기계화 명세를 이용한 JavaScript 언어의 설계 및 구현

세미나 @ 소프트웨어 분석 연구실

박지혁

고려대학교 컴퓨터학과

2023. 03. 10

JavaScript is Everywhere

Client-side Programming

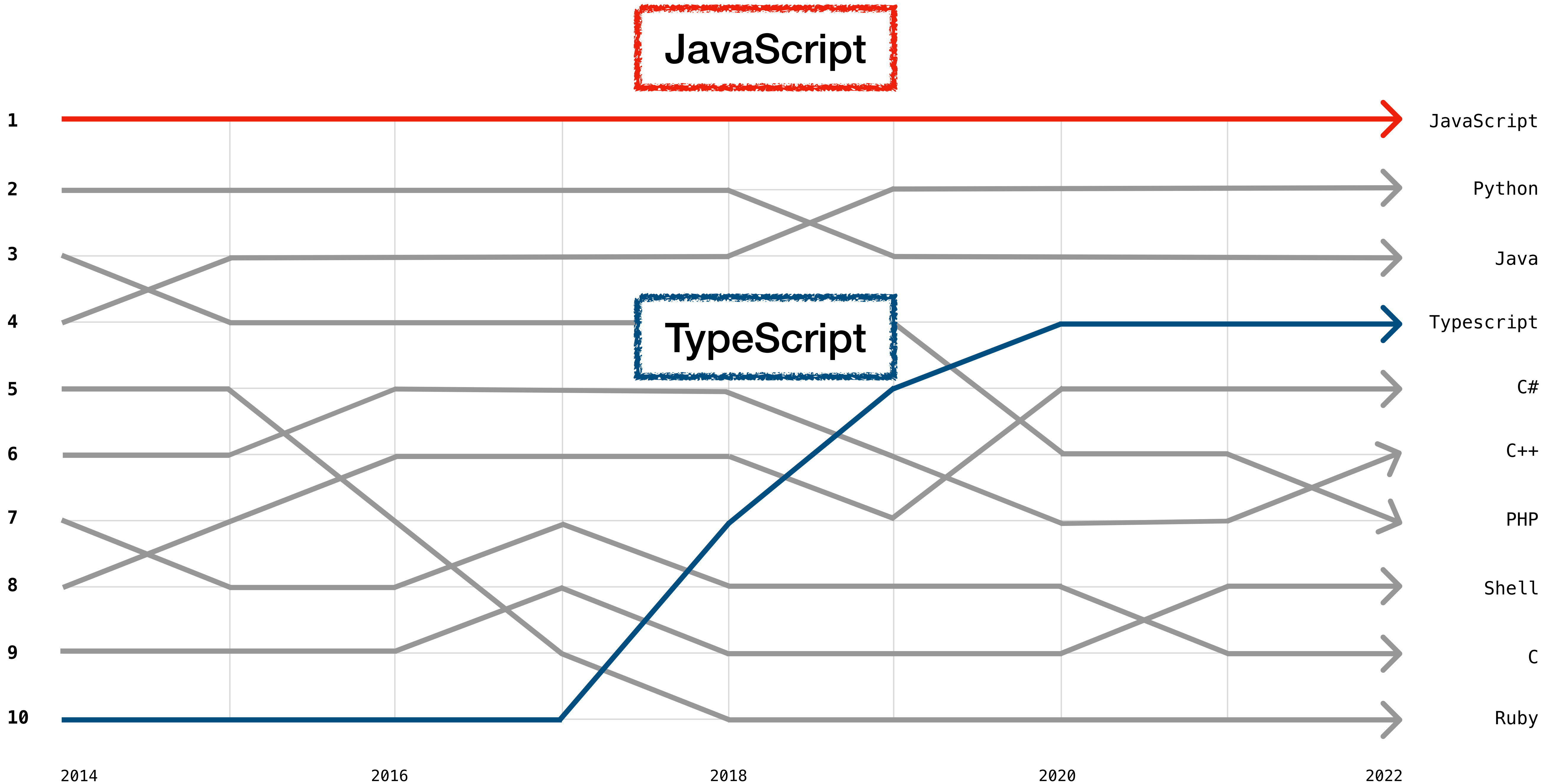
Mobile/Desktop Applications

JS

Server-side Programming

Others (PDF, IoT, Microcontrollers, etc.)

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; }
```

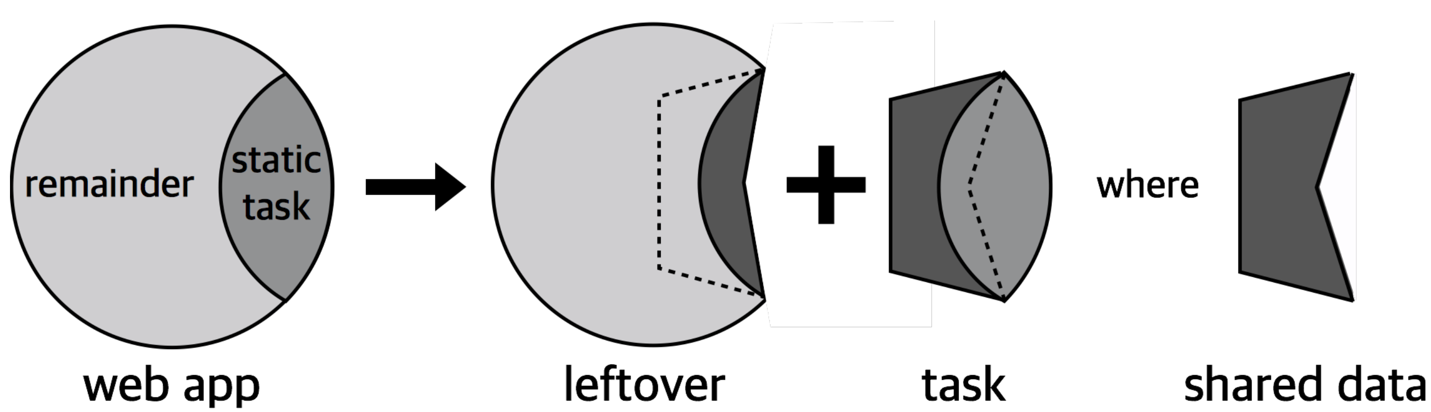
Always return `false`?

NO!!

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

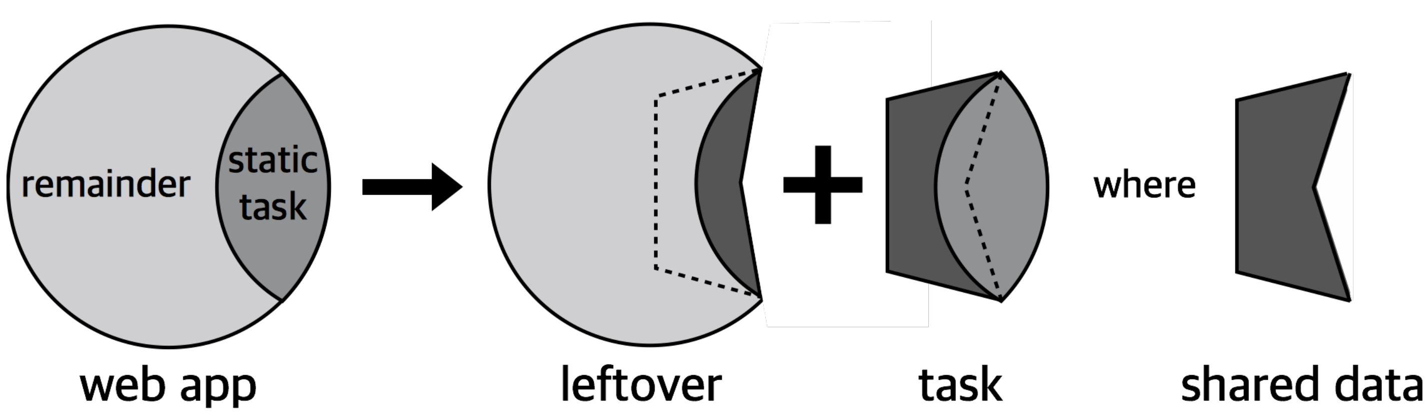
Early Research on JS Static Analysis

- **API Misuse Detection using TypeScript Declarations** (Mod'14)
  - Modeling abstract semantics using TypeScript `d.ts` files
- **SAFE (Scalable Analysis Framework for ECMAScript) 2.0** (ICSE'17 - Demo)
  - Extensibility (Abs. Domain, Sensitivity) / GUI Web Debugger
- **Revisiting Recency Abstraction** (SOAP'17)
  - Explaining why recency abstraction is not monotone
- **Dynamic Inter-Device Task Dispatch** (ProWeb'18)

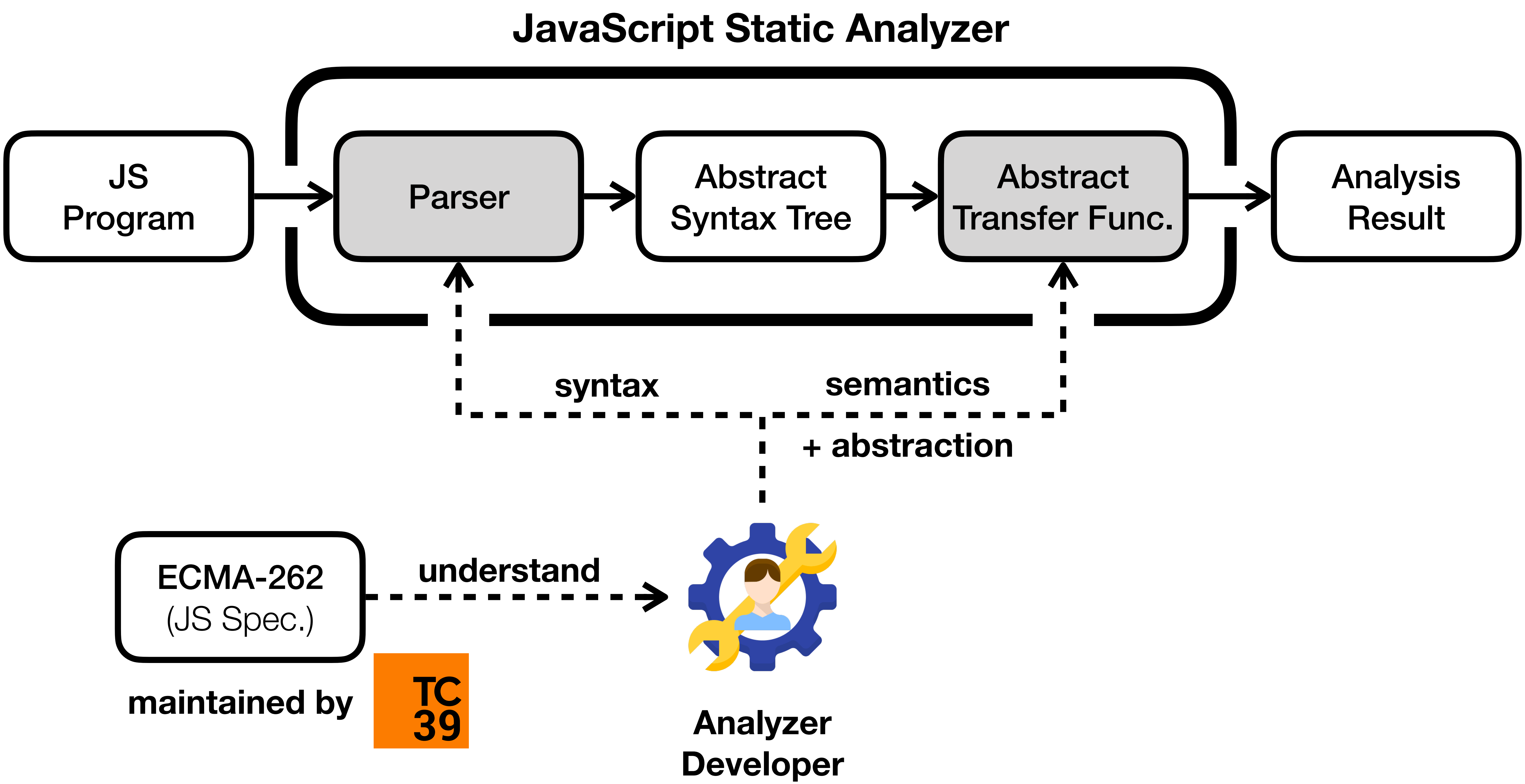


Early Research on JS Static Analysis

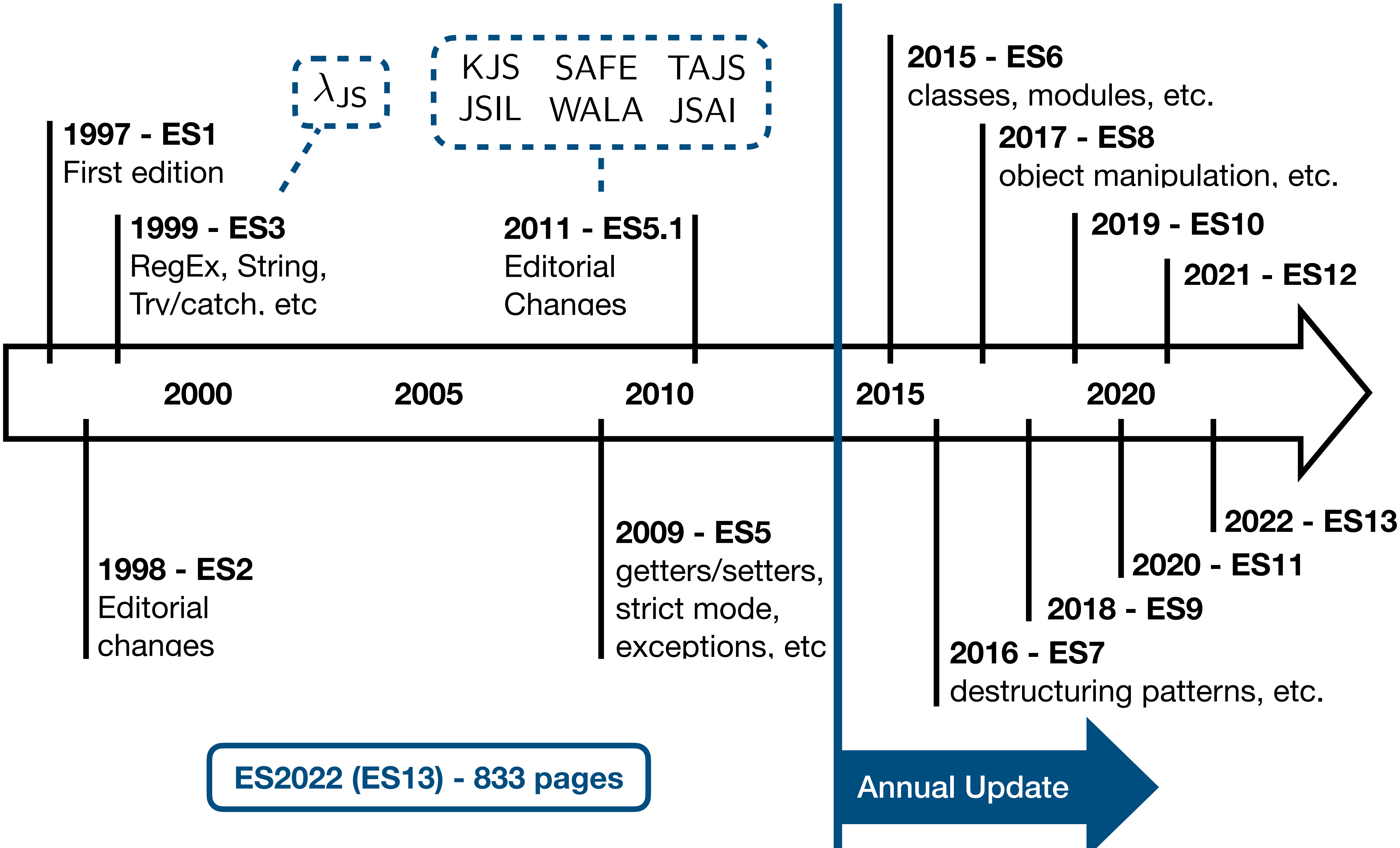
- **API Misuse Detection using TypeScript Declarations** (Mod'14)
  - Modeling abstract semantics using TypeScript `d.ts` files
- **SAFE (Scalable Analysis Framework for ECMAScript) 2.0** (ICSE'17 - Demo)
  - Extensibility (Abs. Domain, Sensitivity) / GUI Web Debugger
- **Revisiting Recency Abstraction** (SOAP'17)
  - Explaining why recency abstraction is not monotone
- **Dynamic Inter-Device Task Dispatch** (ProWeb'18)



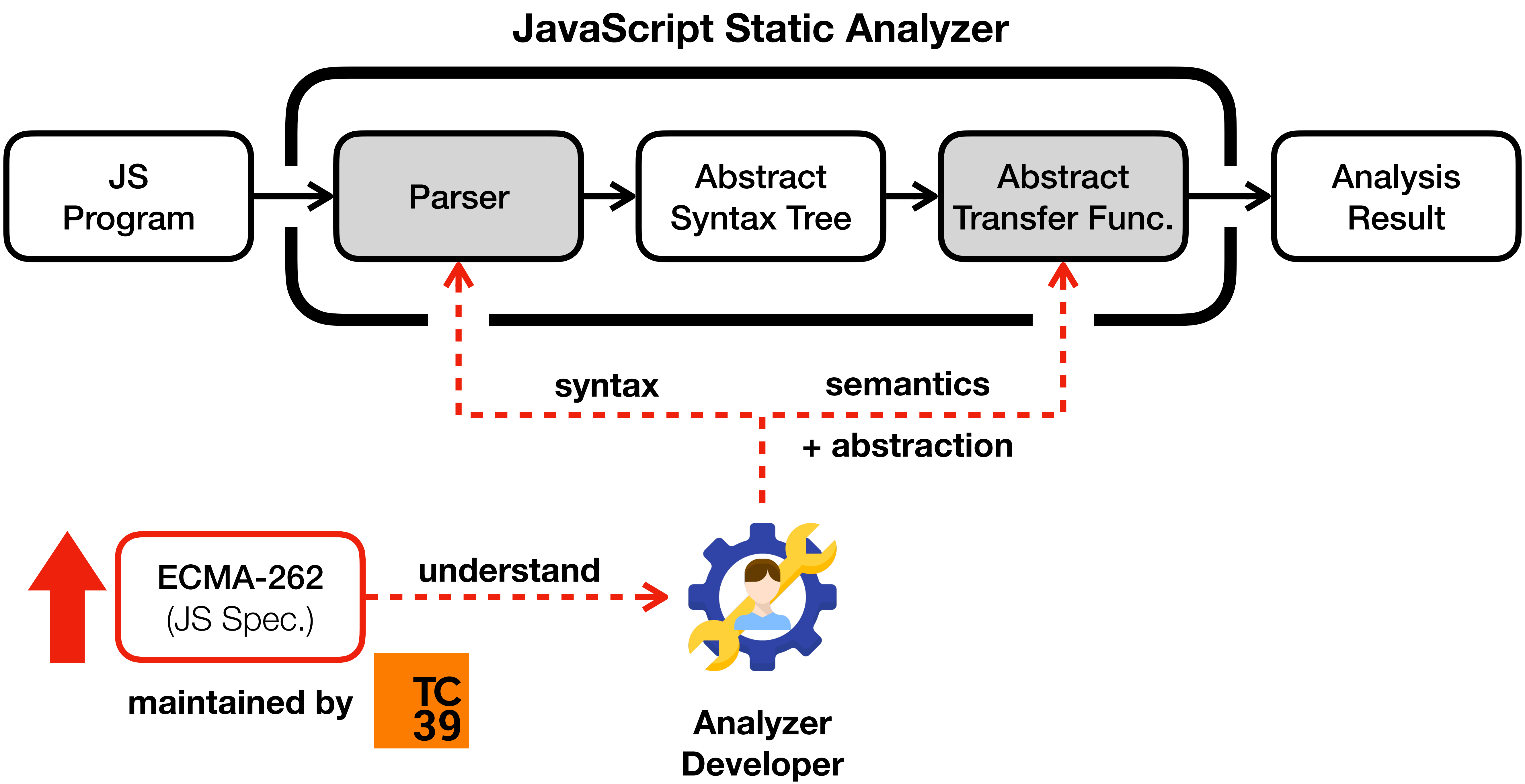
JavaScript Static Analyzers



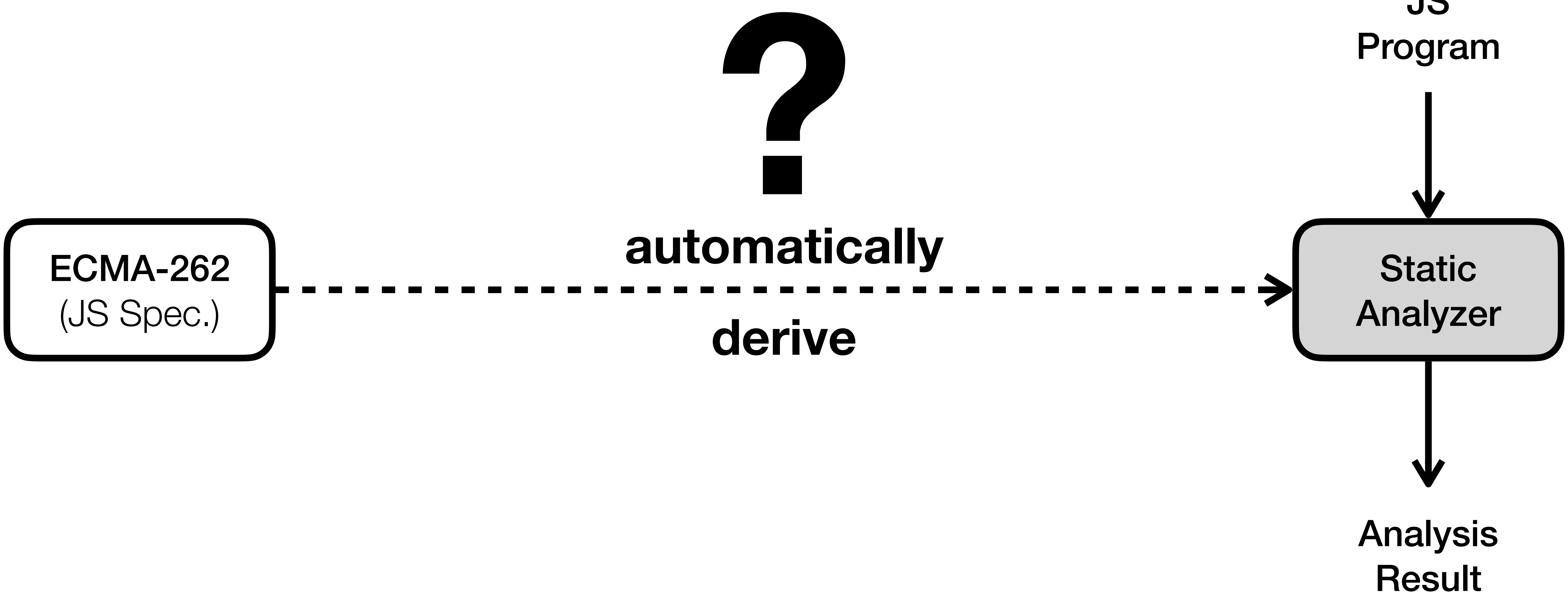
Problem: Fast Evolving JavaScript



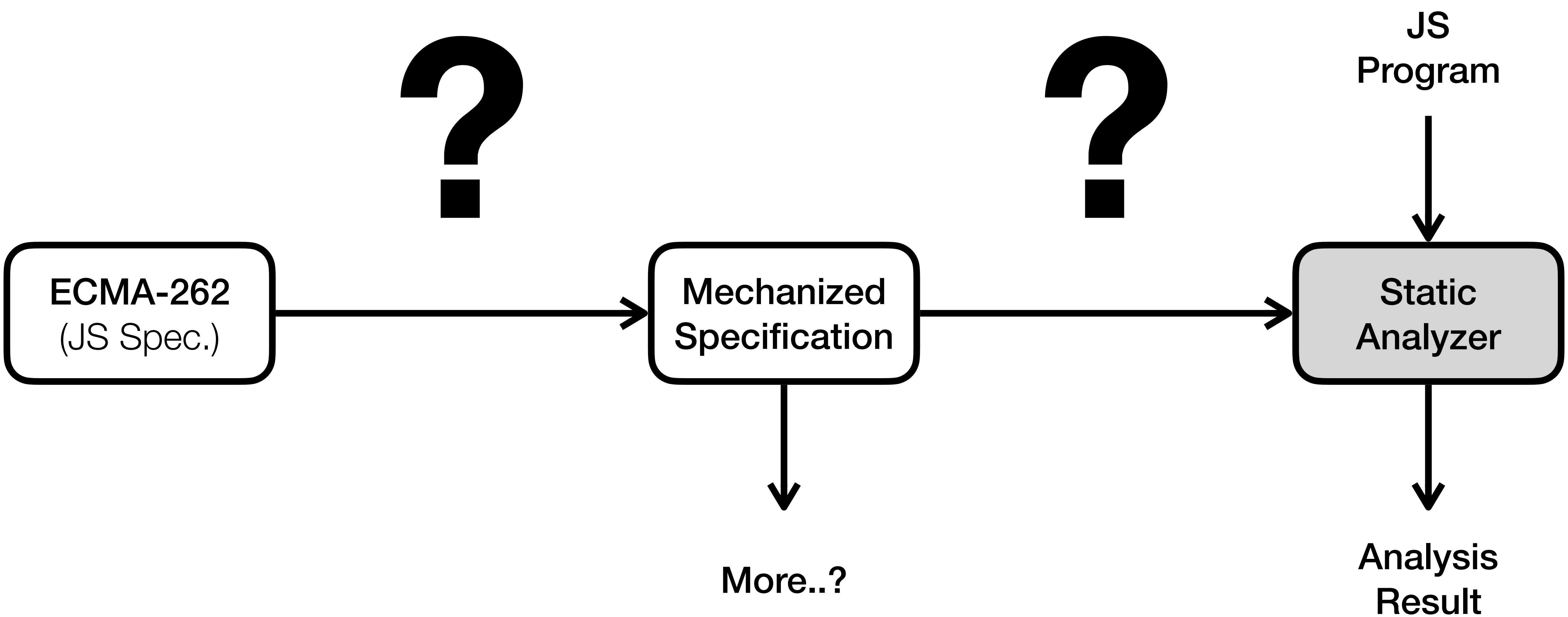
Problem: Manual Update

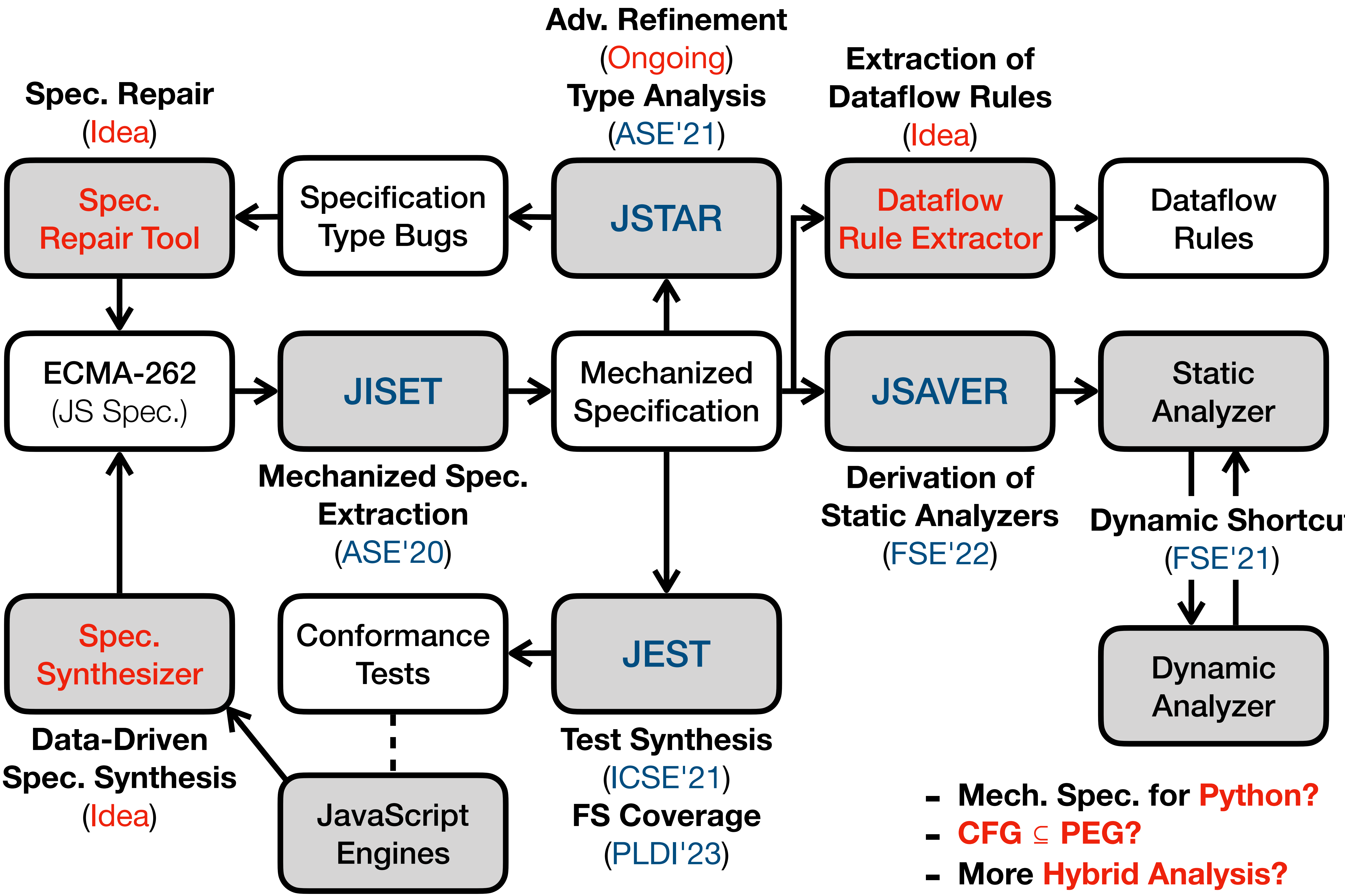


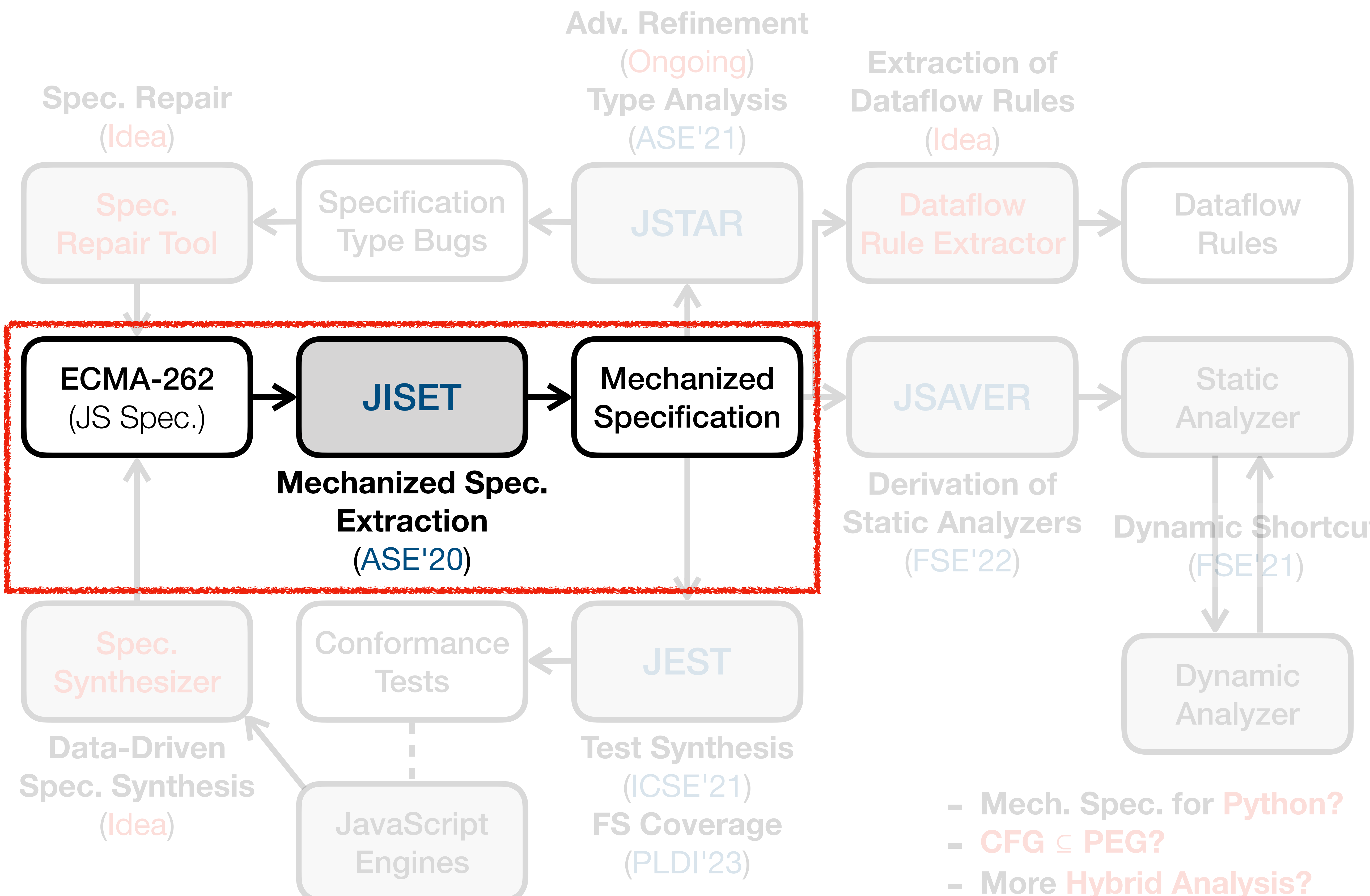
Derivation of Static Analyzer?



Idea: Mechanized Specification




Adv. Refinement
(Ongoing)
Type Analysis
(ASE'21)
Extraction of
Dataflow Rules
(Idea)
Spec. Repair
(Idea)
Spec.
Repair Tool
Specification
Type Bugs
JSTAR
Dataflow
Rule Extractor
Dataflow
Rules
ECMA-262
(JS Spec.)
JISET
Mechanized
Specification
JSAVER
Static
Analyzer
Mechanized Spec.
Extraction
(ASE'20)
Derivation of
Static Analyzers
(FSE'22)
Dynamic Shortcut
(FSE'21)
Spec.
Synthesizer
Conformance
Tests
JEST
Dynamic
Analyzer
Data-Driven
Spec. Synthesis
(Idea)
JavaScript
Engines
Test Synthesis
(ICSE'21)
FS Coverage
(PLDI'23)
- Mech. Spec. for Python?
- CFG ⊆ PEG?
- More Hybrid Analysis?



[ASE'20] J. Park, et al. "JISET: JavaScript IR-based Semantics Extraction Toolchain"

ECMA-262 (JavaScript Spec.)

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

The production of *ArrayLiteral* in ES13

Syntax

TC 39

13.2.5.2 Runtime Semantics: Evaluation

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

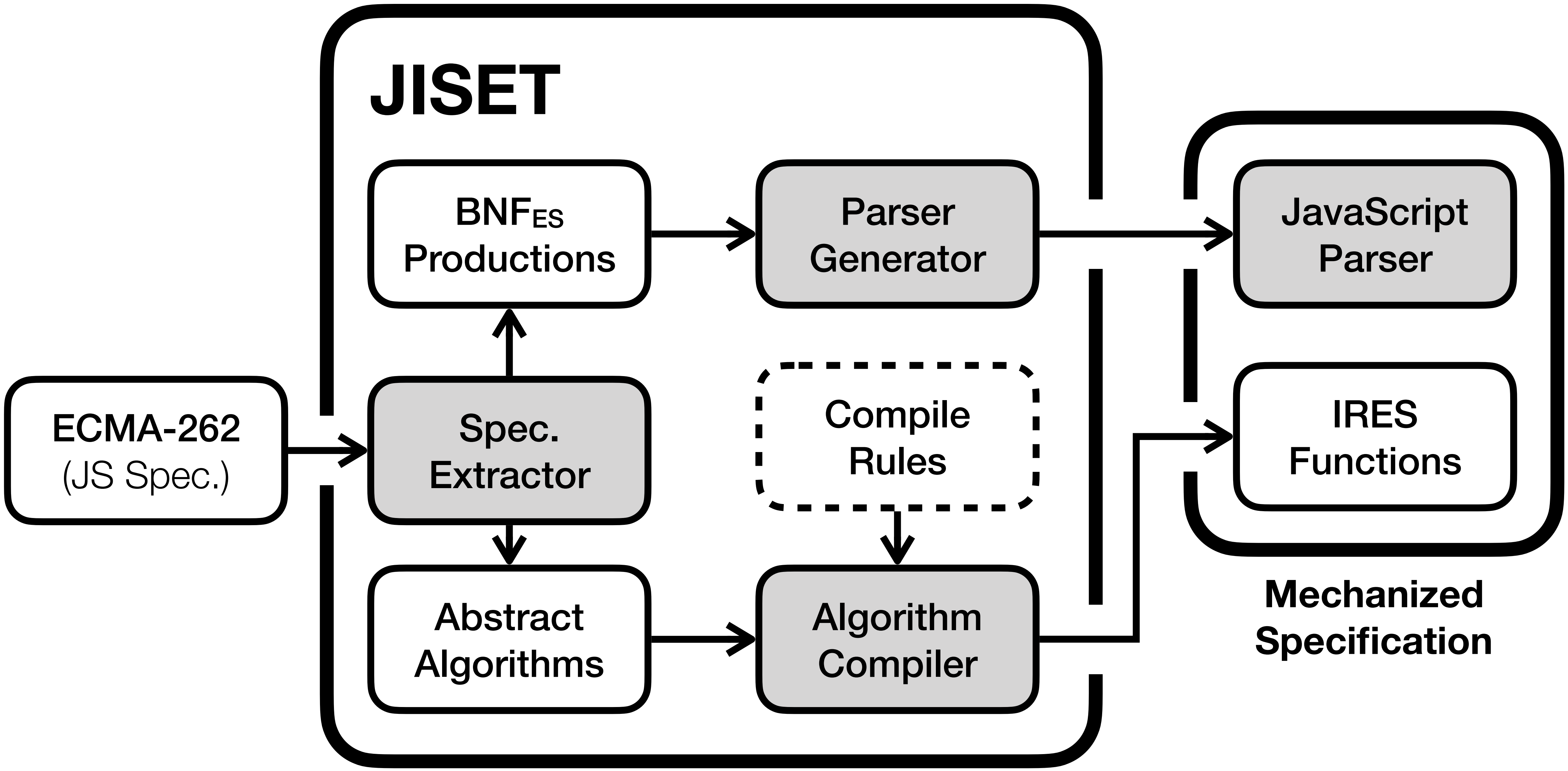
1. Let *array* be ! ArrayCreate(0).
2. Let *nextIndex* be the result of performing ArrayAccumulation for *ElementList* with arguments *array* and 0.
3. ReturnIfAbrupt(*nextIndex*).
4. If *Elision* is present, then
   a. Let *len* be the result of performing ArrayAccumulation for *Elision* with arguments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

Semantics

The Evaluation **algorithm for the third alternative of** *ArrayLiteral* **in ES13**

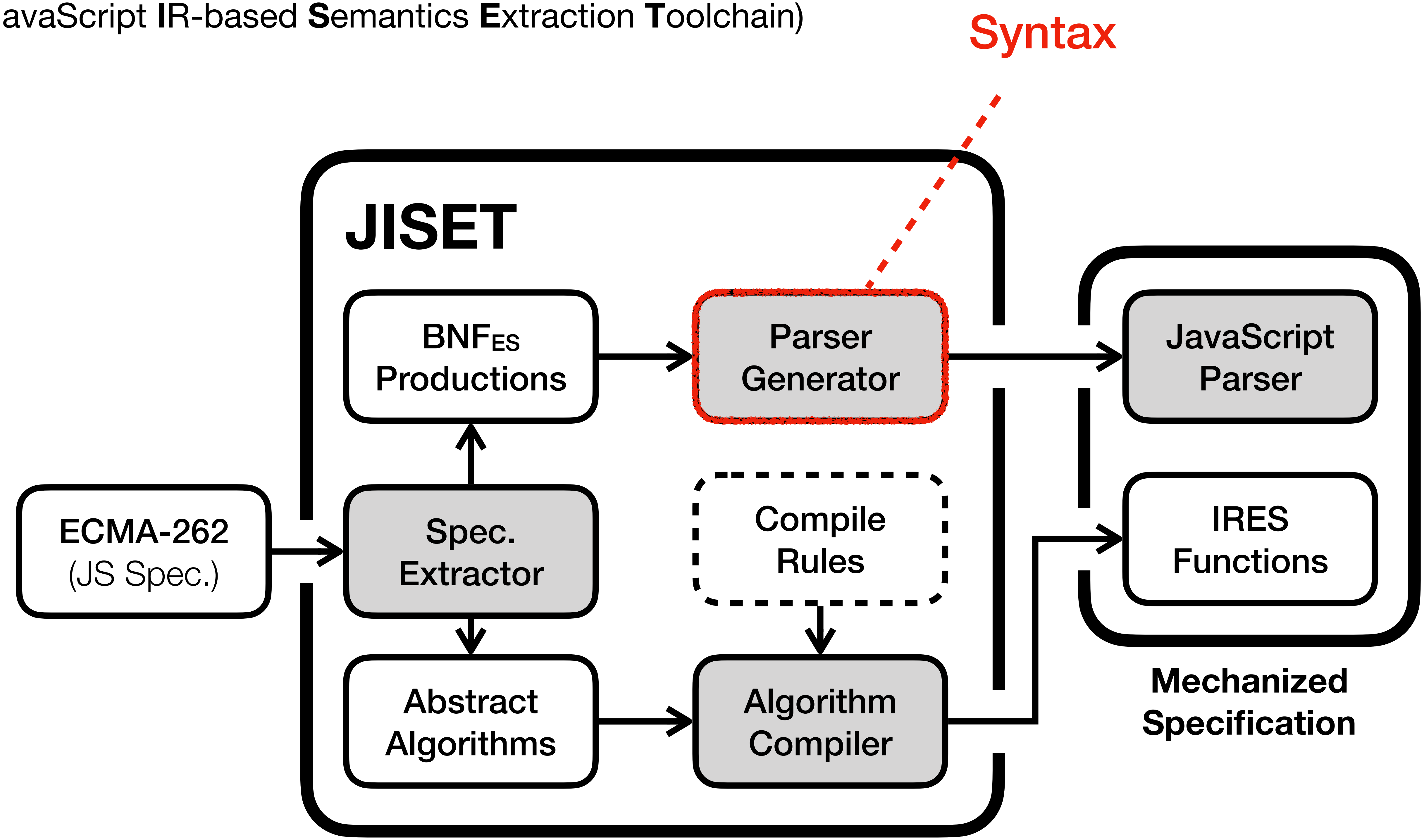
JISET - ASE'20

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

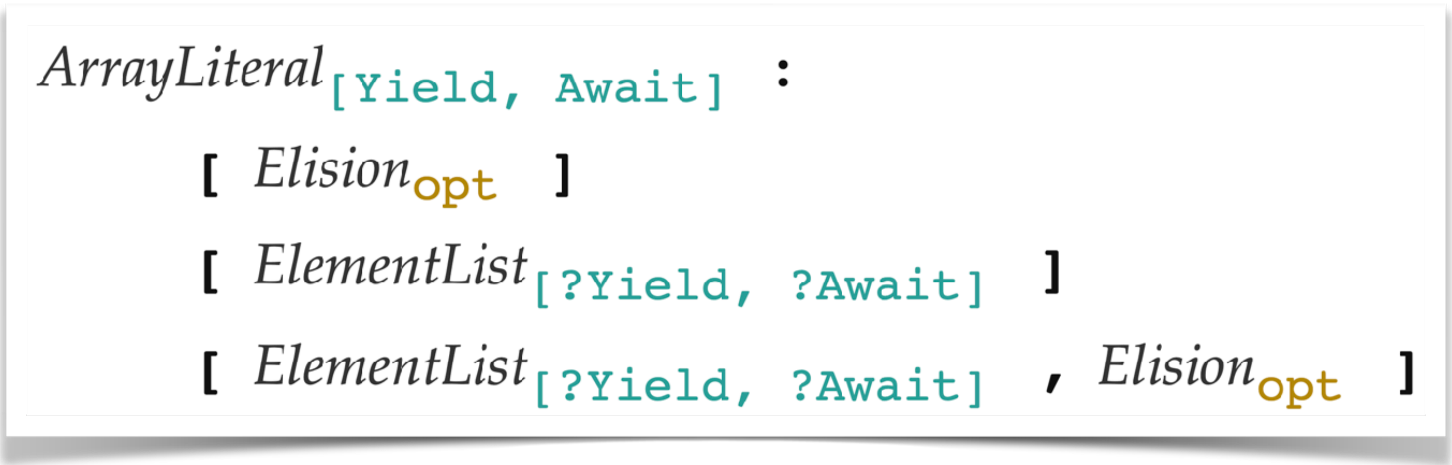


JISET - ASE'20

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



JISET - Parser Generator (Syntax)



Parsing Expression Grammar (PEG)

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

[POPL'04] B. Ford, "Parsing Expression Grammars: A Recognition-based Syntactic Foundation"

JISET - Parser Generator (Syntax)

- **Context-Free Grammar (CFG)**
  - Unordered Choices

$$A ::= B\,\texttt{;} \mid B\;\texttt{+}\;B\,\texttt{;} \qquad \texttt{xy;}$$
$$B ::= \texttt{x} \mid \texttt{xy} \qquad \texttt{x+x;}$$

- **Parsing Expression Grammar (PEG)**
  - Ordered Choices

$$A ::= B\,\texttt{;} \;/\; B\;\texttt{+}\;B\,\texttt{;} \qquad \texttt{xy;}$$
$$B ::= \texttt{x} \;/\; \boxed{\texttt{xy}} \qquad \texttt{x+x;}$$

always ignored

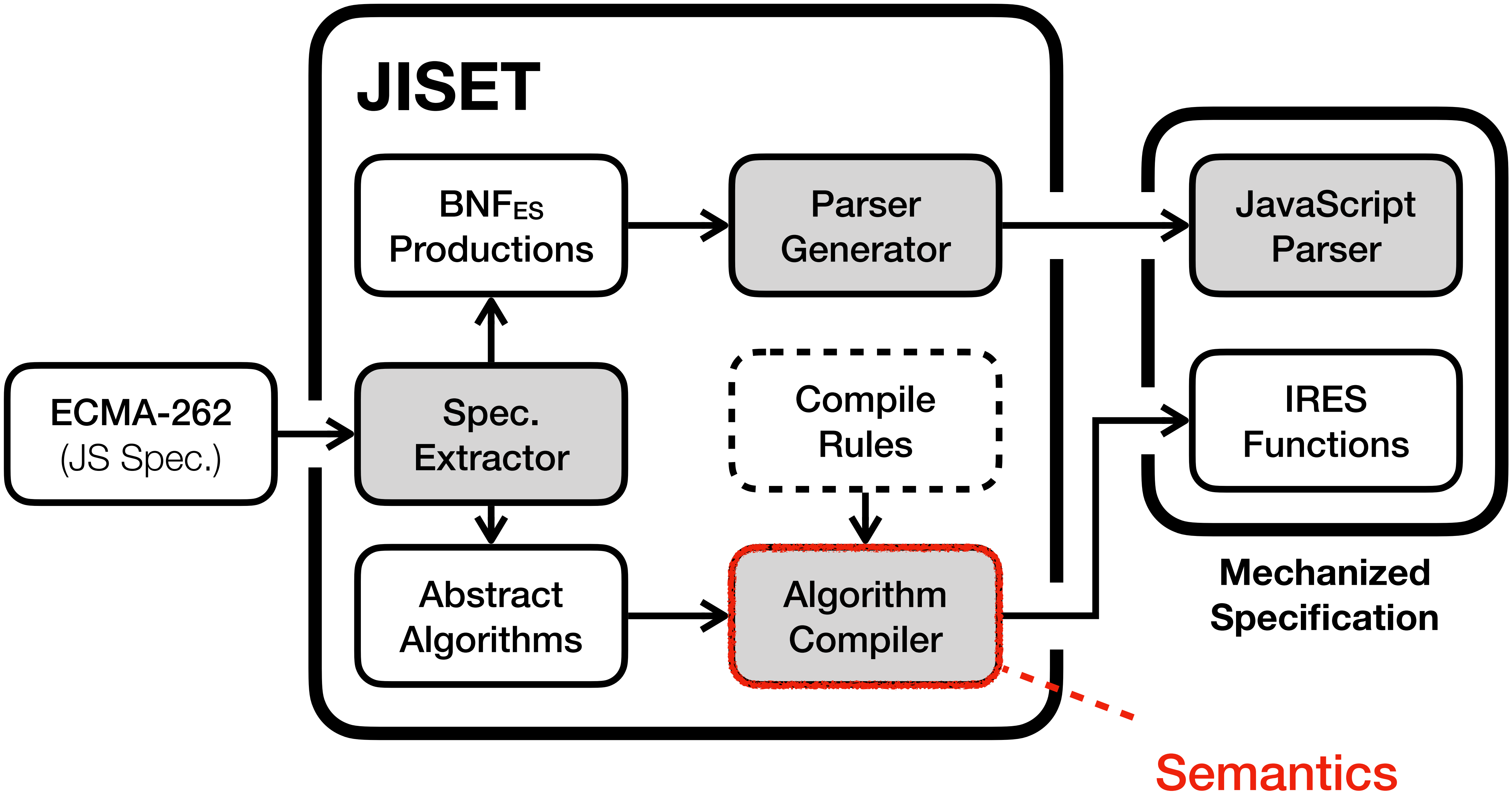
- **PEG with Lookahead Parsing**
  - Ordered Choices with Lookahead Tokens

$$A ::= B\,\texttt{;} \;/\; B\;\texttt{+}\;B\,\texttt{;} \qquad \texttt{xy;}$$
$$B ::= \texttt{x} \;/\; \texttt{xy} \qquad \texttt{x+x;}$$

[POPL'04] B. Ford, "Parsing Expression Grammars: A Recognition-based Syntactic Foundation"

JISET

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



JISET - Metalanguages for ECMA-262

- IR_{ES} - Intermediate Representation for ECMA-262

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

• • •

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

• • •

JISET - Metalanguages for ECMA-262

- IR_{ES} - Intermediate Representation for ECMA-262

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

• • •

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

• • •

JISET - Metalanguages for 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*.

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

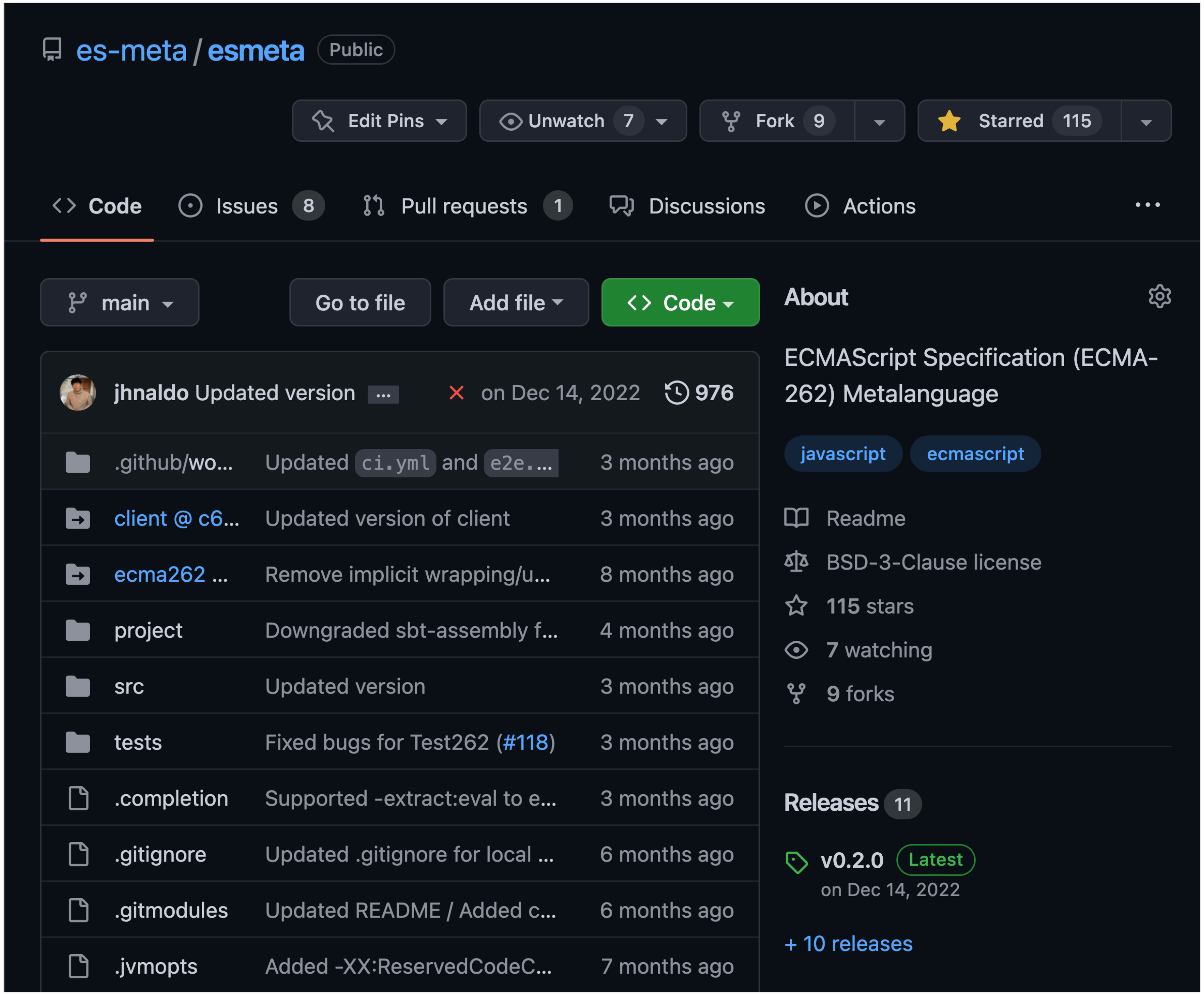
≈ 96% Compiled

■ auto ■ manual

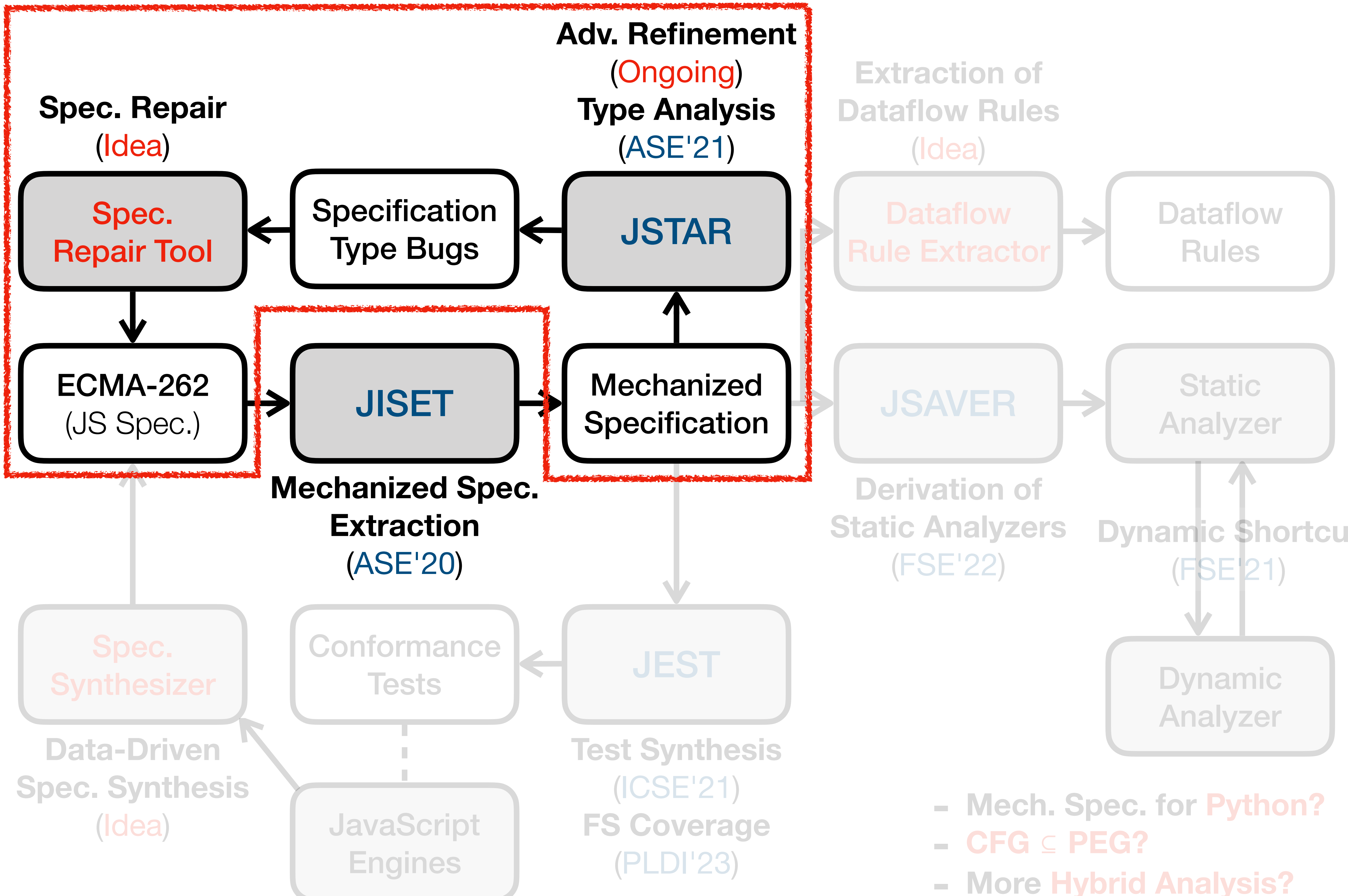
T: Total **L**: Core Language Semantics **B**: Built-in Libraries

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

ESMeta



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



[ASE'21] J. Park, et al. "JSTAR: JavaScript Specification Type Analyzer using Refinement"

Types in ECMA-262

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 ≥ -0.5, return **-0**.

• • •

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

Types in ECMA-262

20.3.2.28 Math.round ($\boxed{x}$) `x: (String ∨ Boolean ∨ Number ∨ Object ∨ ...)`

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

Types in ECMA-262

20.3.2.28 Math.round ($\boxed{x}$) `x: (String ∨ Boolean ∨ Number ∨ Object ∨ ...)`

1. Let n be ? $\boxed{\text{ToNumber}(x)}$ `ToNumber(x): (Number ∨ 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 ≥ -0.5, return **-0**.

• • •

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

Types in ECMA-262

20.3.2.28 Math.round ($\boxed{x}$) `x: (String v Boolean v Number v Object v ...)`

1. Let n be $\boxed{?}$ $\boxed{\text{ToNumber}(x)}$ `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

Types in ECMA-262

20.3.2.28 Math.round ($\boxed{x}$) `x: (String ∨ Boolean ∨ Number ∨ Object ∨ ...)`

1. Let $\boxed{n}$ be $\boxed{?}$ $\boxed{\text{ToNumber}(x)}$ `ToNumber(x): (Number ∨ Exception) ∧ 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 ≥ -0.5, return **-0**.

• • •

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

Types in ECMA-262

20.3.2.28 Math.round ($\boxed{x}$) `x: (String ∨ Boolean ∨ Number ∨ Object ∨ ...)`

1. Let $\boxed{n}$ be $\boxed{?}$ $\boxed{\text{ToNumber}(x)}$ `ToNumber(x): (Number ∨ Exception) ∧ n: (Number)`
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**.

Type Mismatch for
numeric operator \`>\`

• • •

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

Types in ECMA-262

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

1. Let n be ? ToNumber(x). `ToNumber(x): (Number ∨ Exception) ∧ 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

Types in ECMA-262

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

1. Let n be ? ToNumber(x). `ToNumber(x): (Number ∨ Exception) ∧ 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) = ?
```

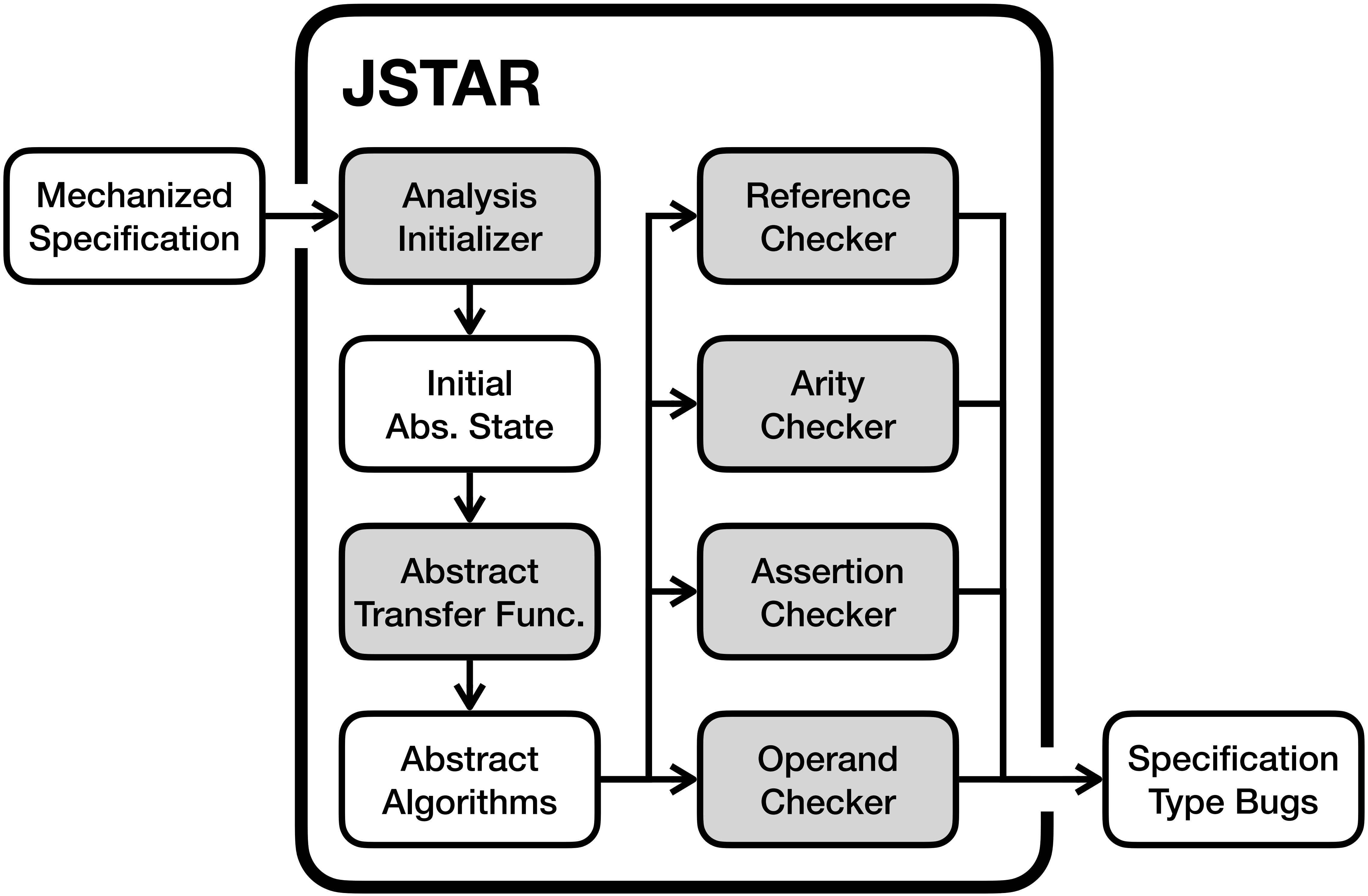
3. If $n < 0.5$ and $n > 0$, return **+0**.
4. If $n < 0$ and $n \geq$ -0.5, return **-0**.

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

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

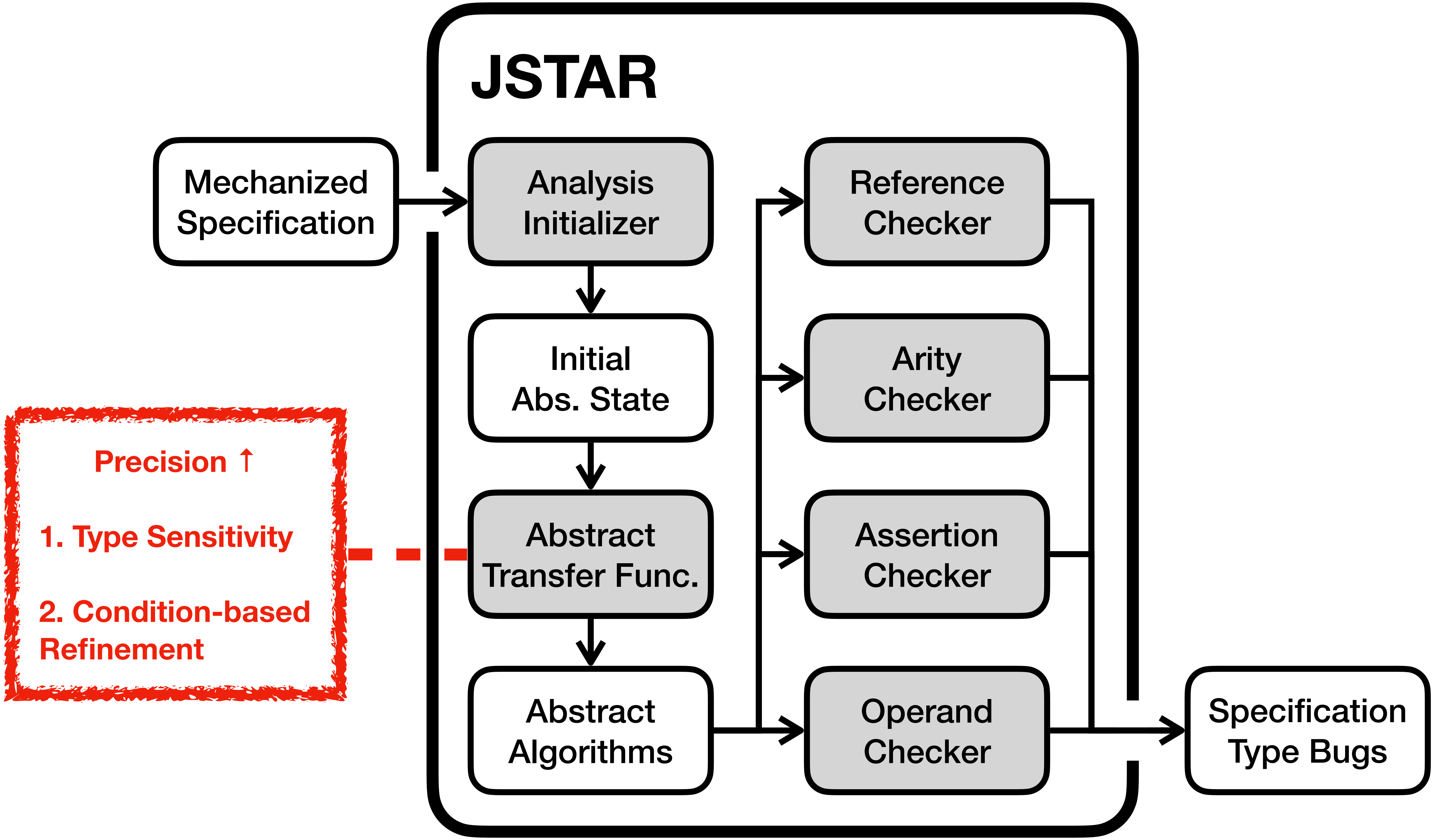
JSTAR - ASE'21

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



JSTAR - ASE'21

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



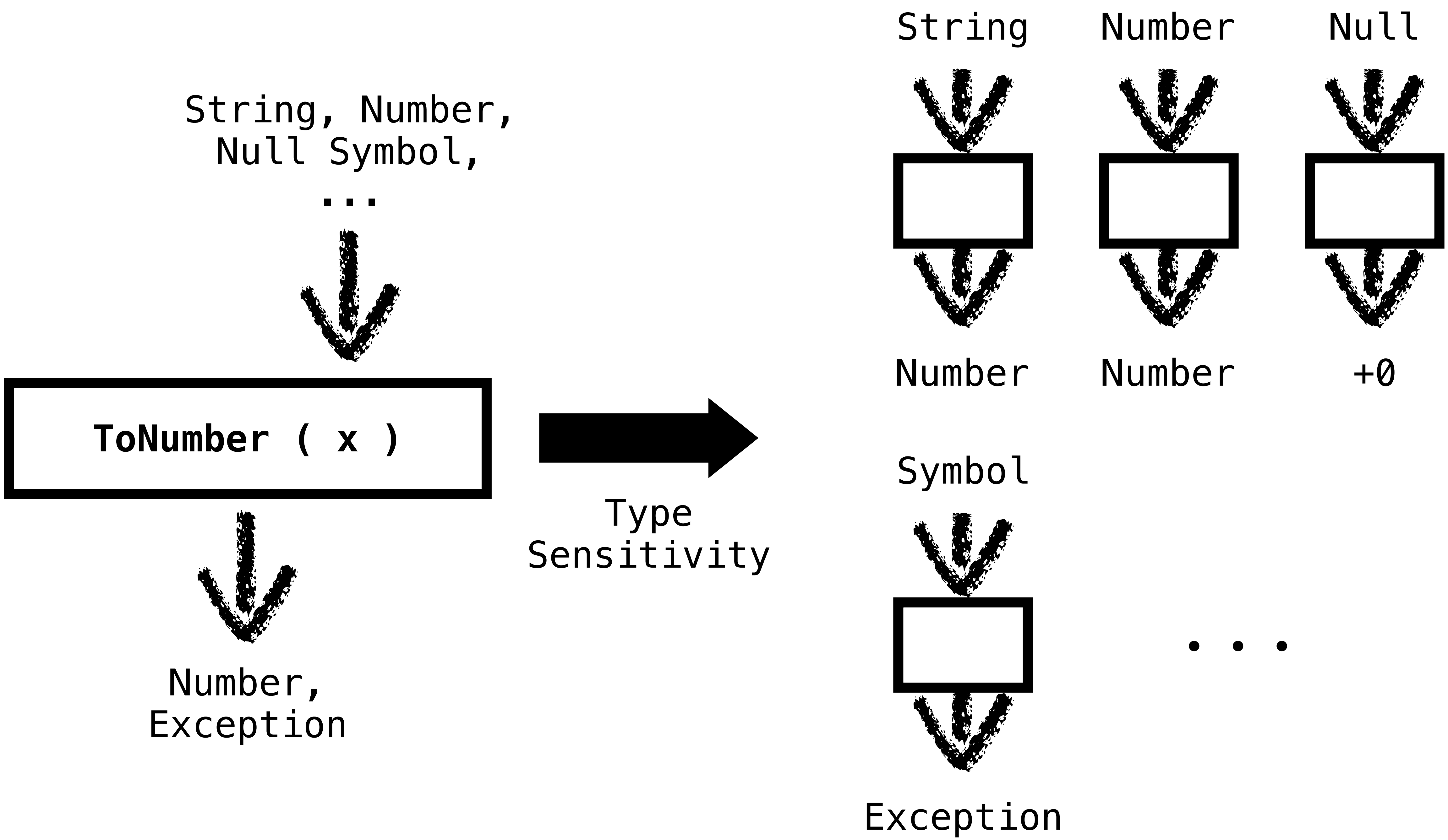
JSTAR - Type Sensitivity

`String, Number,`
`Null Symbol,`
`...`

`ToNumber ( x )`

`Number,`
`Exception`

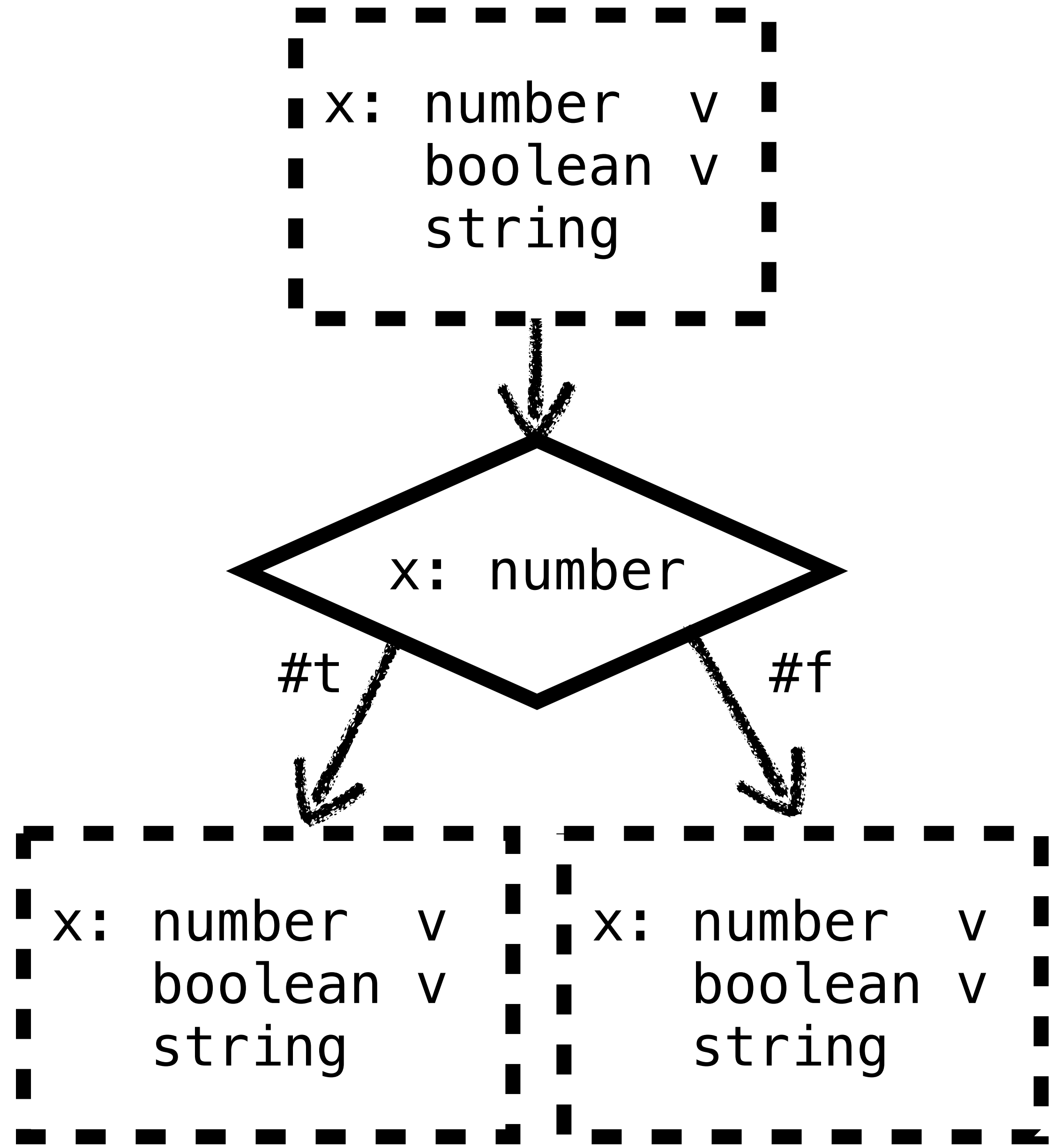
JSTAR - Type Sensitivity



JSTAR - Condition-based Refinement

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

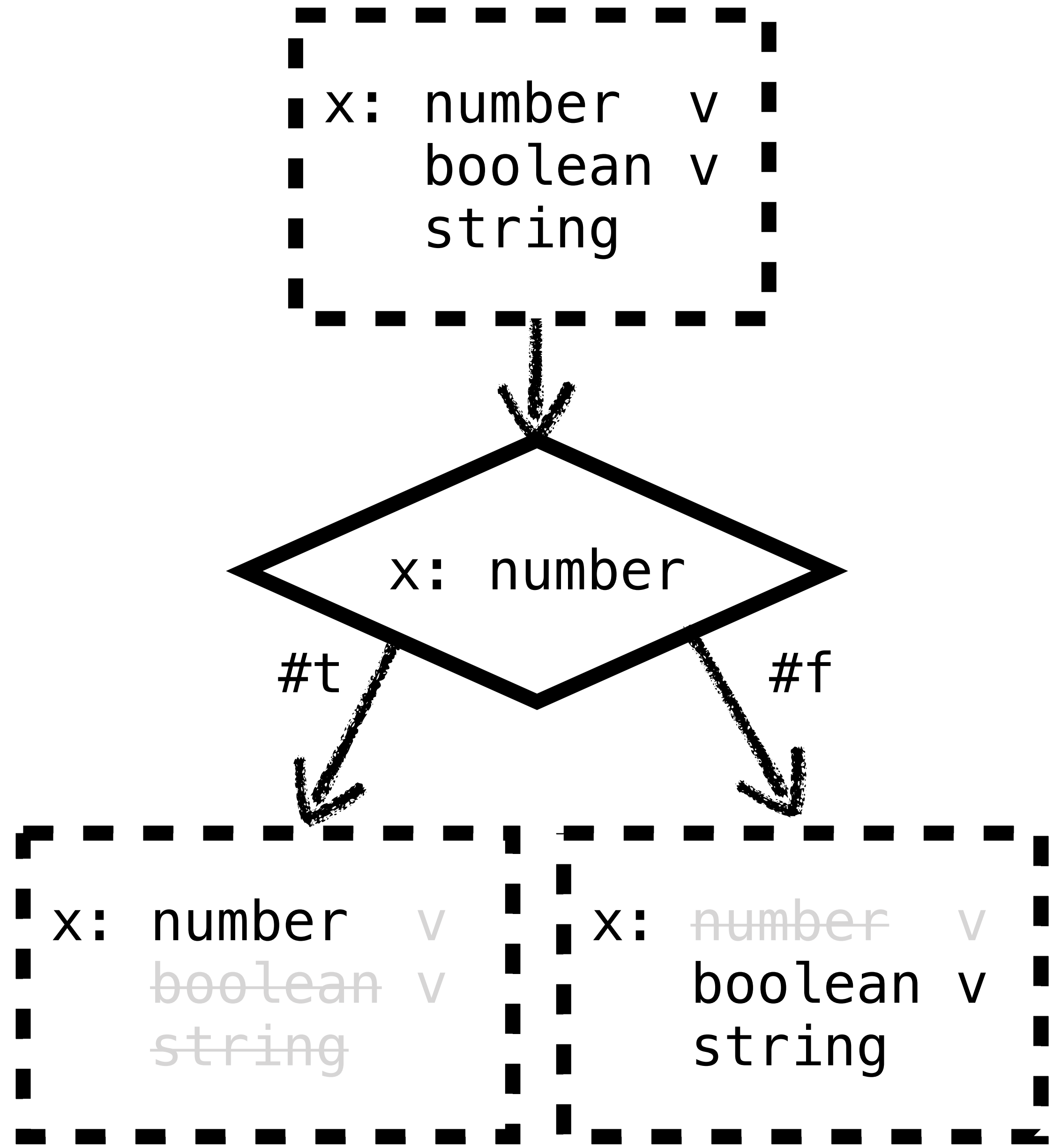
where $\sigma_j^\sharp = \texttt{refine}(e_j, b)(\sigma^\sharp)$ for $j = 0, 1$, $\tau_e^\sharp = [\![e]\!]_e^\sharp(\sigma^\sharp)$, and $\lfloor \tau^\sharp \rfloor$ returns $\{\tau\}$ if $\tau^\sharp$ denotes a singleton type τ, or returns $\varnothing$, otherwise.



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

59.2% Precision

93 Bugs Detected

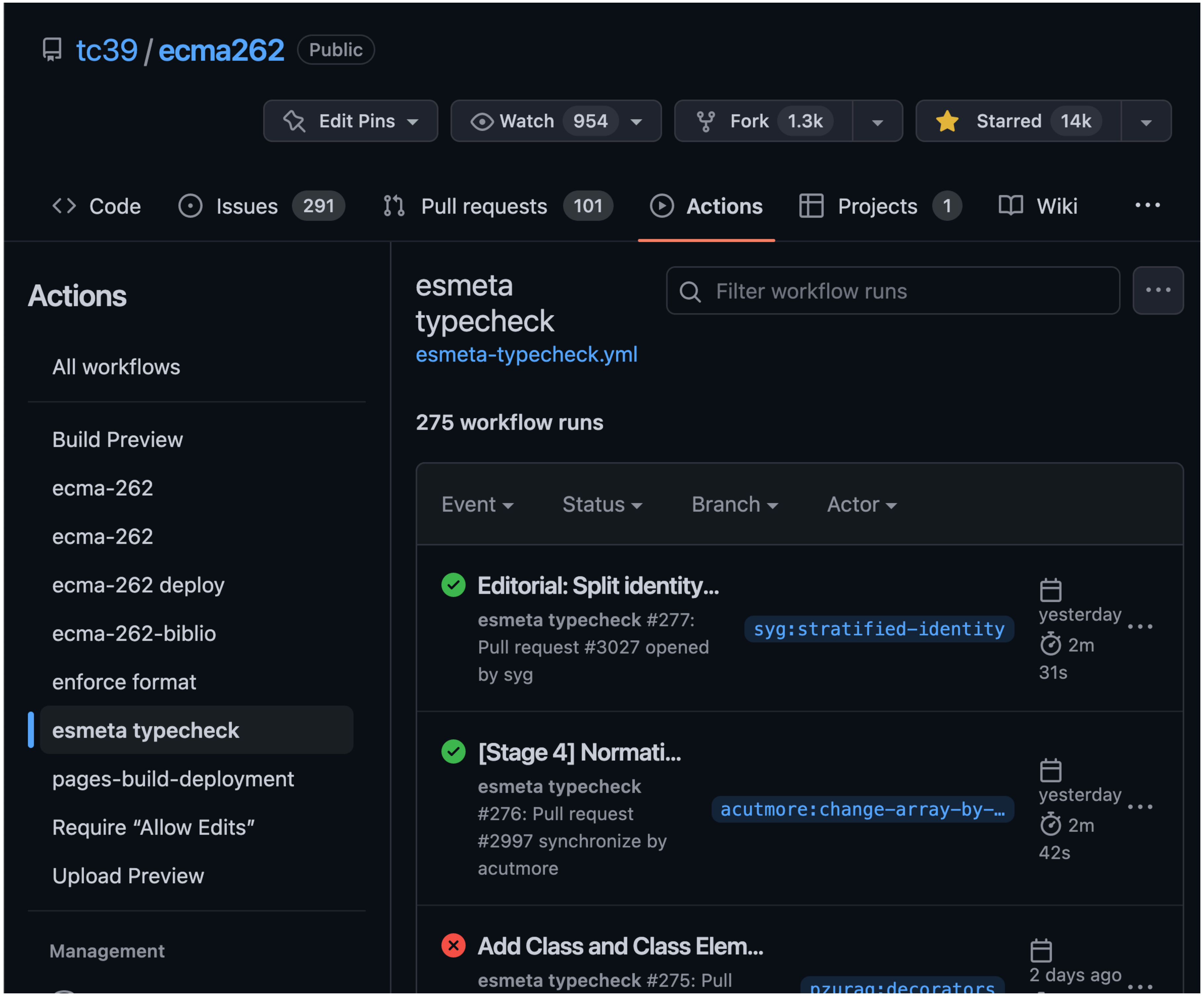
- Target: 864 versions of ECMA-262 in 3 years

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

CI System of ECMA-262



Advanced Refinement (Ongoing)

7.3.11 GetMethod (*V*, *P*)

1. Let *func* be ? GetV(*V*, *P*).
2. If *func* is either **undefined** or **null**, return **undefined**.
3. If IsCallable(*func*) is **false**, throw a **TypeError** exception.
4. Return *func*.

Specification Repair Tool (Idea)

20.3.2.28 Math.round (x)

1. Let 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 $\boxed{x \geq \text{-}0.5}$, return **-0**.

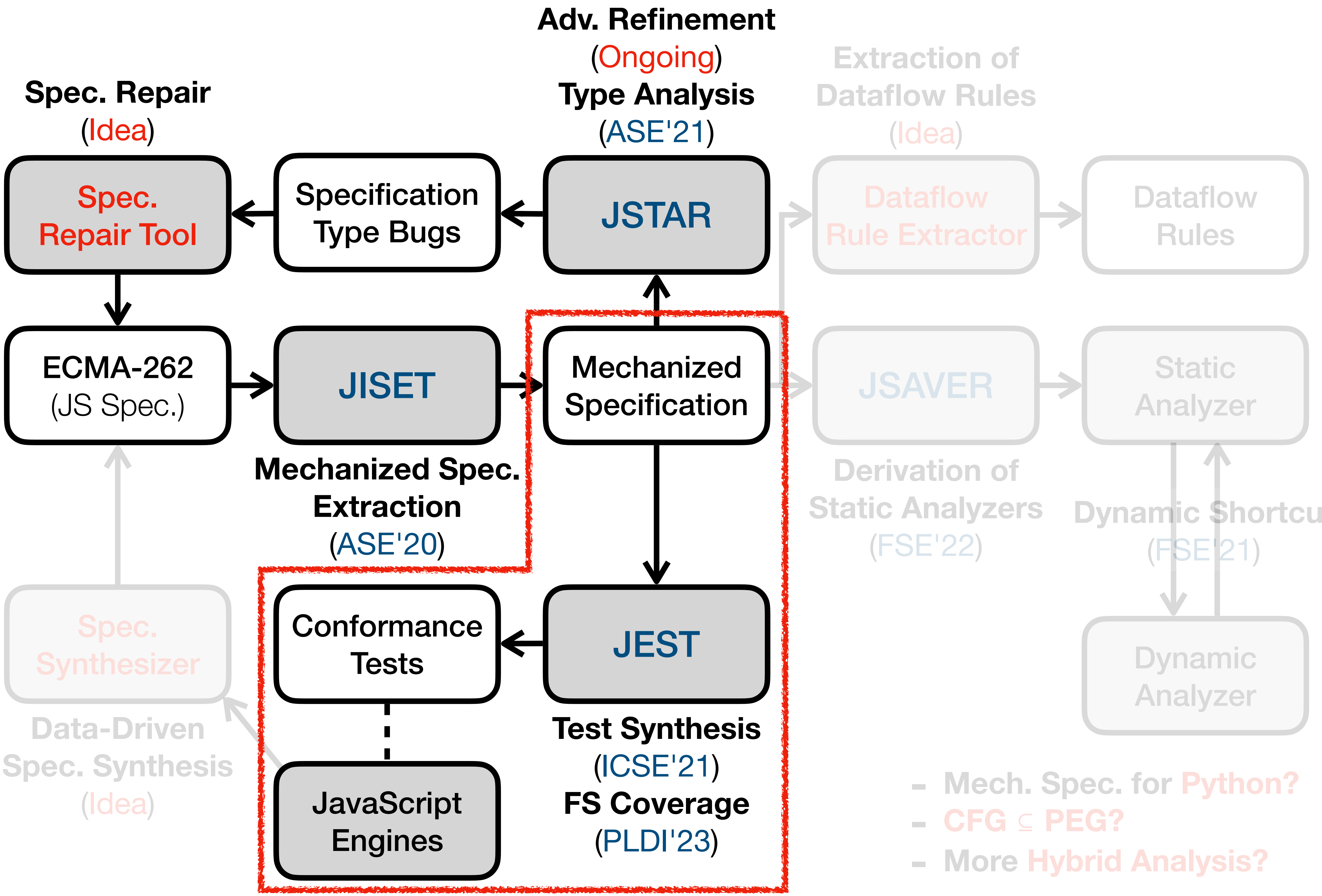
```
Type Mismatch for
numeric operator `>`
```

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

Auto Patch?

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

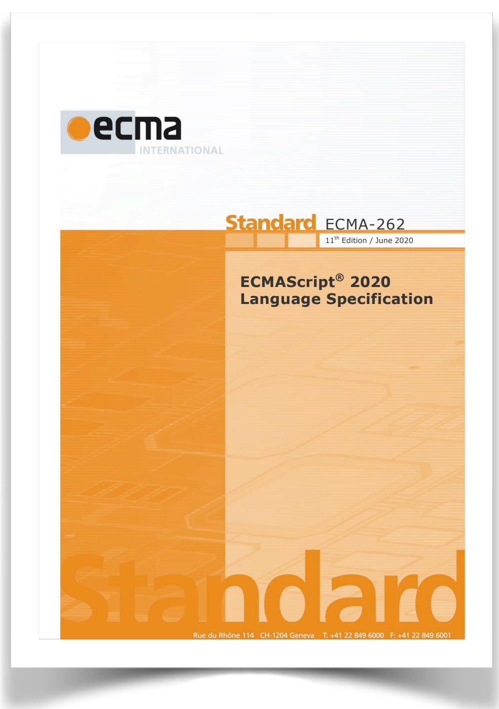
```
Math.round(true)  = 1
Math.round(false) = 0
```



[ICSE'21] J. Park, et al. "JEST: N+1-version Differential Testing of Both JavaScript Engines"

[PLDI'23] J. Park, et al. "Feature-Sensitive Coverage for Conformance Testing of Programming Language Implementations"

Conformance with Engines



ECMA-262

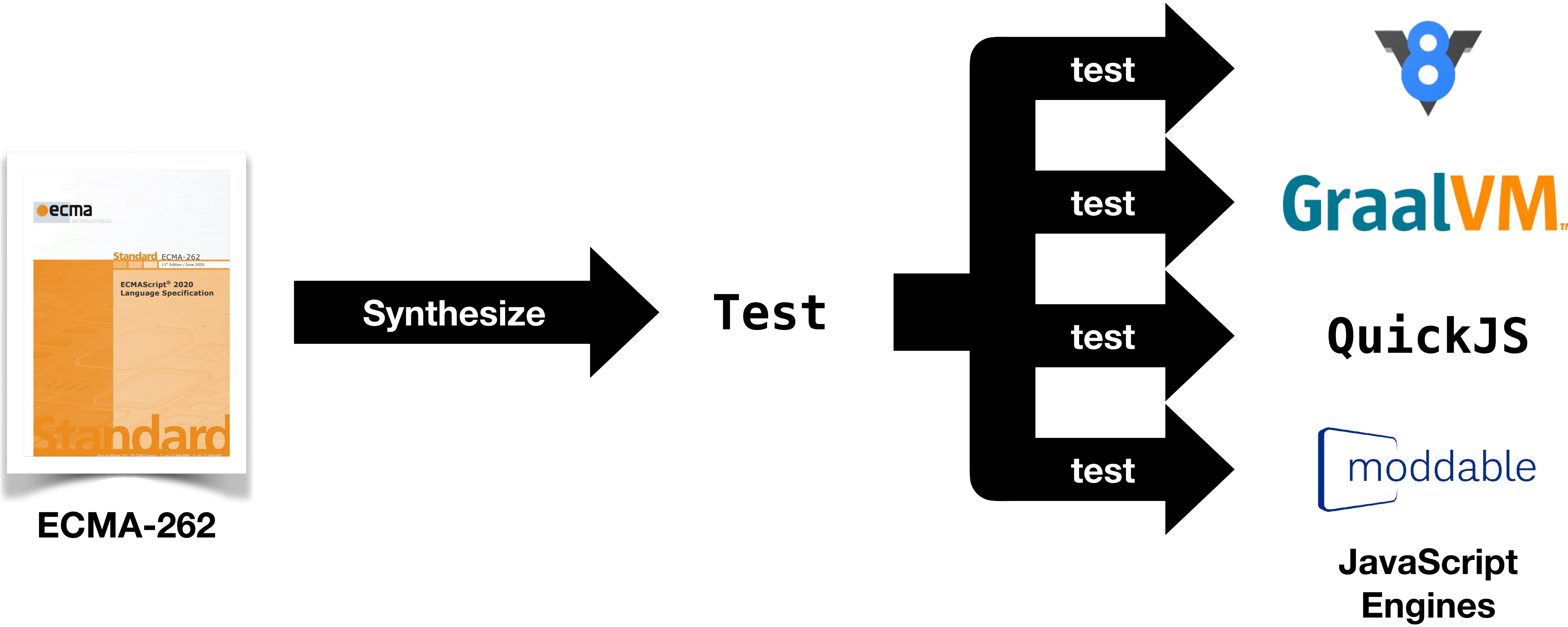
GraalVM

QuickJS

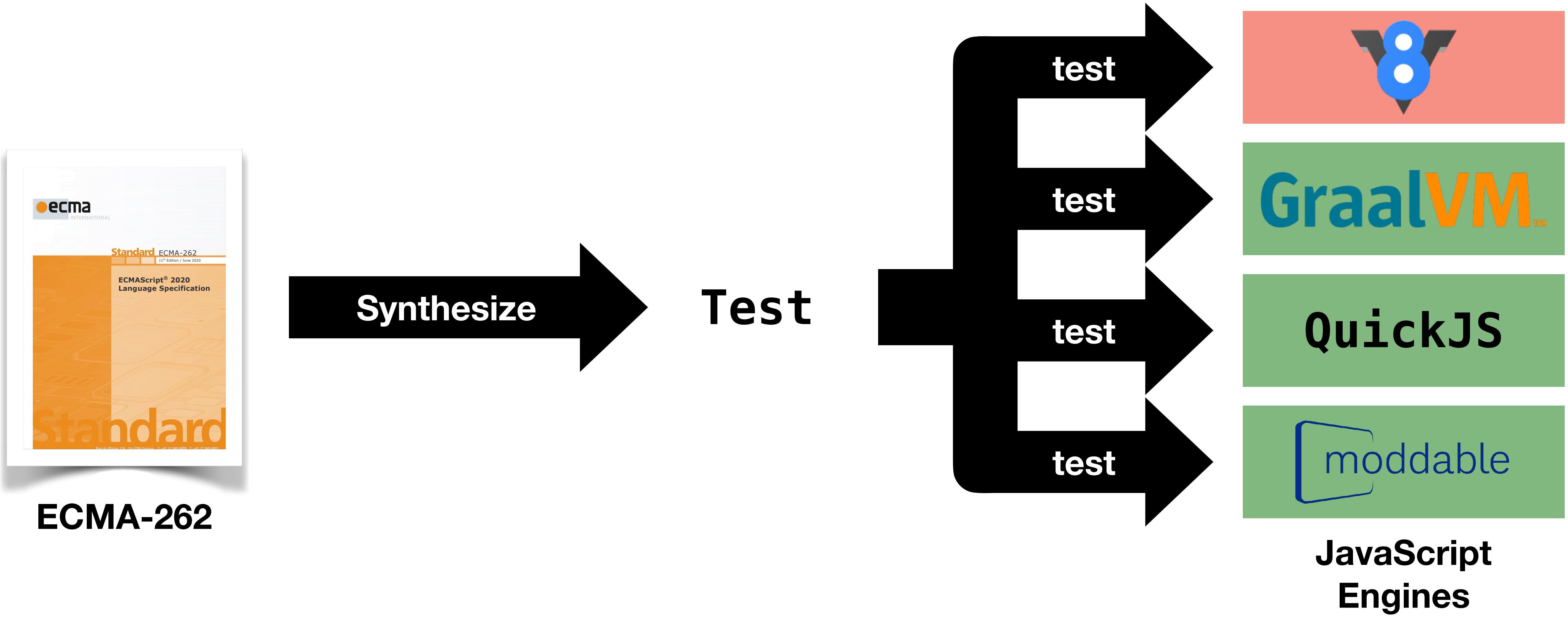
moddable

JavaScript
Engines

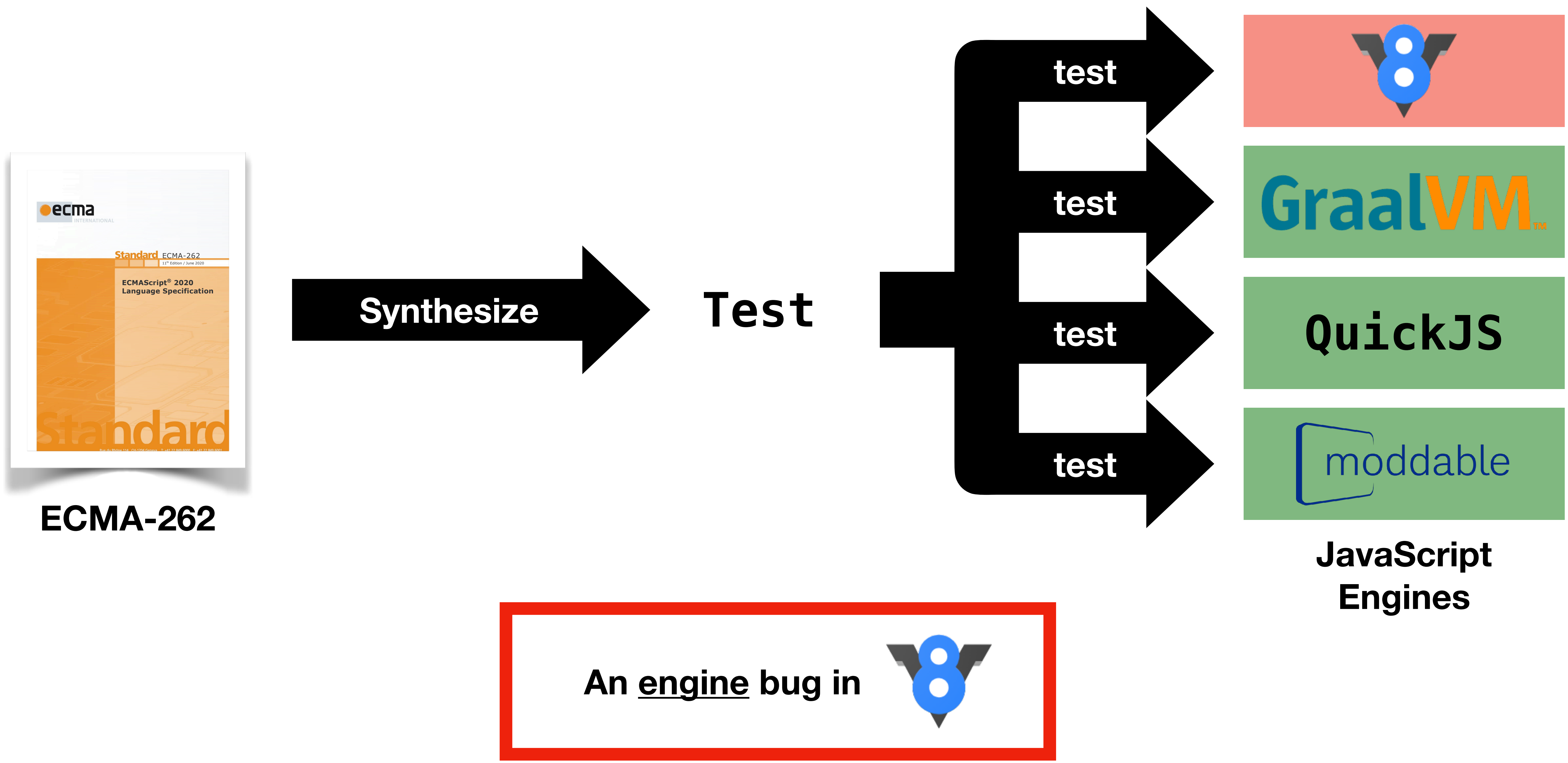
N+1-version Differential Testing



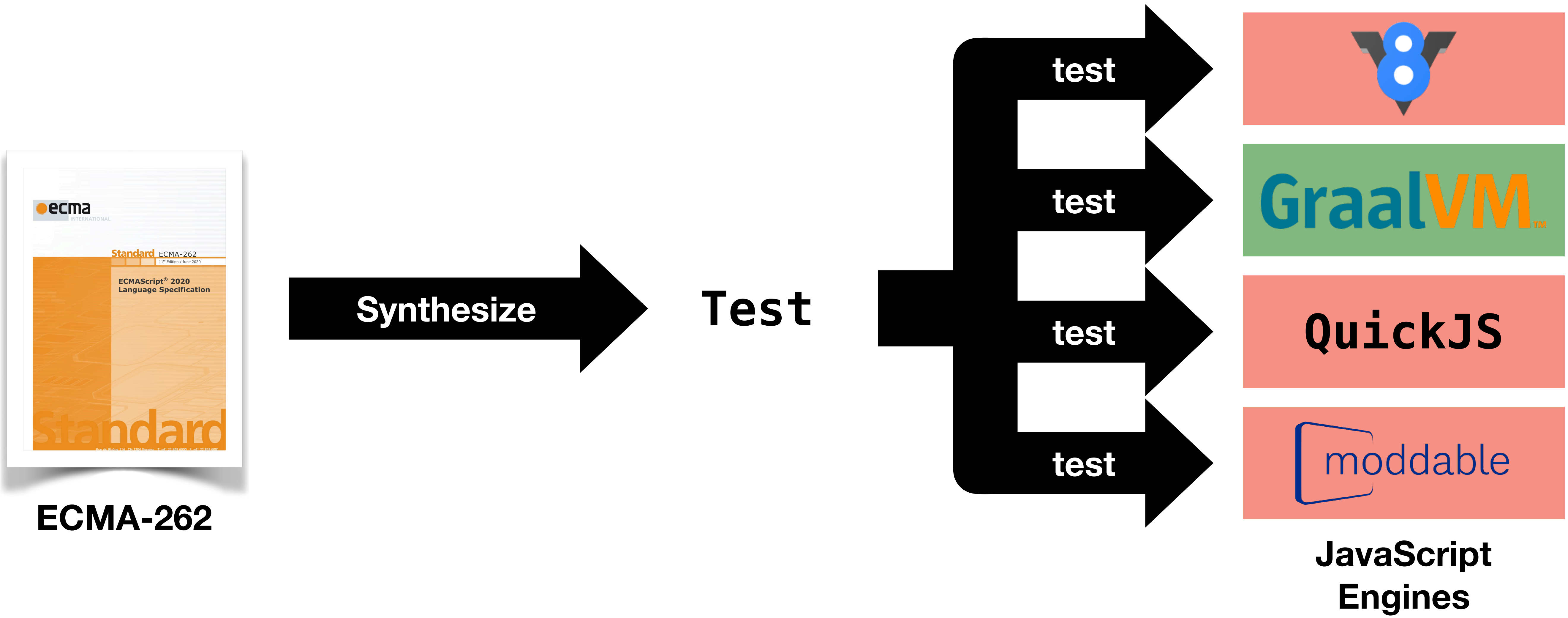
N+1-version Differential Testing



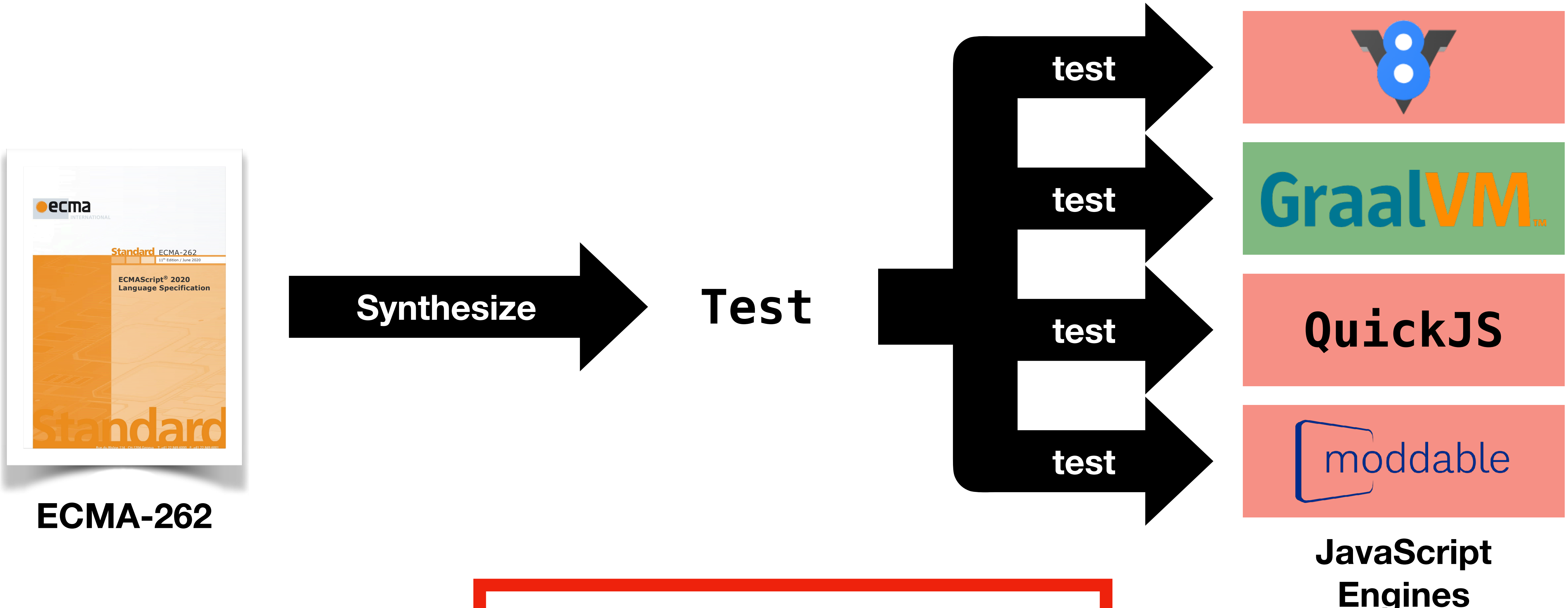
N+1-version Differential Testing



N+1-version Differential Testing



N+1-version Differential Testing

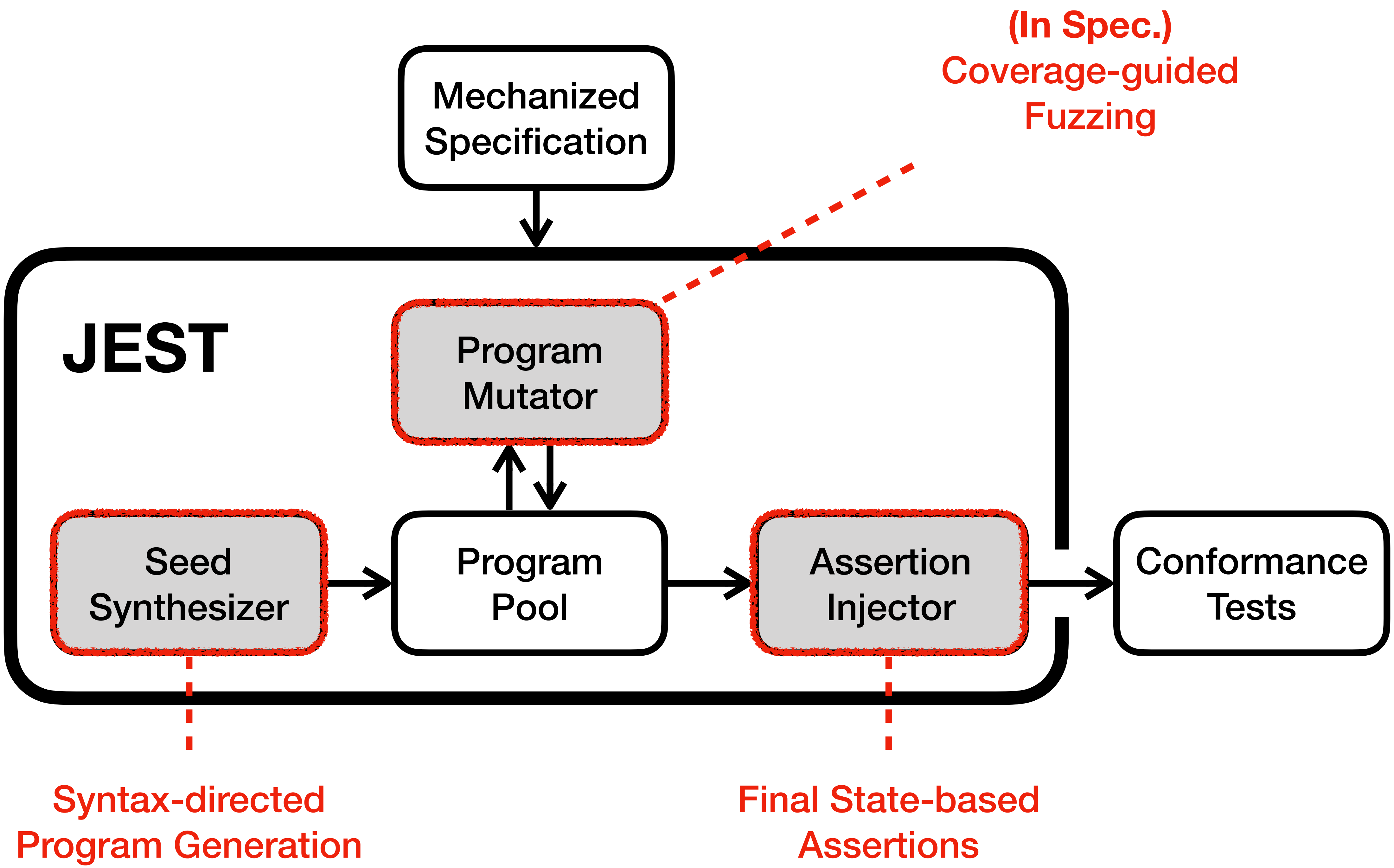


A specification bug in ECMA-262

An engine bug in GraalVM

JEST - ICSE'21

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



JEST - Coverage-guided Fuzzing (in Spec.)

7.1.3 ToNumeric (*value*)

1. Let *primValue* be $\boxed{?}$ ToPrimitive(*value*, number).
2. If Type(*primValue*) is BigInt, return *primValue*.
3. Return ? ToNumber(*primValue*).

```
0 + { valueOf() { throw 42; } }
```

JEST - Assertion Injection

```
  function f() {}

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

JEST - Evaluation

- JEST synthesized 1,700 conformance tests from ES11

44 Bugs in Engines

27 Bugs in Spec.

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

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

Feature-Sensitive (FS) Coverage - PLDI'23

AdditiveExpression **+** *MultiplicativeExpression*

7.1.3 ToNumeric (*value*)

1. Let *primValue* be $\boxed{?}$ ToPrimitive(*value*, number).
2. If Type(*primValue*) is BigInt, return *primValue*.
3. Return ? ToNumber(*primValue*).

```
0 + { valueOf() { throw 42; } }
```

Feature-Sensitive (FS) Coverage - PLDI'23

AdditiveExpression **+** *MultiplicativeExpression*

AdditiveExpression **-** *MultiplicativeExpression*

7.1.3 ToNumeric (*value*)

1. Let *primValue* be ? ToPrimitive(*value*, number).
2. If Type(*primValue*) is BigInt, return *primValue*.
3. Return ? ToNumber(*primValue*).

```
0 + { valueOf() { throw 42; } }
```

Feature-Sensitive (FS) Coverage - PLDI'23

AdditiveExpression **+** *MultiplicativeExpression*

AdditiveExpression **-** *MultiplicativeExpression*

7.1.3 ToNumeric (*value*)

1. Let *primValue* be ? ToPrimitive(*value*, number).
2. If Type(*primValue*) is BigInt, return *primValue*.
3. Return ? ToNumber(*primValue*).

```
0 + { valueOf() { throw 42; } }

0 - { valueOf() { throw 42; } }
```

FS Coverage - Evaluation

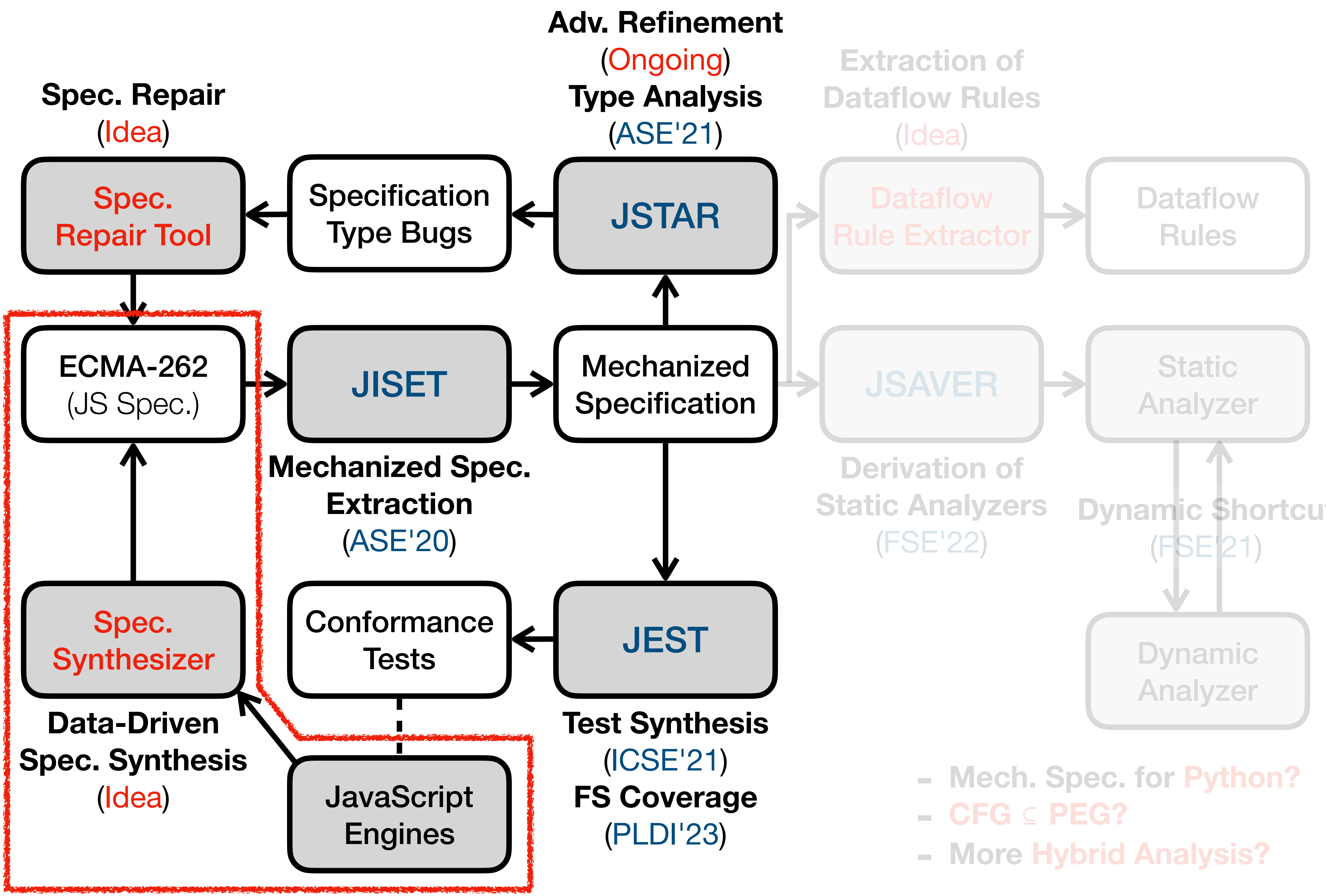
1.51x ~ 2.02x detected bugs

Table 2. Comparison of synthesized conformance tests guided by five graph coverage criteria

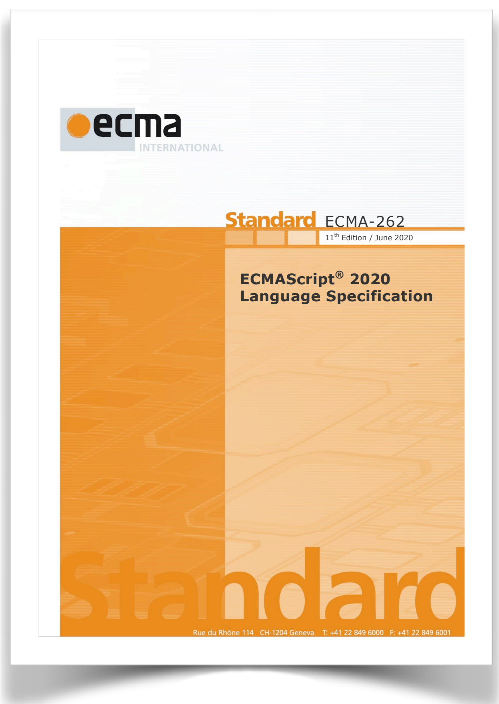
Coverage Criteria $C_{\mathbb{G}}$	# Covered k-F(CP)S-TR (k)			# Syn. Test	# Bug
	# Node	# Branch	# Total		
0-FS node-or-branch (0-fs)	10.0	5.6	15.6	2,111	55
1-FS node-or-branch (1-fs)	79.3	45.7	125.0	6,766	83
2-FS node-or-branch (2-fs)	1,199.8	696.3	1,896.1	97,423	102
1-FCPS node-or-branch (1-fcps)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

Table 1. Detected conformance bugs in JavaScript engines and transpilers

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**


Adv. Refinement
(Ongoing)
Type Analysis
(ASE'21)
Spec. Repair
(Idea)
Extraction of
Dataflow Rules
(Idea)
Spec.
Repair Tool
Specification
Type Bugs
JSTAR
Dataflow
Rule Extractor
Dataflow
Rules
ECMA-262
(JS Spec.)
JISET
Mechanized
Specification
JSAVER
Static
Analyzer
Mechanized Spec.
Extraction
(ASE'20)
Derivation of
Static Analyzers
(FSE'22)
Dynamic Shortcut
(FSE'21)
Spec.
Synthesizer
Conformance
Tests
JEST
Dynamic
Analyzer
Data-Driven
Spec. Synthesis
(Idea)
JavaScript
Engines
Test Synthesis
(ICSE'21)
FS Coverage
(PLDI'23)
- Mech. Spec. for Python?
- CFG ⊆ PEG?
- More Hybrid Analysis?

Specification (ECMA-262) Synthesis



ECMA-262

?

Synthesis

GraalVM

QuickJS

moddable

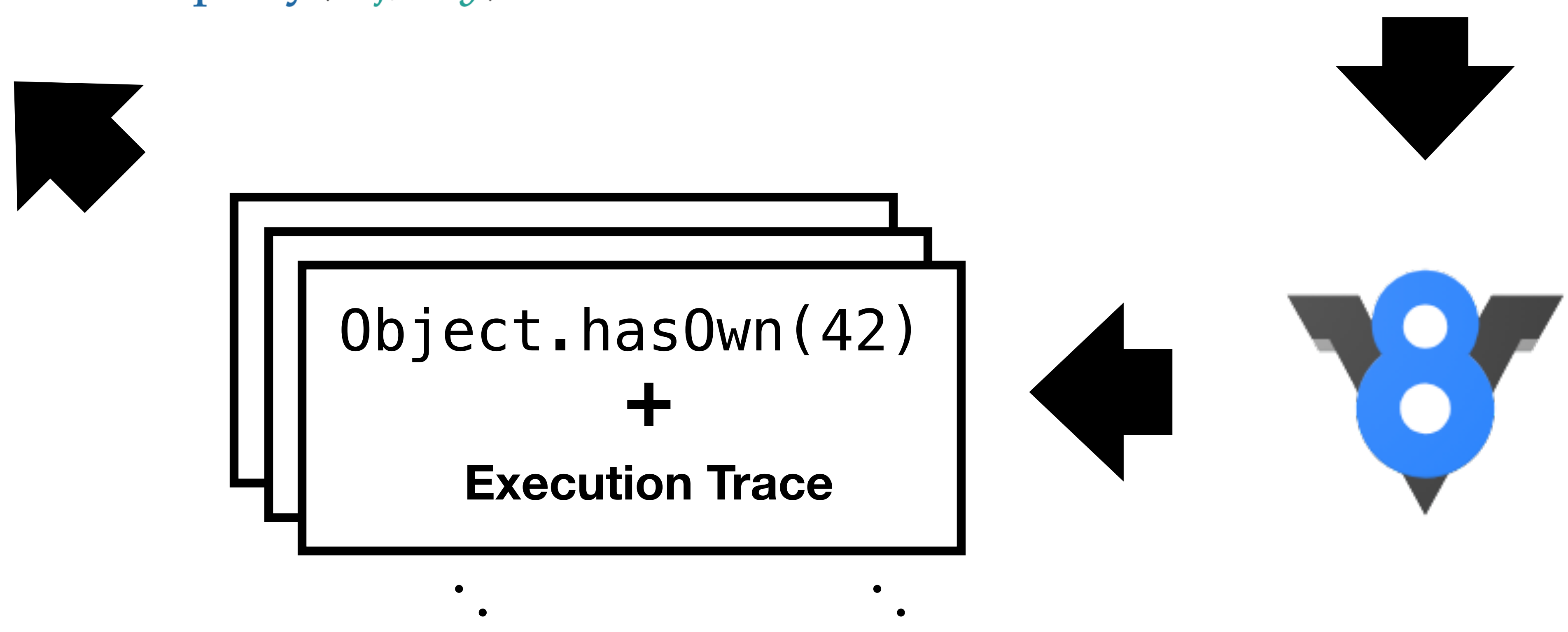
JavaScript
Engines

Specification (ECMA-262) Synthesis (Idea)

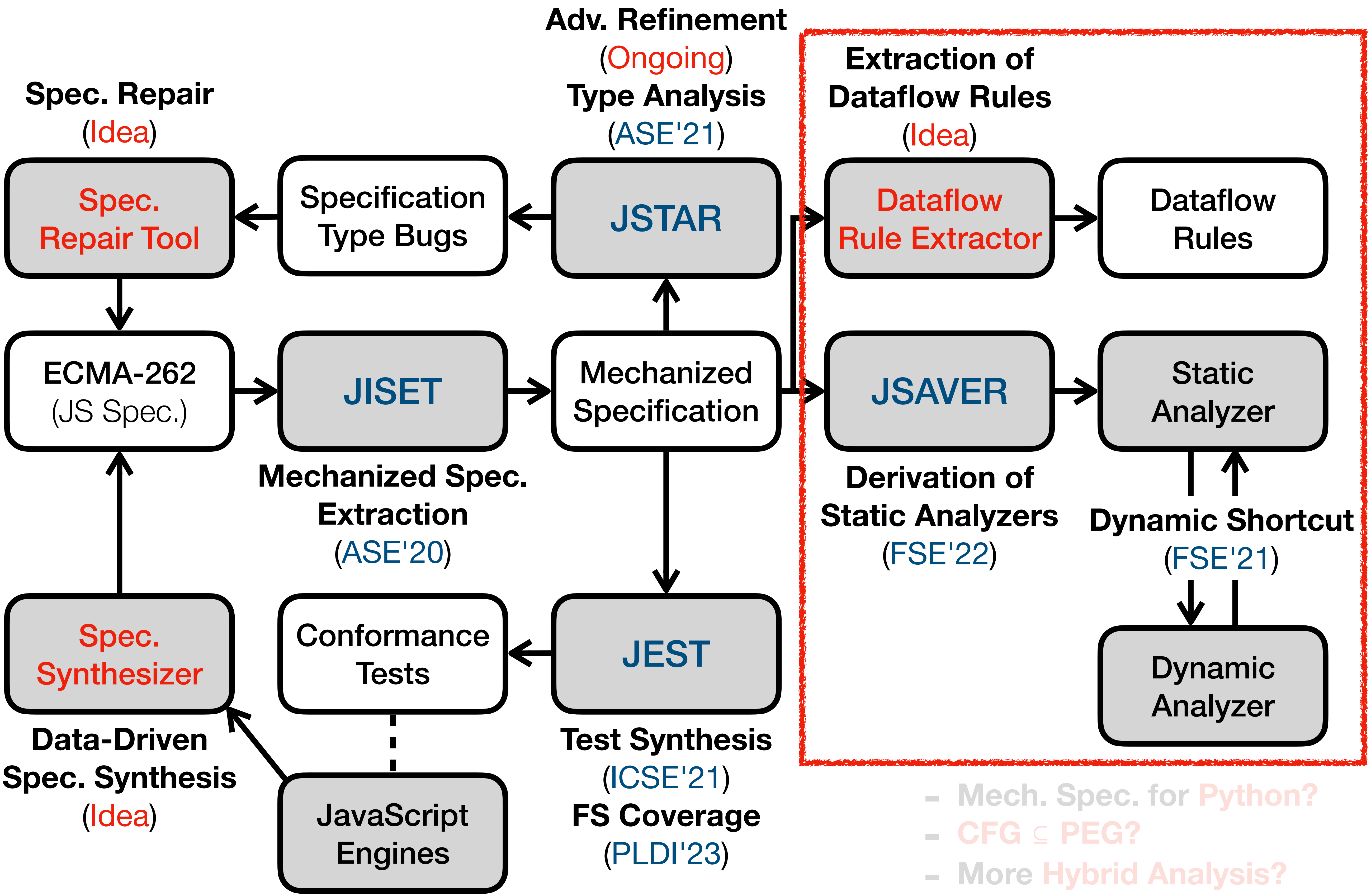
20.1.2.13 Object.hasOwn (*O*, *P*)

1. Let *obj* be ? ToObject(*O*).
2. Let *key* be ? ToPropertyKey(*P*).
3. Return ? HasOwnProperty(*obj*, *key*).

`Object.hasOwn`

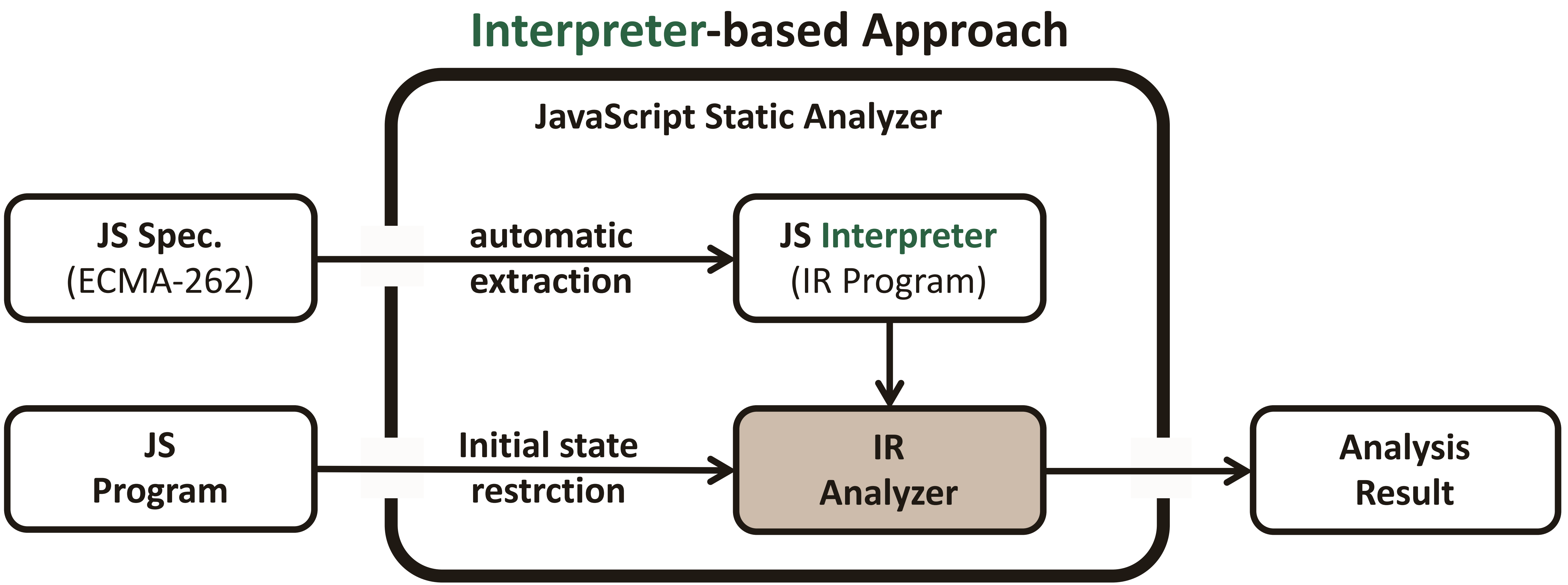


Related Work - [FSE'15] S. Heule, et al. "Mimic: Computing Models for Opaque Code"

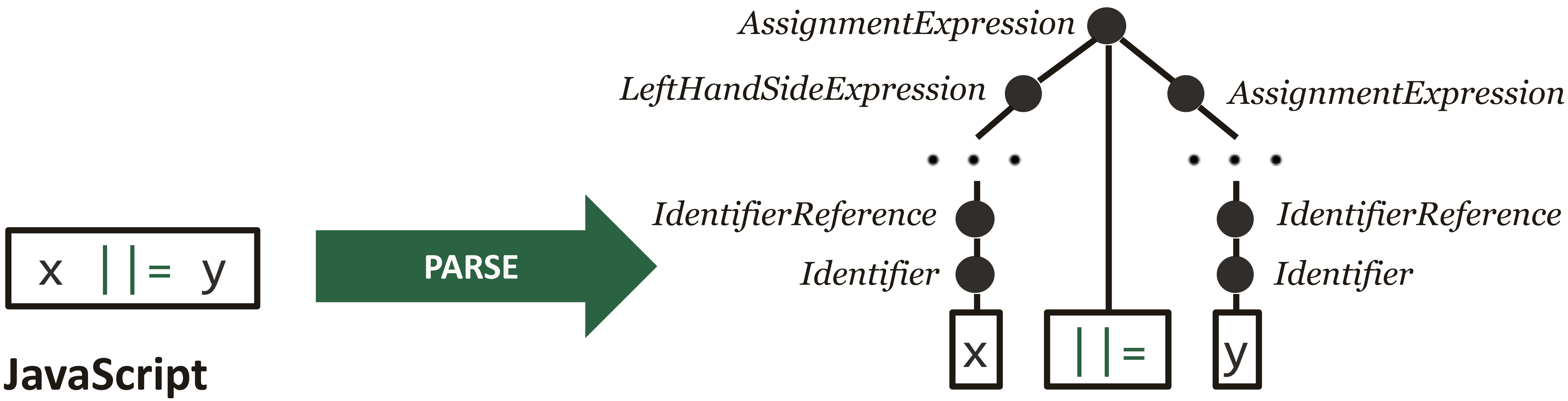


[FSE'21] J. Park, et al. "Accelerating JavaScript Static Analysis via Dynamic Shortcuts"
[FSE'22] J. Park, et al. "Automatically Deriving JavaScript Static Analyzers from Language Specifications"

Meta-level Static Analysis



Meta-level Static Analysis - Example

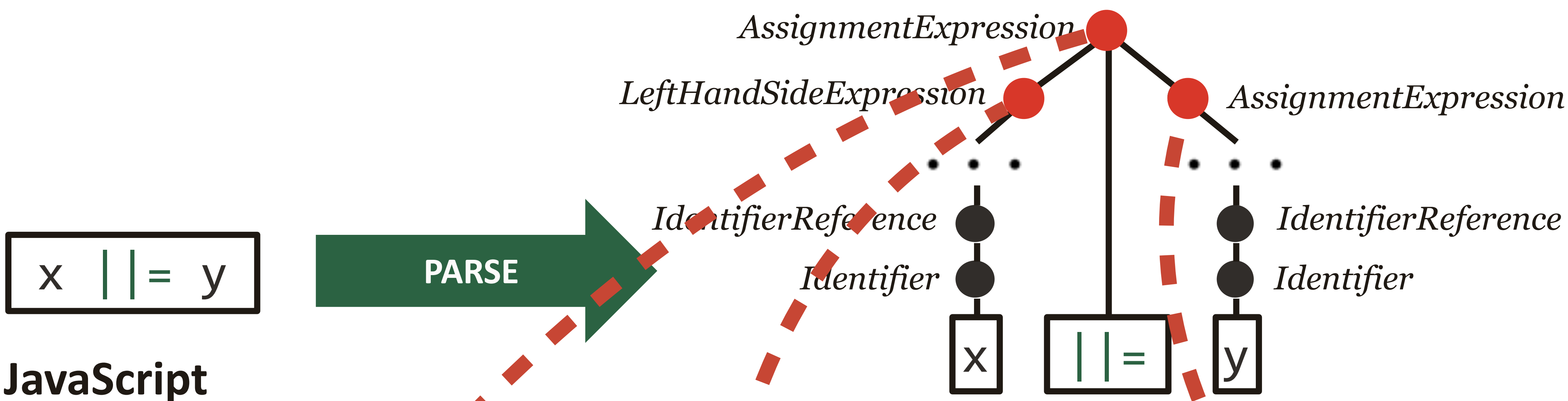


JavaScript

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
  ...
}
```

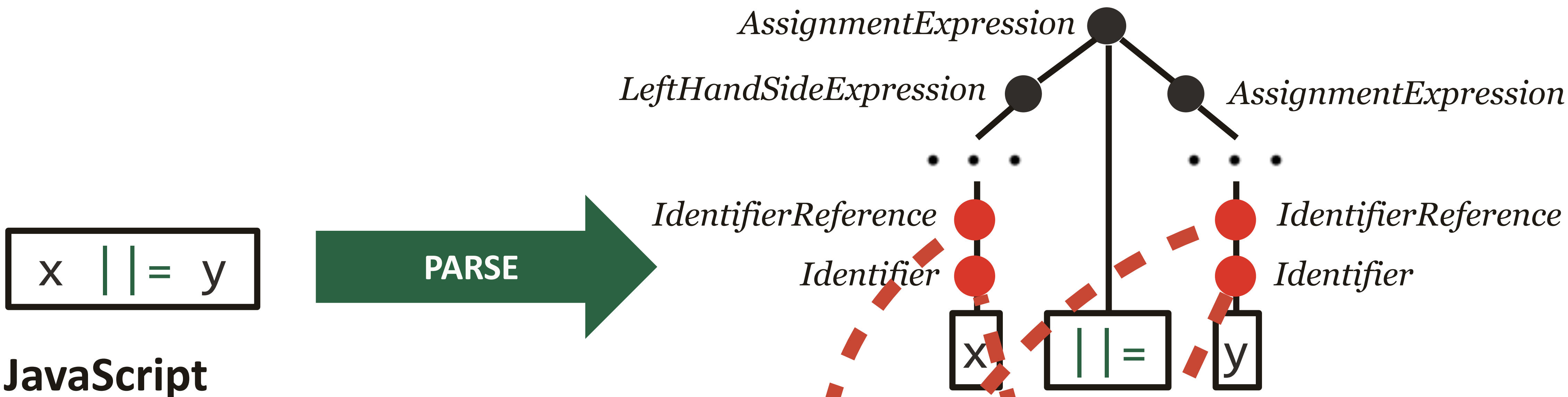
Meta-level Static Analysis - Example



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
  ...
}
```

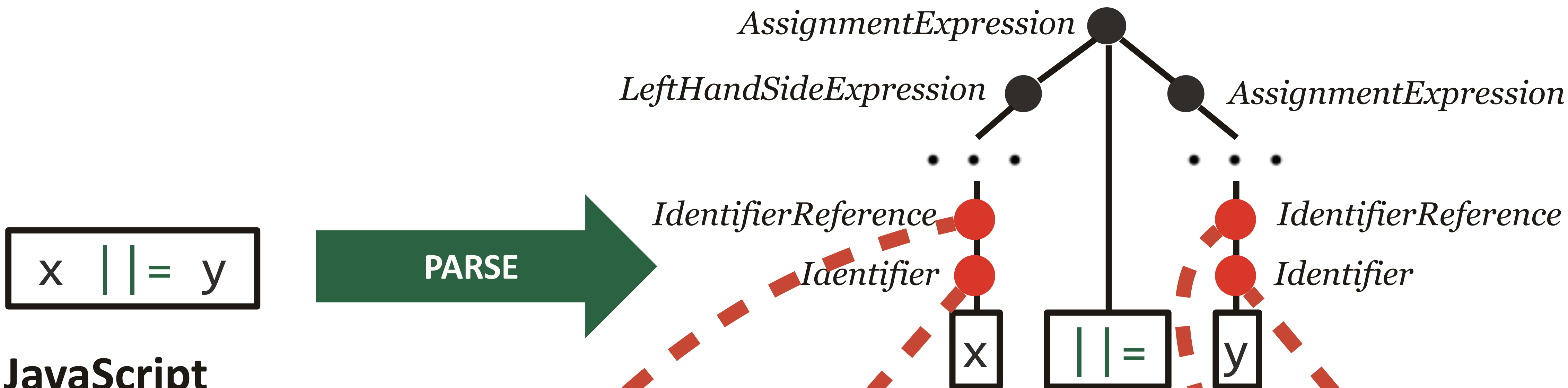
AST Sensitivity



IR_{ES}

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

AST Sensitivity



JavaScript

IR_{ES}

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

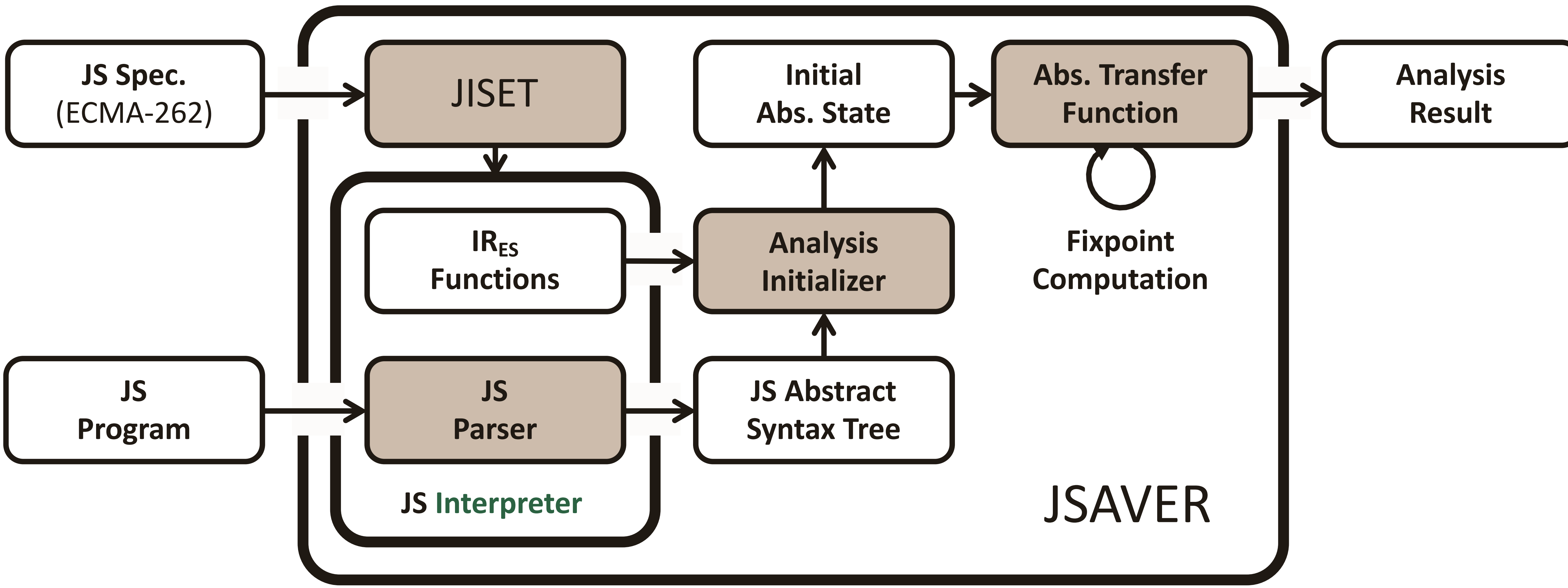
```
syntax def IdentifierReference[0]
  .Evaluation(this Identifier) {
  return [?
    (ResolveBinding
      (Identifier.StringValue)
    )
  ]
}
```

AST Sensitivity

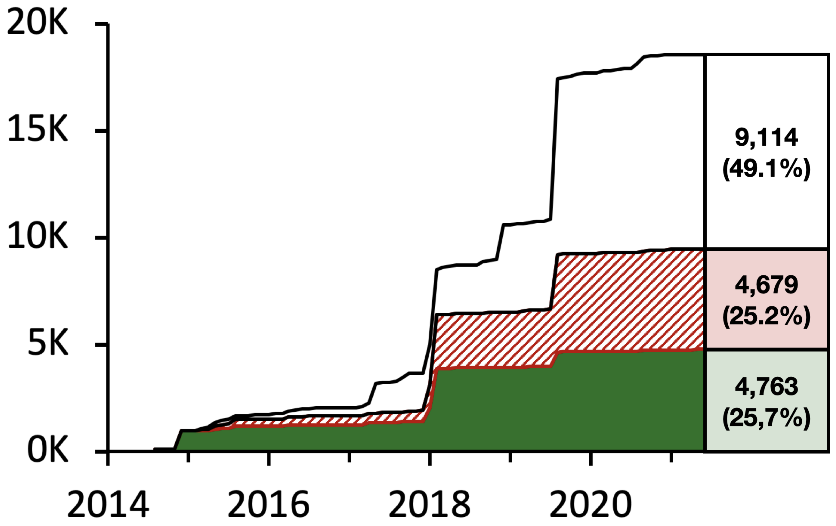
JavaScript	AST Sensitivity in IR_{ES}
Flow-Sensitivity	$\delta^{\text{js-flow}}(t_\perp) = \{\sigma = (\_, \_, \overline{c}, \_) \in \mathbb{S} \mid \text{ast}(\overline{c}) = t_\perp\}$
k-Callsite-Sensitivity	$\delta^{\text{js-}k\text{-cfa}}([t_1, \cdots, t_n]) = \{\sigma = (\_, \_, \overline{c}, \_) \in \mathbb{S} \mid n \leq k \land (n = k \lor \text{js-ctxt}^{n+1}(\overline{c}) = \perp) \land \forall 1 \leq i \leq n.\ \text{ast} \circ \text{js-ctxt}^{i}(\overline{c}) = t_i\}$

JSAVER - FSE'22

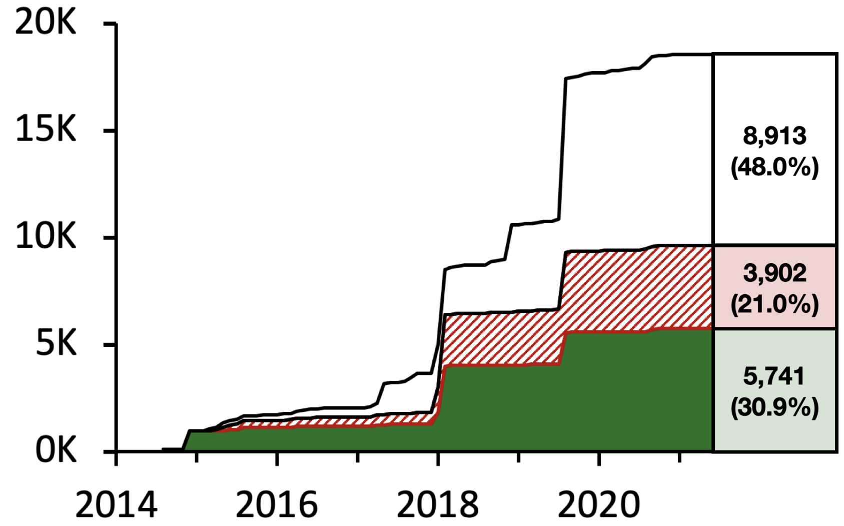
(**J**avaScript **S**tatic **A**nalyzer **v**ia **E**CMAScript **R**epresentation)



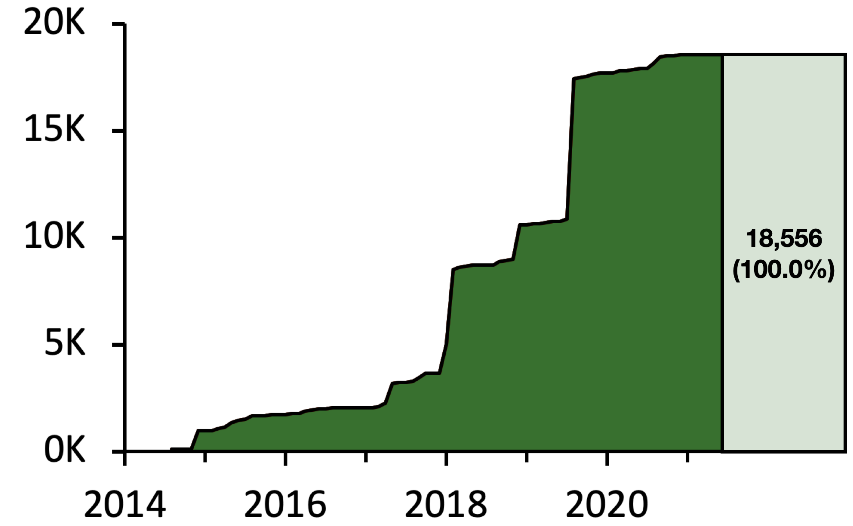
JSAVER - Evaluation (RQ1: Soundness)



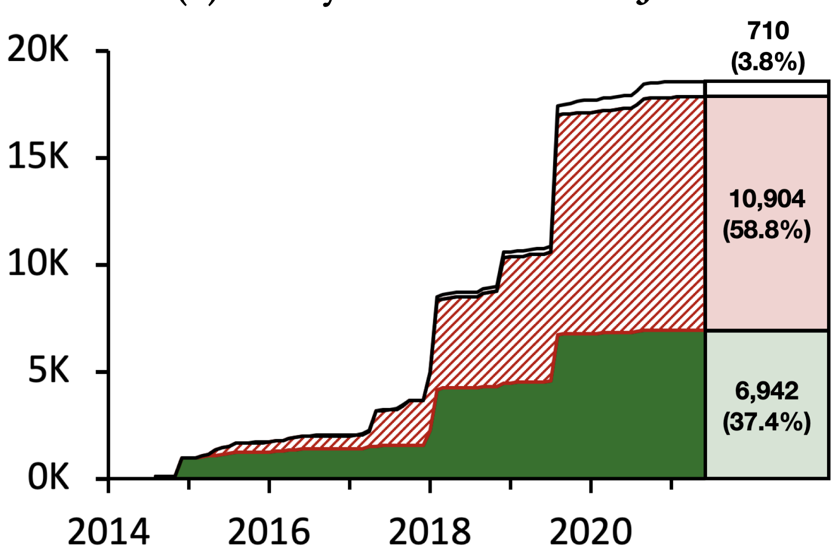
(a) Analysis results of TAJS



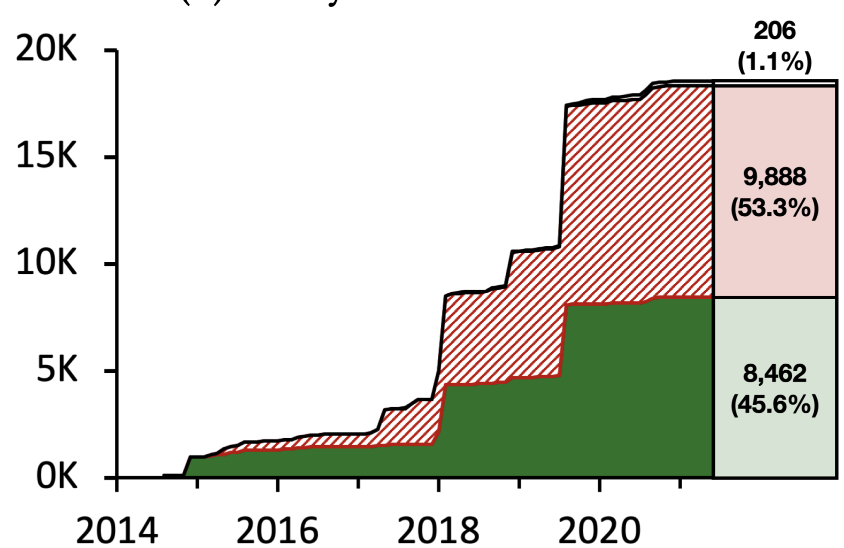
(b) Analysis results of SAFE



(c) Analysis results of JSA_{ES12}



(d) Analysis results of TAJS with Babel



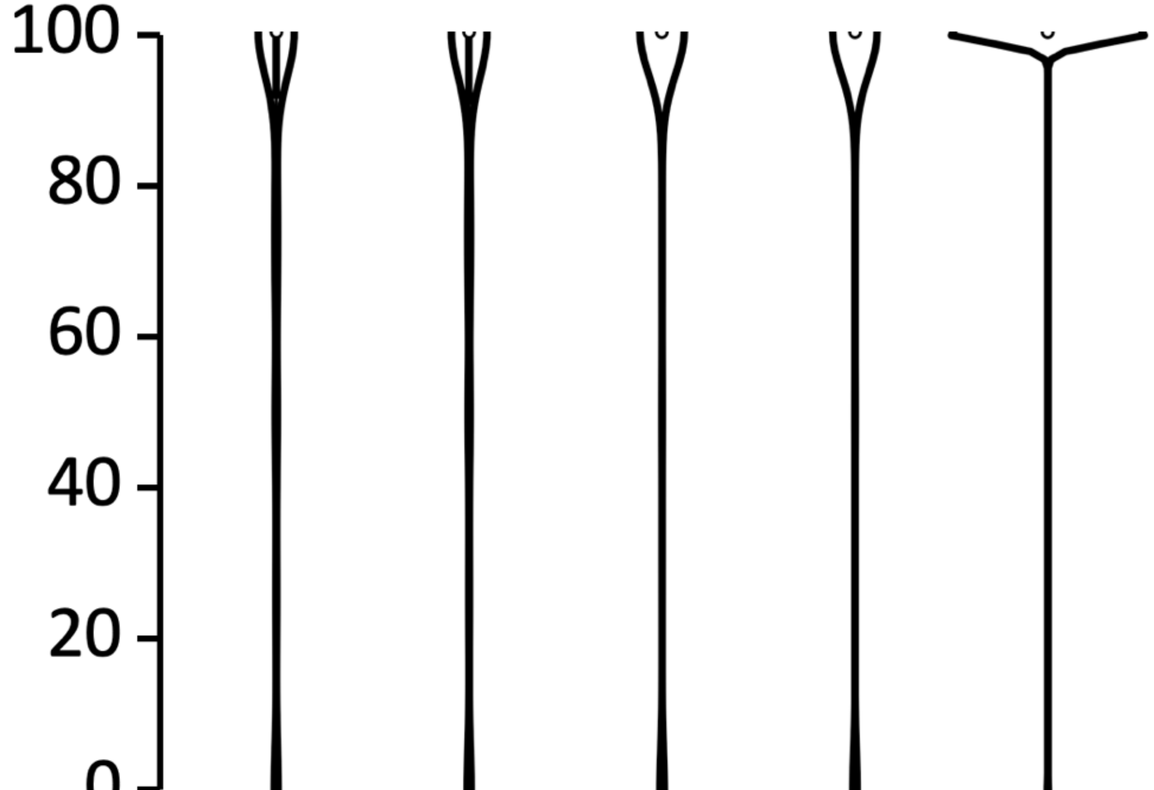
(e) Analysis results of SAFE with Babel

legend : error / unsound / sound

x-axis : creation time (year)
y-axis : # tests

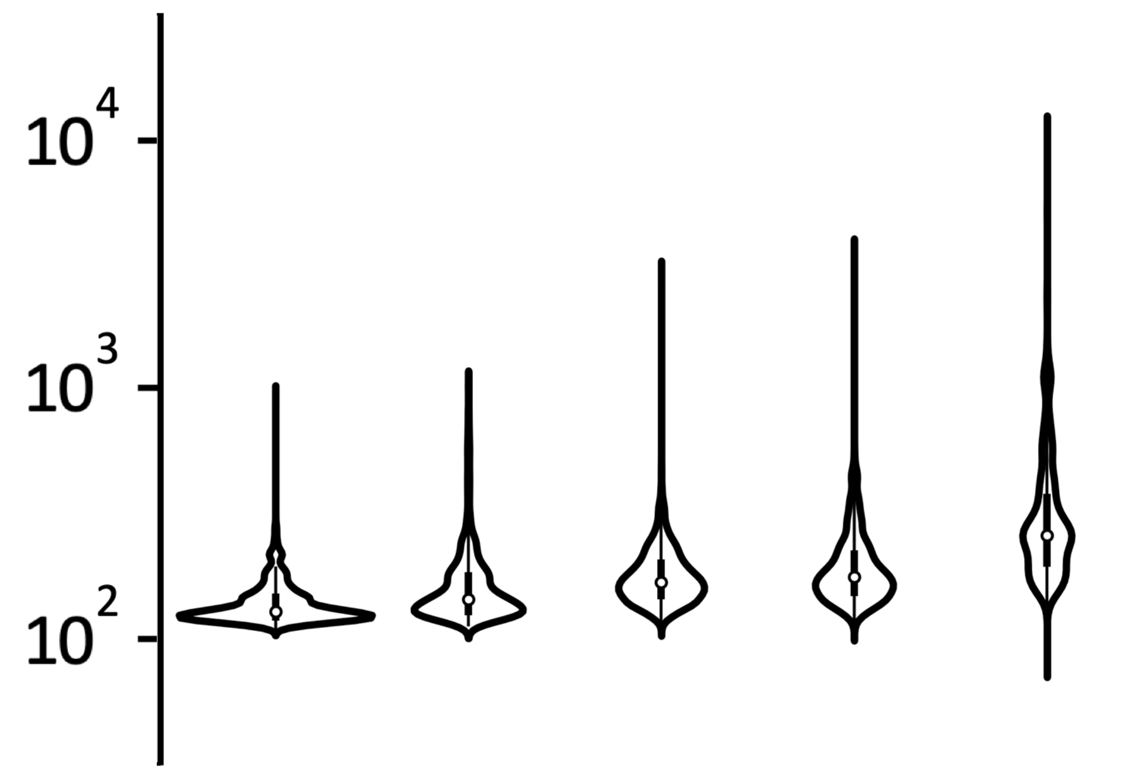
JSAVER - Evaluation (RQ2: Prec. & Perf.)

	TAJS	TAJS (Babel)	SAFE	SAFE (Babel)	JSA_{ES12}
Avg. :	85.1%	84.9%	89.9%	89.9%	99.5%



(a) The analysis precision

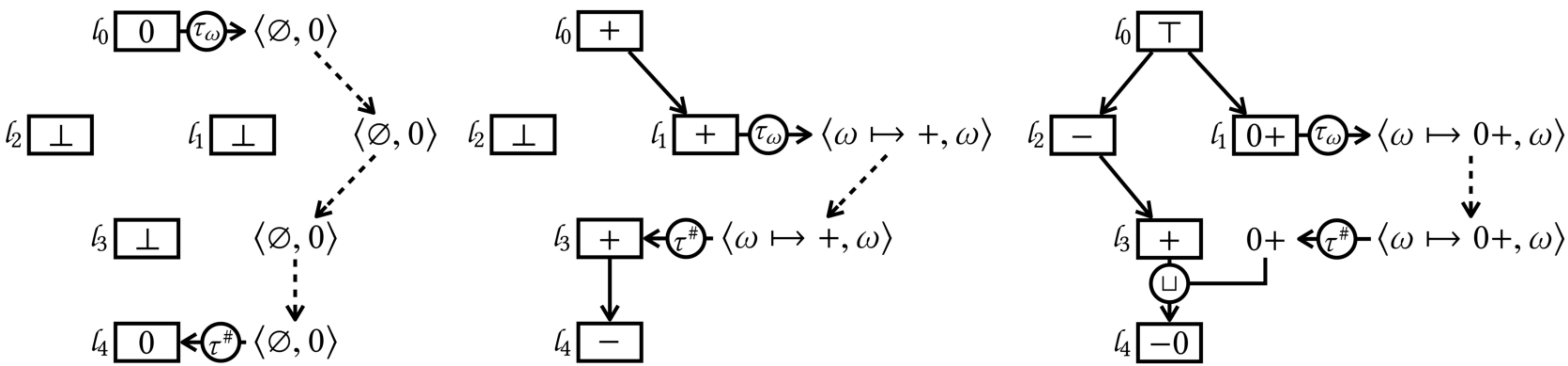
	TAJS	TAJS (Babel)	SAFE	SAFE (Babel)	JSA_{ES12}
Avg. :	139 ms	169 ms	181 ms	199 ms	357 ms



(b) The analysis performance

Dynamic Shortcut - FSE'21

- ℓ_0 `if ( x ≥ 0 )` else
- ℓ_1 `x = x;`
- ℓ_2 `x = −x;`
- ℓ_3 `x = −x;`
- ℓ_4



(b) $\mathbf{x} = 0$ (c) $\mathbf{x} > 0$ (d) $\mathbf{x} \in \mathbb{N}$

Dynamic Shortcut - Evaluation

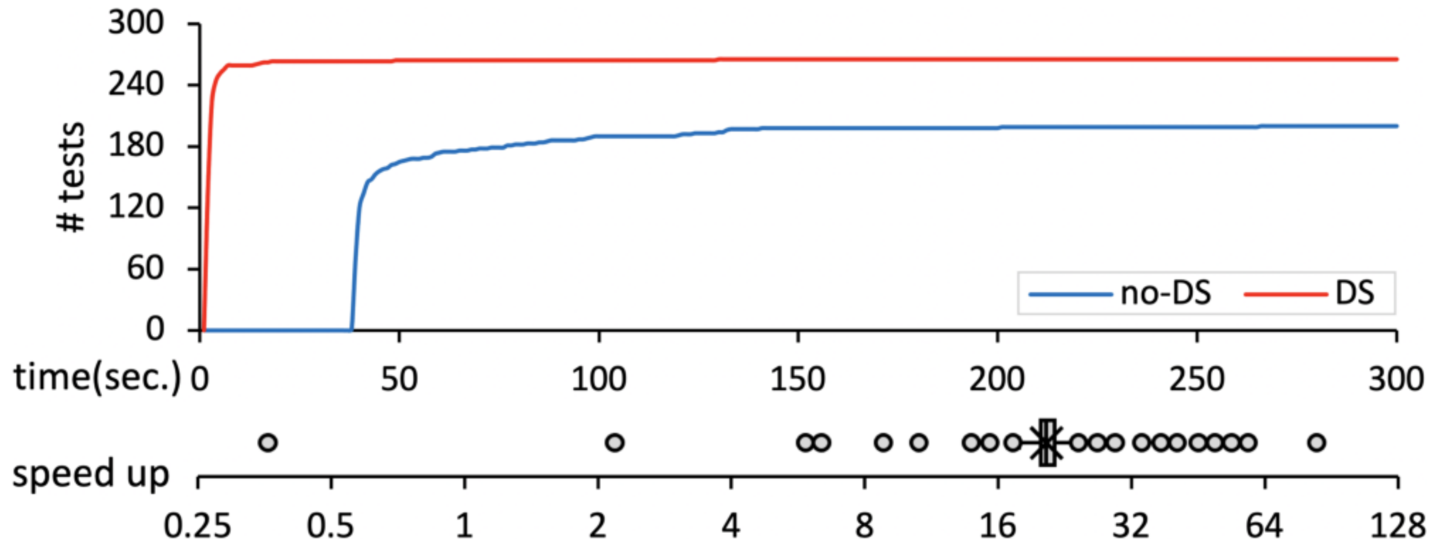


Figure 6: Analysis time for Lodash 4 *original* tests without (no-DS) and with (DS) dynamic shortcuts within 5 minutes

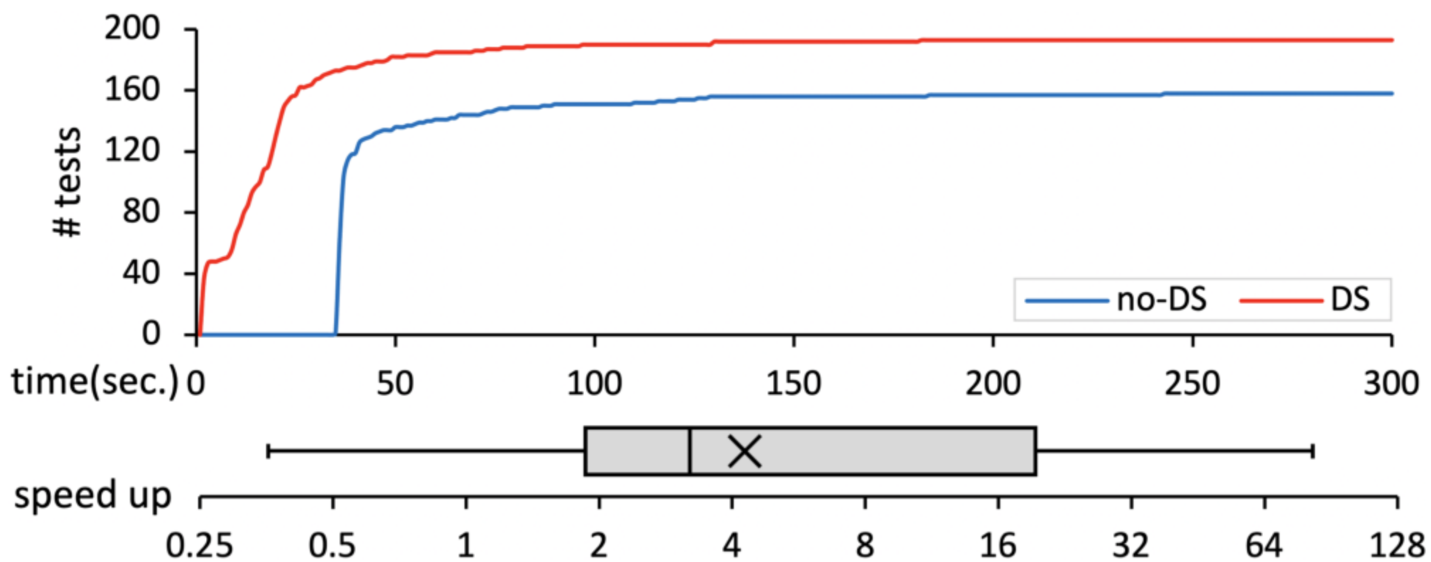
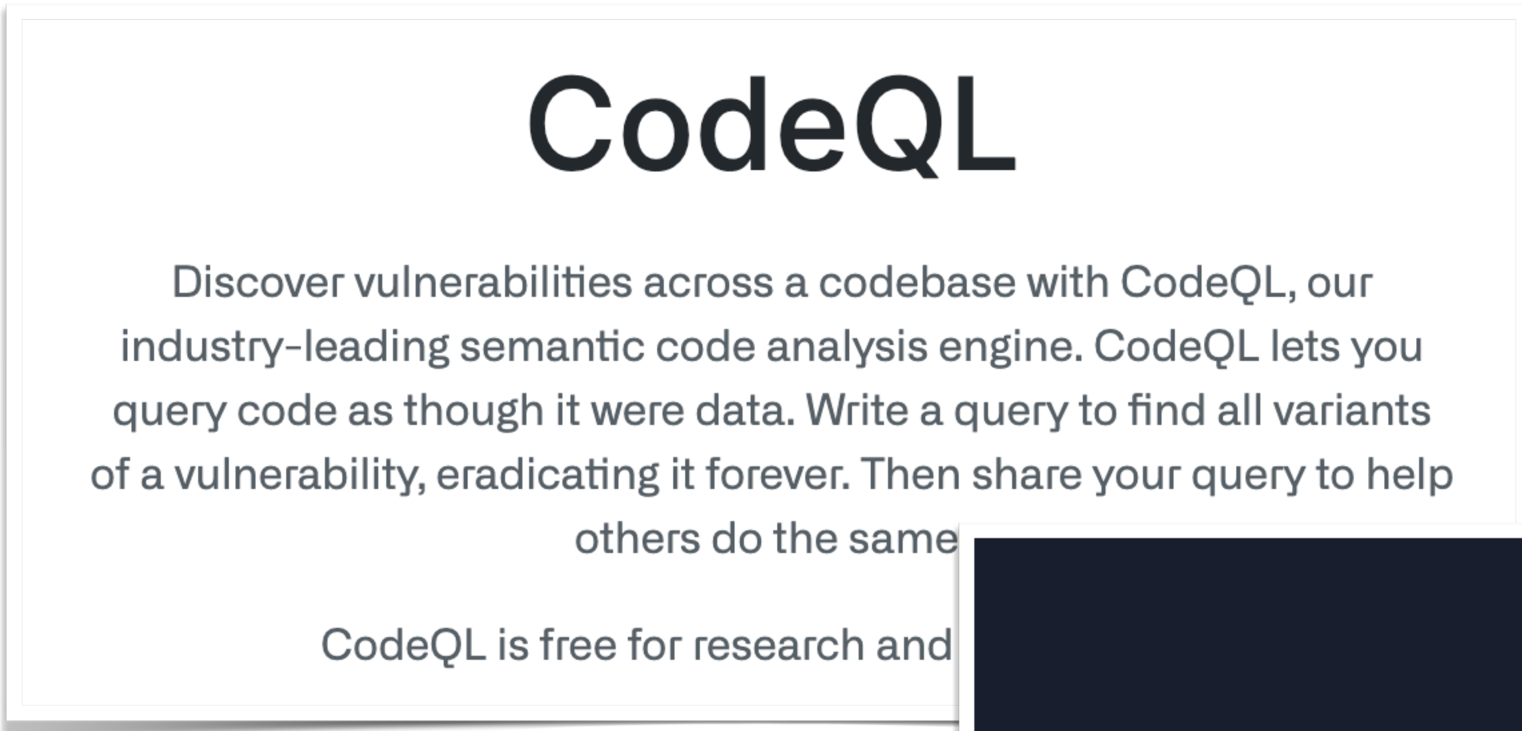
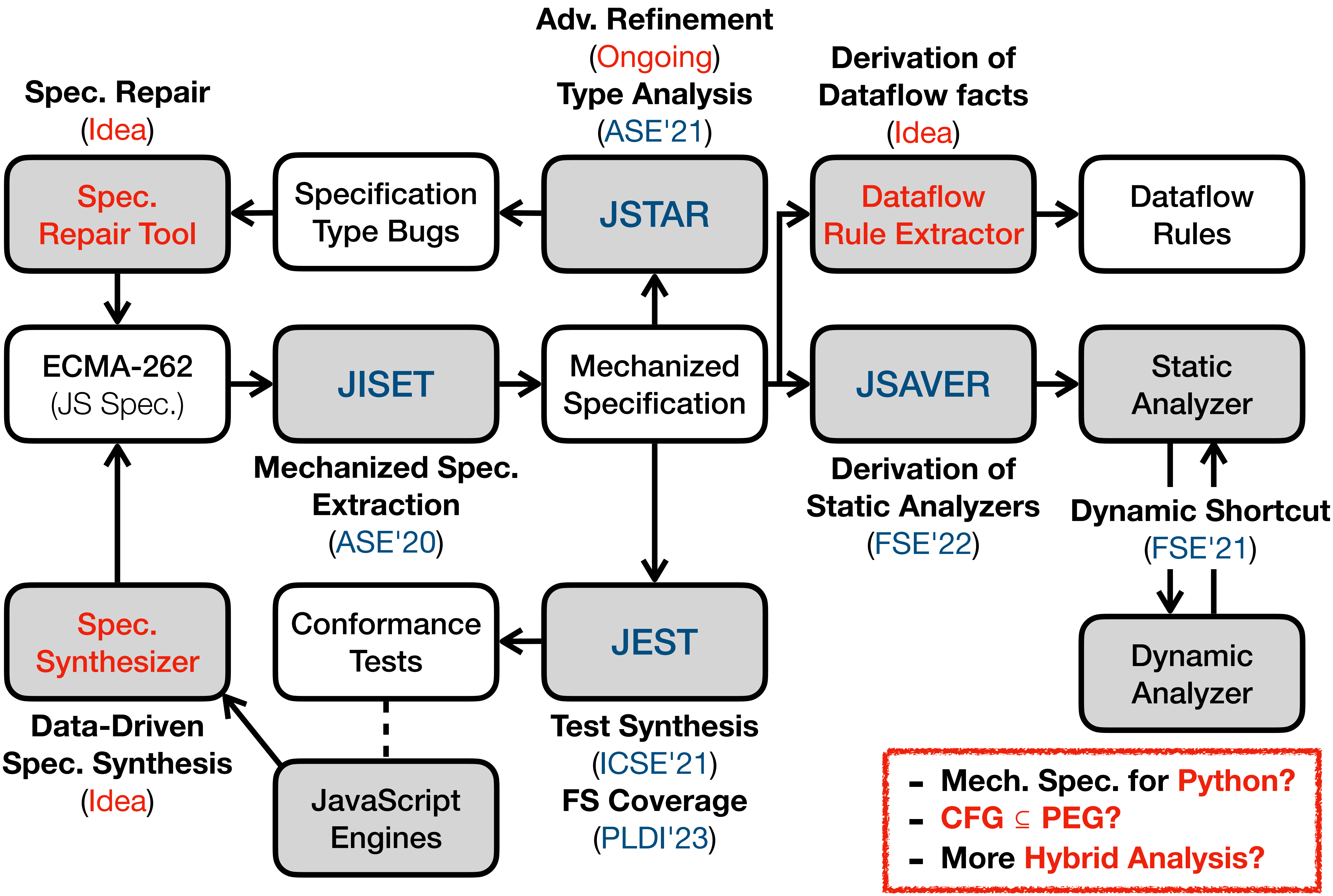


Figure 7: Analysis time for Lodash 4 *abstracted* tests without (no-DS) and with (DS) dynamic shortcuts within 5 minutes

Extraction of Dataflow Rules (Idea)





Adv. Refinement
(Ongoing)
Type Analysis
(ASE'21)
Spec. Repair
(Idea)
Derivation of
Dataflow facts
(Idea)
Spec.
Repair Tool
Specification
Type Bugs
JSTAR
Dataflow
Rule Extractor
Dataflow
Rules
ECMA-262
(JS Spec.)
JISET
Mechanized
Specification
JSAVER
Static
Analyzer
Mechanized Spec.
Extraction
(ASE'20)
Derivation of
Static Analyzers
(FSE'22)
Dynamic Shortcut
(FSE'21)
Spec.
Synthesizer
Conformance
Tests
JEST
Dynamic
Analyzer
Data-Driven
Spec. Synthesis
(Idea)
JavaScript
Engines
Test Synthesis
(ICSE'21)
FS Coverage
(PLDI'23)
- Mech. Spec. for Python?
- CFG ⊆ PEG?
- More Hybrid Analysis?