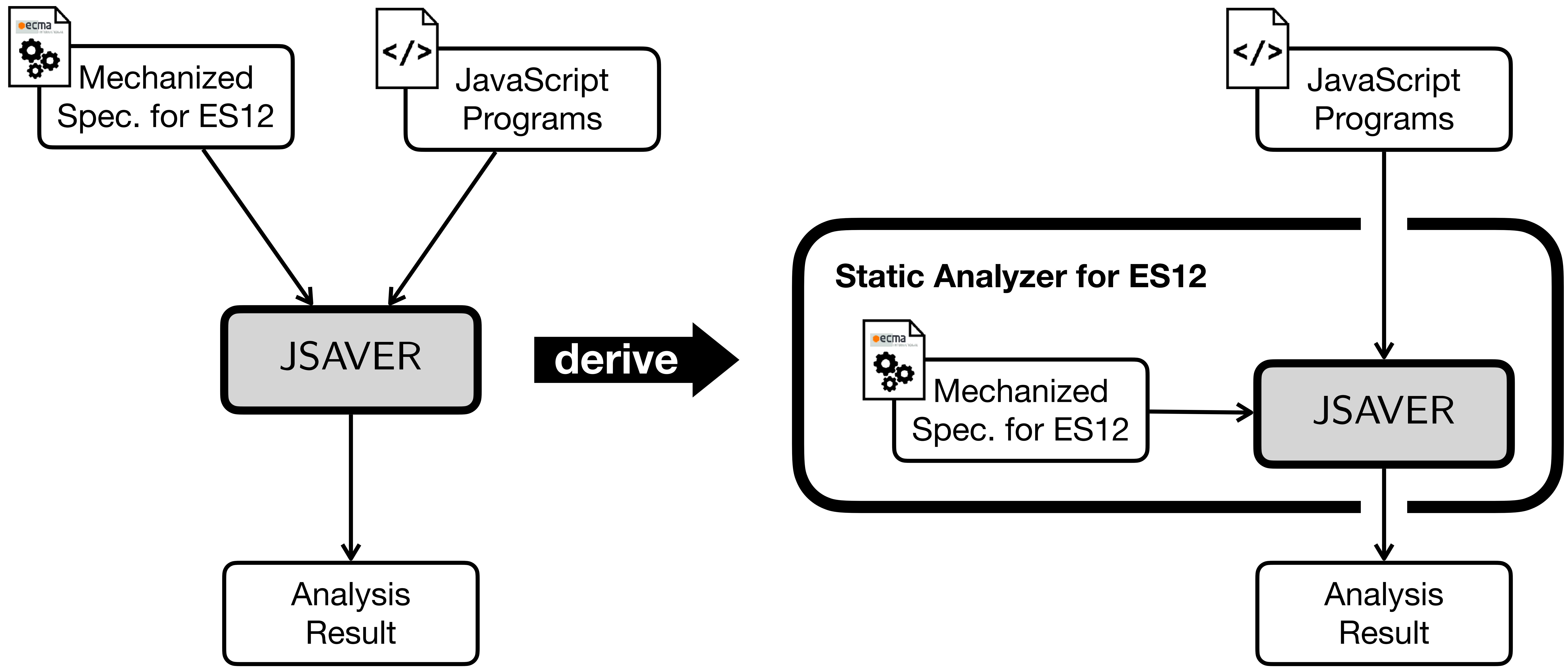
```
syntax def AssignmentExpression[8].Evaluation(
  this, LeftHandSideExpression, AssignmentExpression
) {
  let lref = (LeftHandSideExpression.Evaluation)
  let lval = [? (GetValue lref)]
  let lbool = [! (ToBoolean lval)]
  if (= lbool true) return lval
  ...
}
```

JSAVER Submitted to [ICSE'22]

JavaScript Static Analyzer via ECMAScript Representation

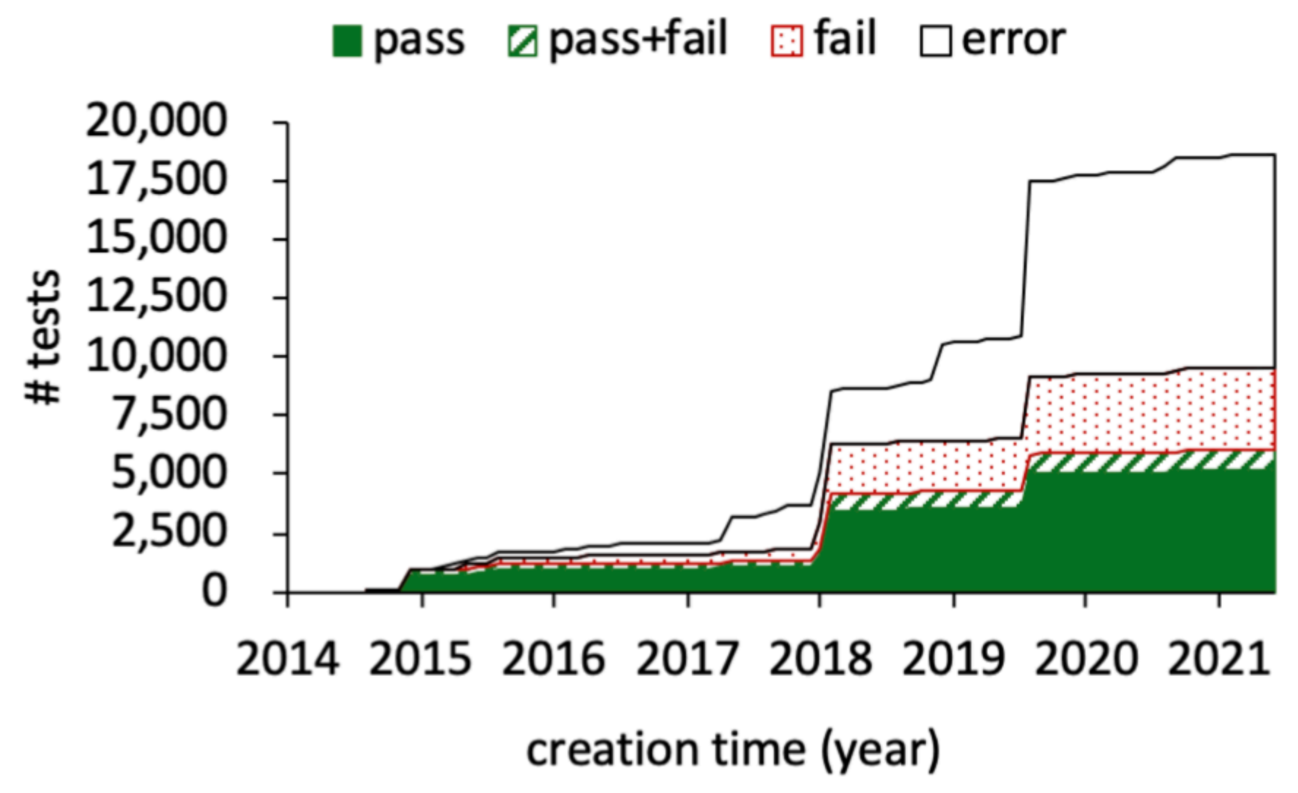


JSAVER - Static Analyzer Derivation

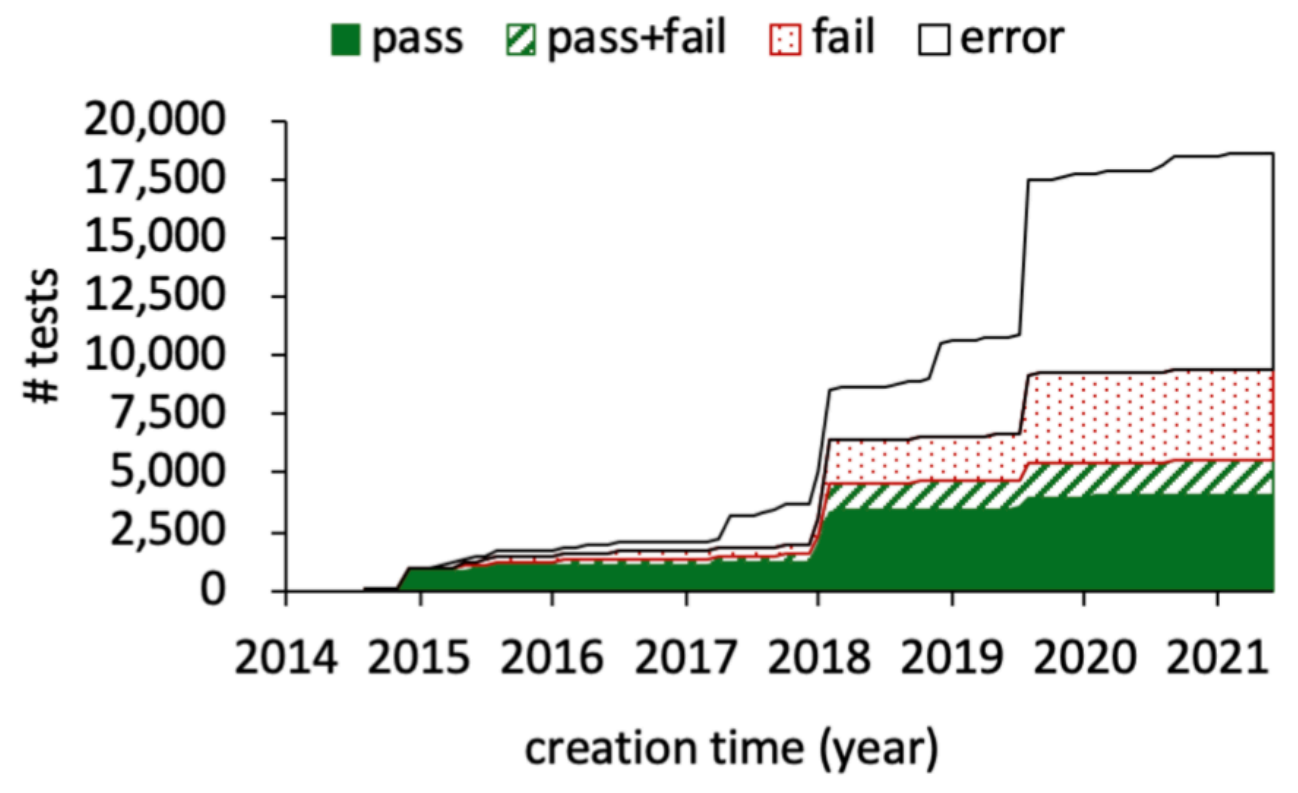


JSAVER - Evaluation

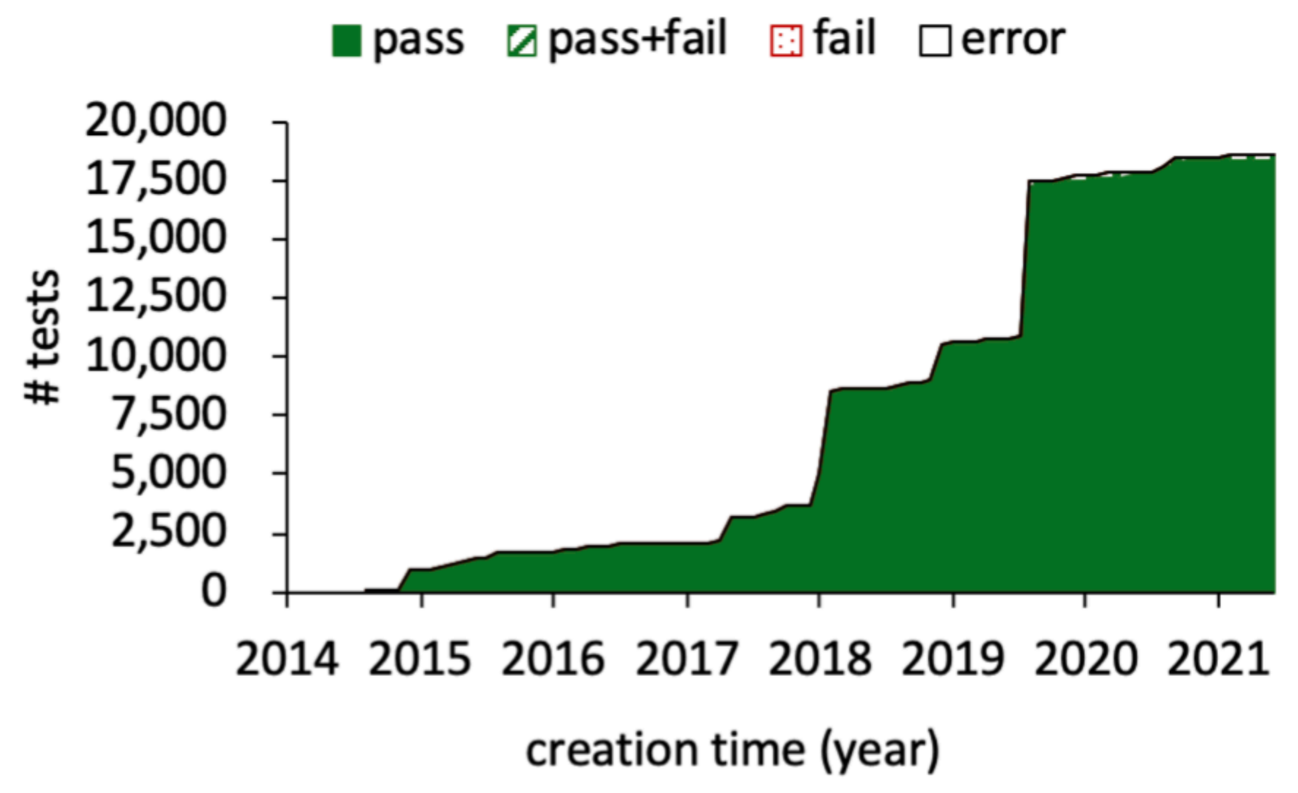
- **Coverage - Test262** (Official Conformance Tests)



(a) Analysis results of SAFE

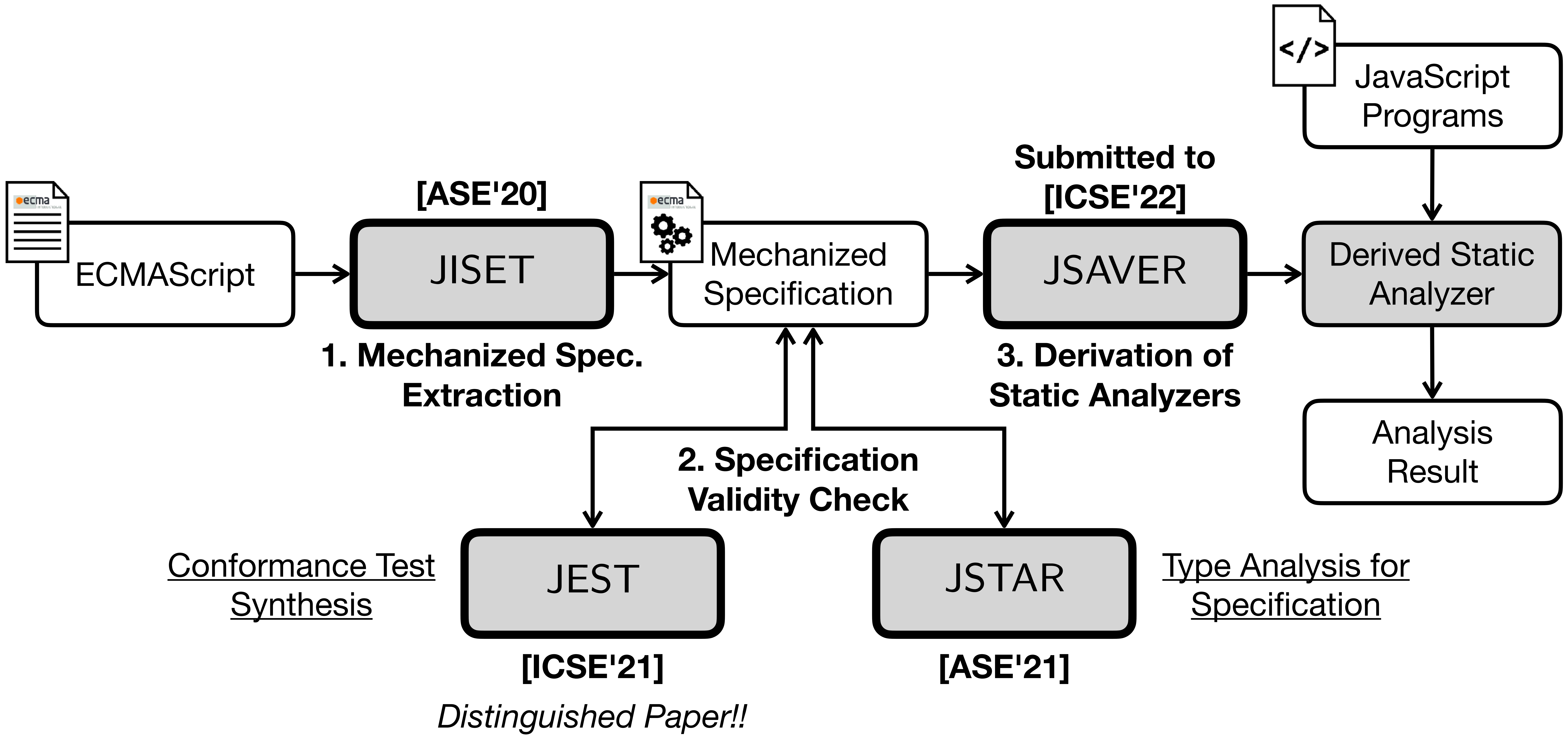


(b) Analysis results of TAJS



(c) Analysis results of JSA_{ES12}

- **Expressiveness - Abstract Domains / Analysis Sensitivity**
  - String Abstract Domain: String Set / Character Inclusion / Prefix-Suffix
  - Analysis Sensitivity: k-callsite sensitivity / loop sensitivity
- **Adaptability - Future Language Features**
  - Pipeline operator (`|>`) / `Observable` built-in library



JavaScript Programs
Submitted to [ICSE'22]
[ASE'20]
ECMAScript
JISET
Mechanized Specification
JSAVER
Derived Static Analyzer
1. Mechanized Spec. Extraction
3. Derivation of Static Analyzers
Analysis Result
2. Specification Validity Check
Conformance Test Synthesis
JEST
JSTAR
Type Analysis for Specification
[ICSE'21]
[ASE'21]
Distinguished Paper!!