



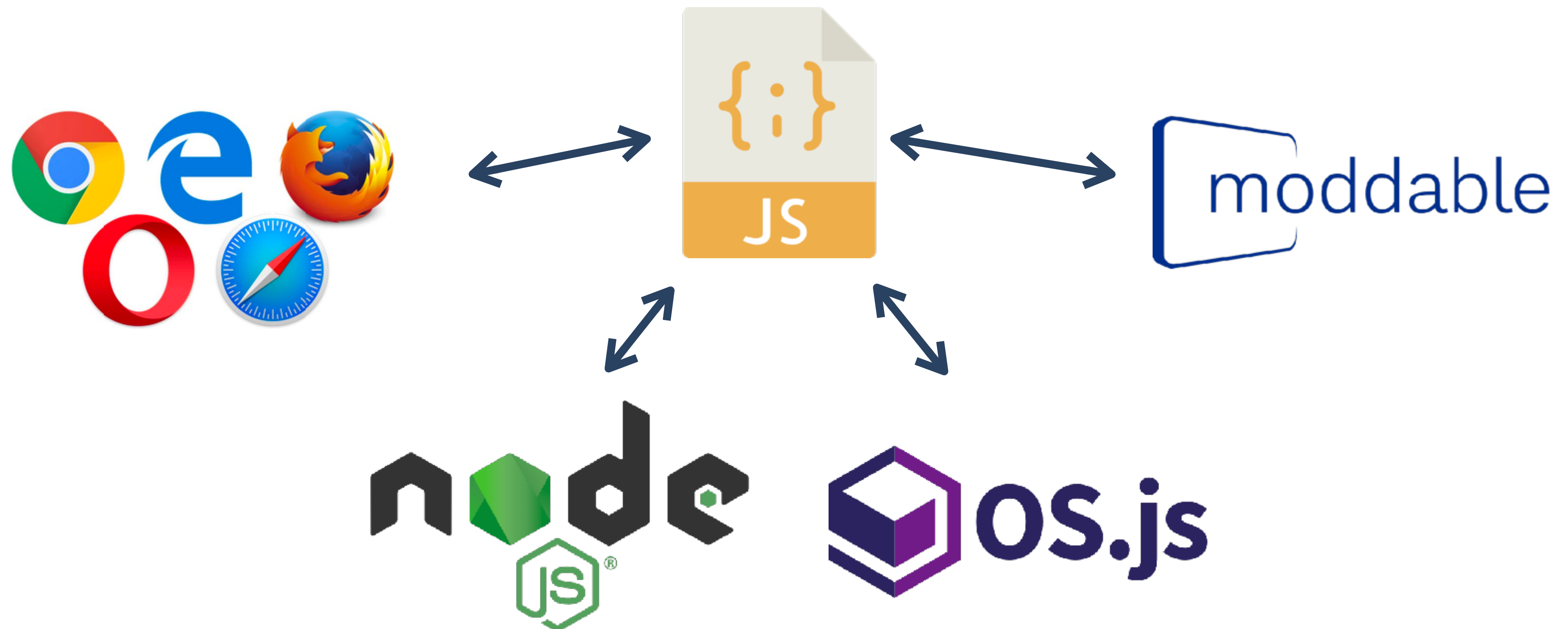
# JSTAR: JavaScript Specification Type Analyzer using Refinement

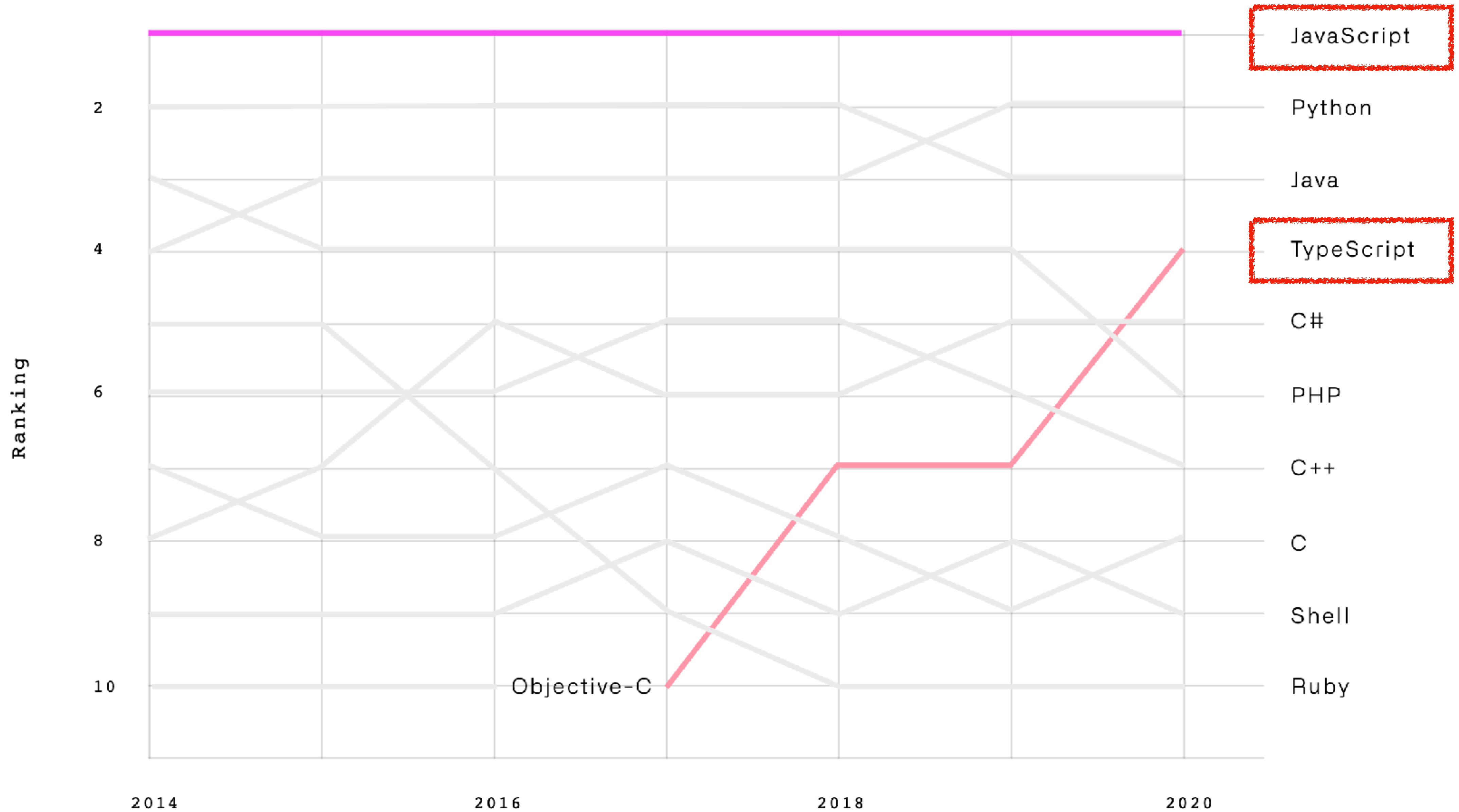
The 36th IEEE/ACM International Conference on Automated Software  
Engineering (ASE'21)

Jihyeok Park, Seungmin An, Wonho Shin,  
Yusung Sim, Sukyoung Ryu

PLRG @ KAIST  
November 17, 2021

# JavaScript is Everywhere





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

# ECMAScript: JavaScript Specification



Semantics

Syntax

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

## 13.2.5.2 Runtime Semantics: Evaluation

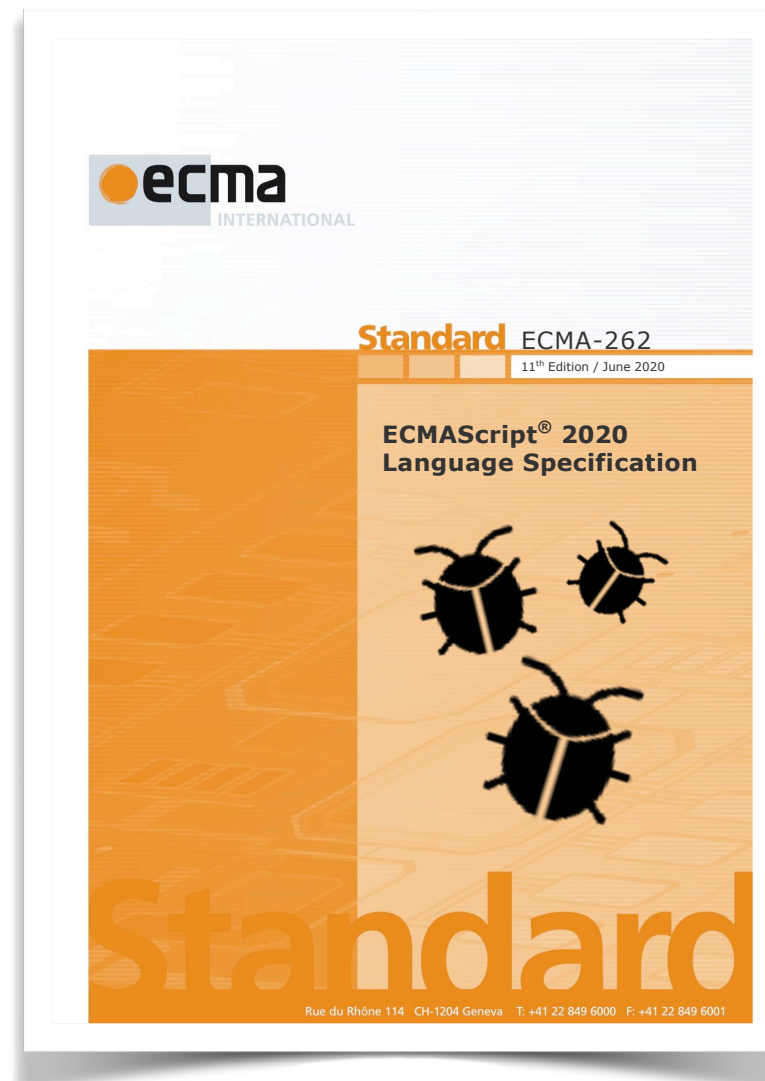
*ArrayLiteral* : [ *ElementList* , *Elision*<sub>opt</sub> ]

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*.

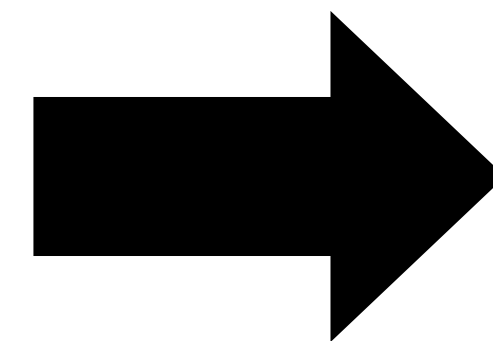
The production of *ArrayLiteral* in ES12

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

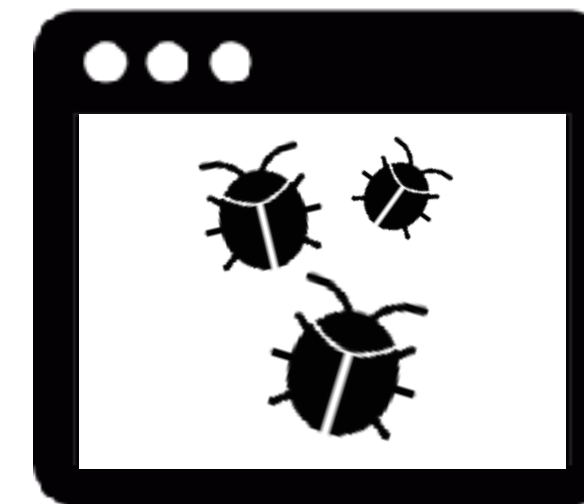
# Correctness of ECMAScript is Important



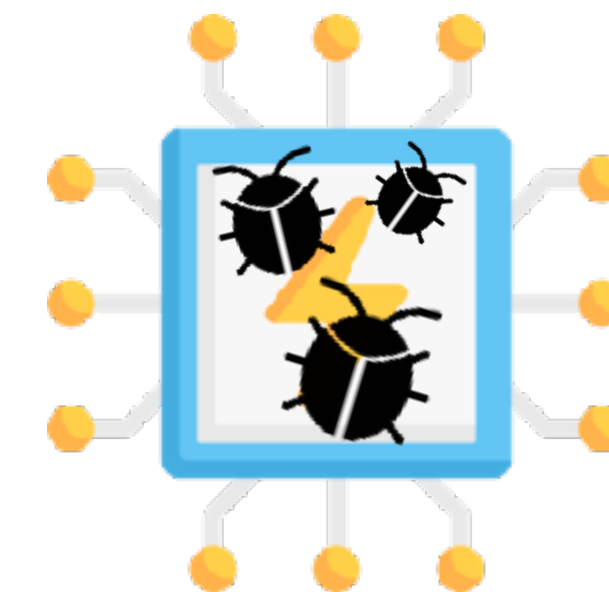
**ECMAScript**



**Web Applications**

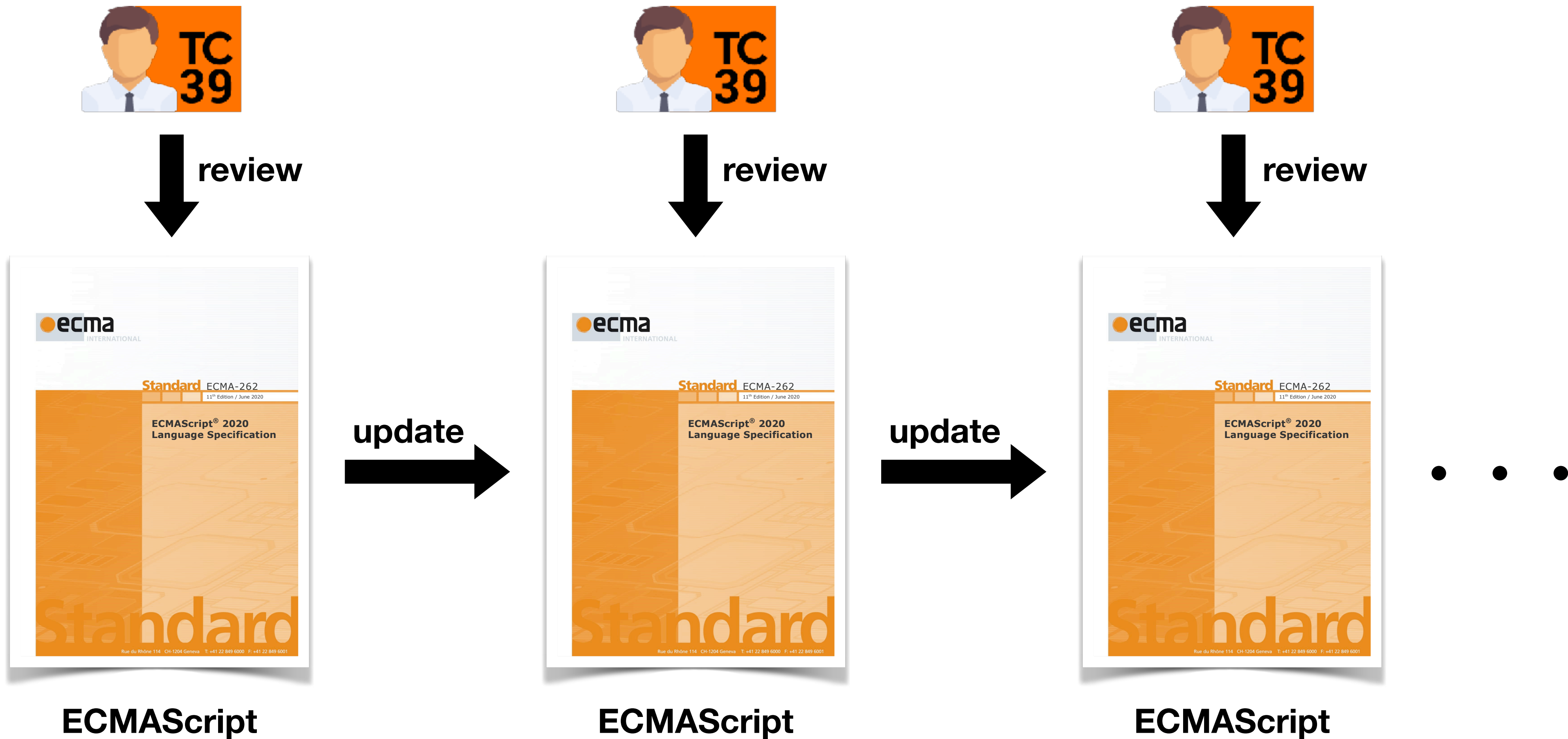


**Server-side Programs**

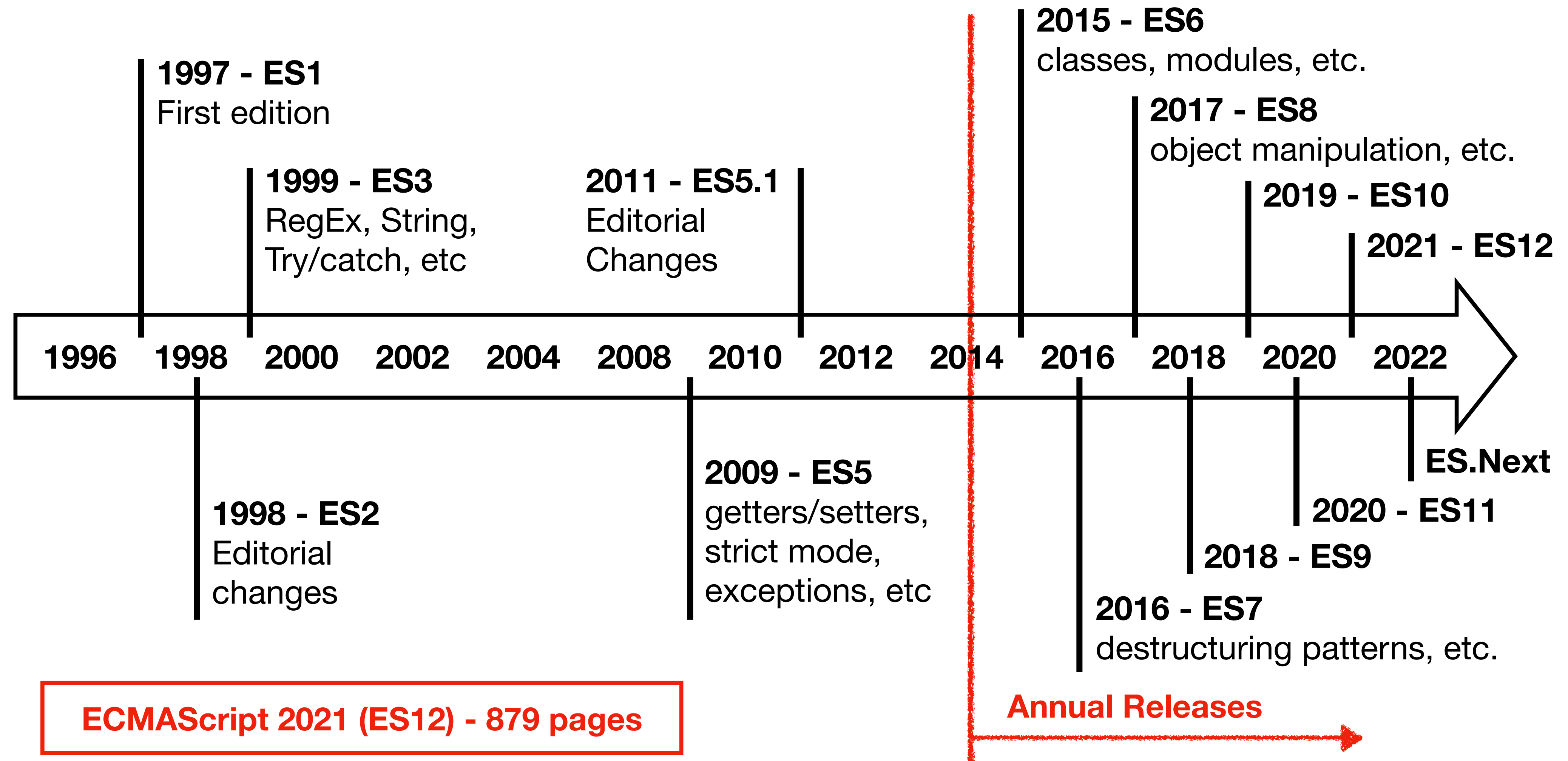


**Embedded Systems**

# Problem: Manual Review of ECMAScript



# Problem: Fast Evolving JavaScript



# Problem: Open Development Process

The image shows a screenshot of the GitHub repository page for `tc39/ecma262`. The repository is public and has 262 issues and 83 pull requests. A search bar at the top right contains the query `is:pr is:closed is:merged`. A dropdown menu is open, showing a search icon, the query, and a button to clear the search. Below the search bar, a box highlights `965 Total` next to a pull request icon. The main content area shows a list of pull requests. The first pull request, by user `ljharb`, is titled `Meta: change master to main i...` and has a clock icon next to the number `2,179`, which is highlighted with a red dashed box. Below the pull request list, there are buttons for `main`, `Go to file`, `Add file`, and `Code`. The `About` section on the right provides a description of the repository: `Status, process, and documents for ECMA-262`, along with a link to `tc39.es/ecma262/` and tags for `javascript` and `ecmascript`.



# Solution: Type Analysis for ECMAScript

**20.3.2.28 Math.round ( $x$ )**  $x$ : (String  $\vee$  Boolean  $\vee$  Number  $\vee$  Object  $\vee$  ...)

1. Let  $n$  be ? `ToNumber`( $x$ ).  $n$ : (Number)  $\wedge$  `ToNumber`( $x$ ): (Number  $\vee$  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.

Type Mismatch for numeric operator `>`

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

...

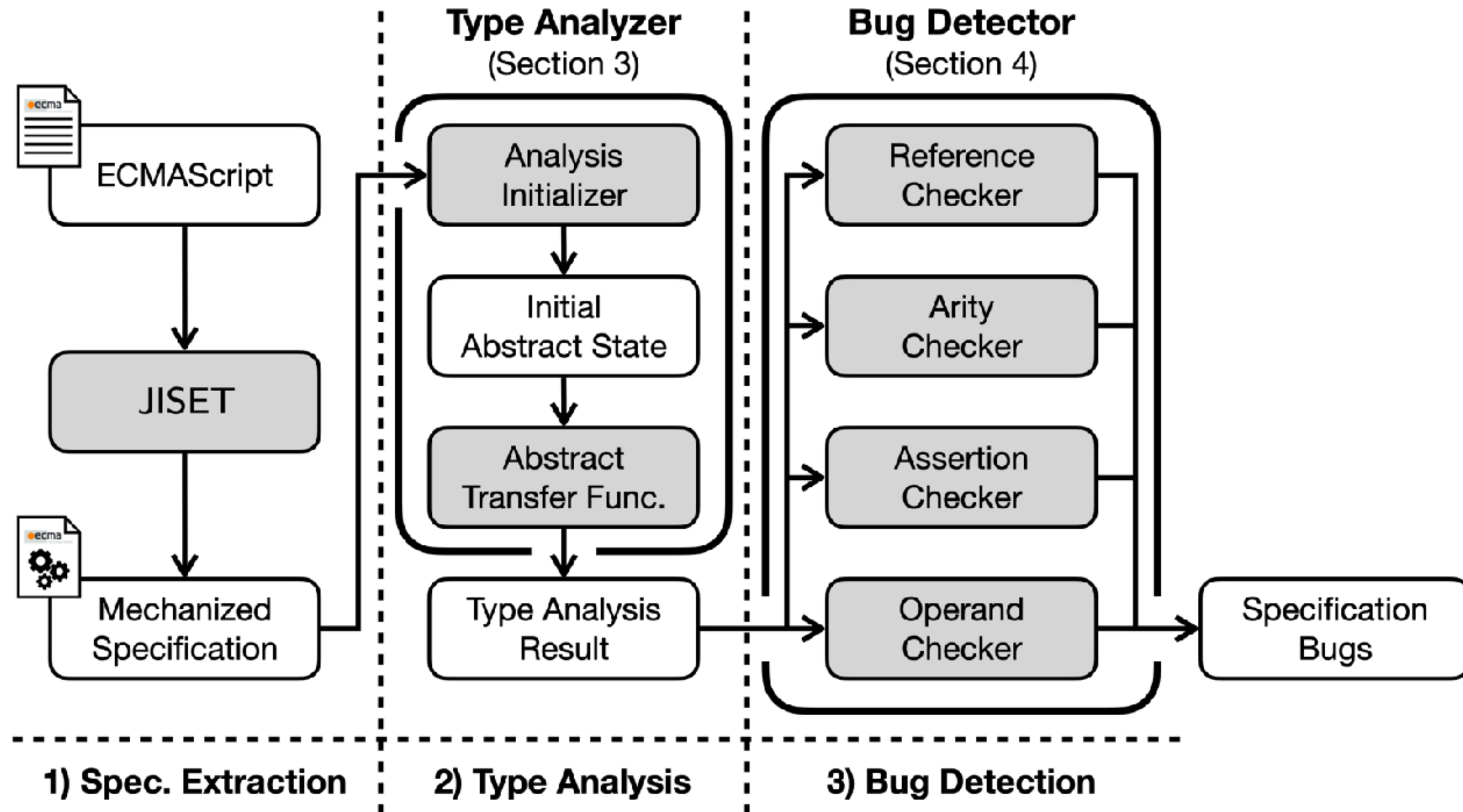
3. If  $n < 0.5$  and  $n > 0$ , return +0.
4. If  $n < 0$  and  $n \geq -0.5$ , return -0.

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

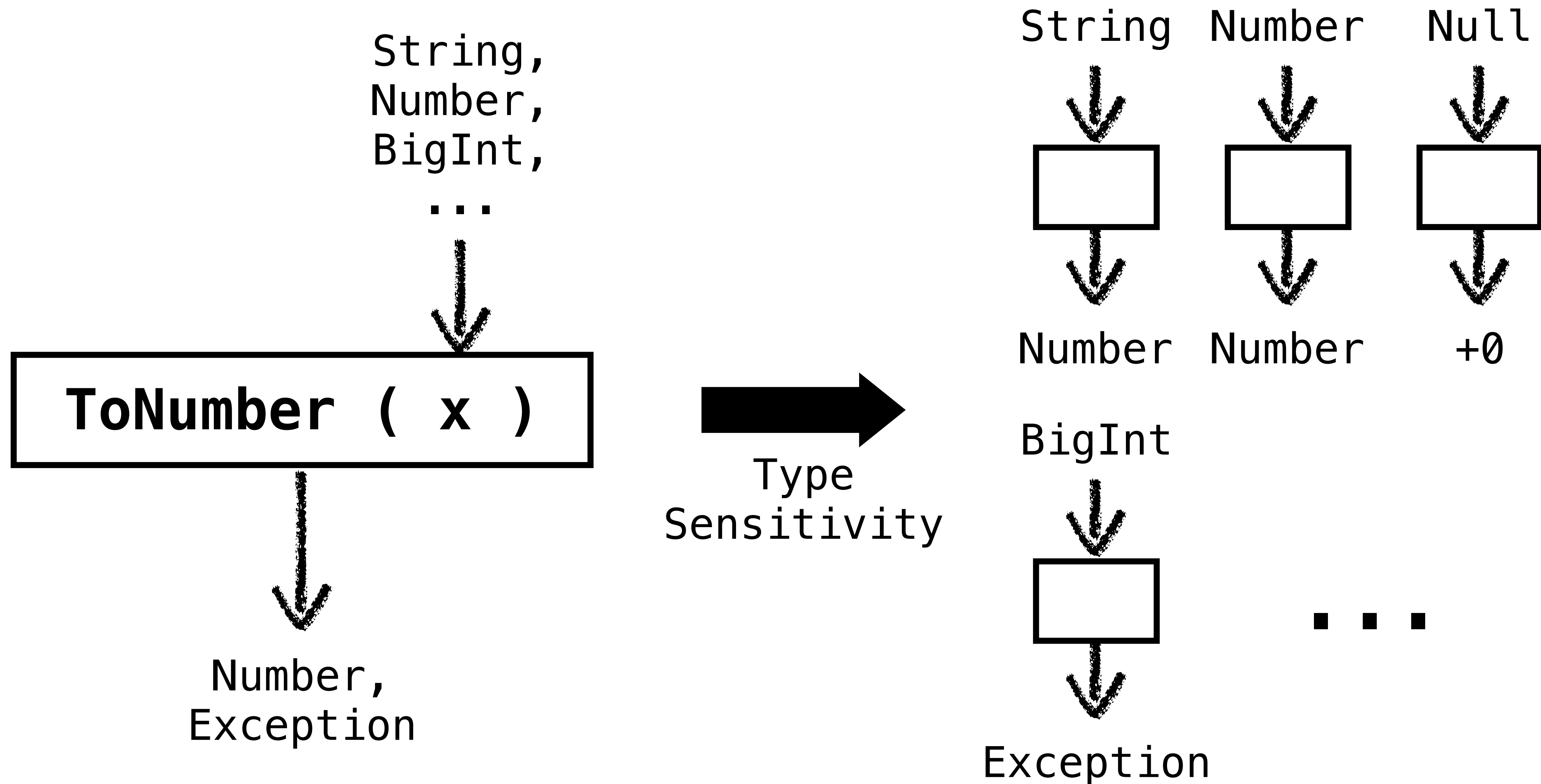
<https://github.com/tc39/ecma262/tree/575149cfd77aebcf3a129e165bd89e14caafc31c>

# Overall Structure of JSTAR

JavaScript Specification Type Analyzer using Refinement



# Precision ↑ - 1) Type Analysis



# Precision $\uparrow$ - 2) Condition-based Refinement

$$\text{refine}(!e, b)(\sigma^\#) = \text{refine}(e, \neg b)(\sigma^\#)$$

$$\text{refine}(e_0 \parallel e_1, b)(\sigma^\#) = \begin{cases} \sigma_0^\# \sqcup \sigma_1^\# & \text{if } b \\ \sigma_0^\# \sqcap \sigma_1^\# & \text{if } \neg b \end{cases}$$

$$\text{refine}(e_0 \ \&\& \ e_1, b)(\sigma^\#) = \begin{cases} \sigma_0^\# \sqcap \sigma_1^\# & \text{if } b \\ \sigma_0^\# \sqcup \sigma_1^\# & \text{if } \neg b \end{cases}$$

$$\text{refine}(x.\text{Type} == c_{\text{normal}}, \#t)(\sigma^\#) = \sigma^\#[x \mapsto \tau_x^\# \sqcap \text{normal}(\mathbb{T})]$$

$$\text{refine}(x.\text{Type} == c_{\text{normal}}, \#f)(\sigma^\#) = \sigma^\#[x \mapsto \tau_x^\# \sqcap \{\text{abrupt}\}]$$

$$\text{refine}(x == e, \#t)(\sigma^\#) = \sigma^\#[x \mapsto \tau_x^\# \sqcap \tau_e^\#]$$

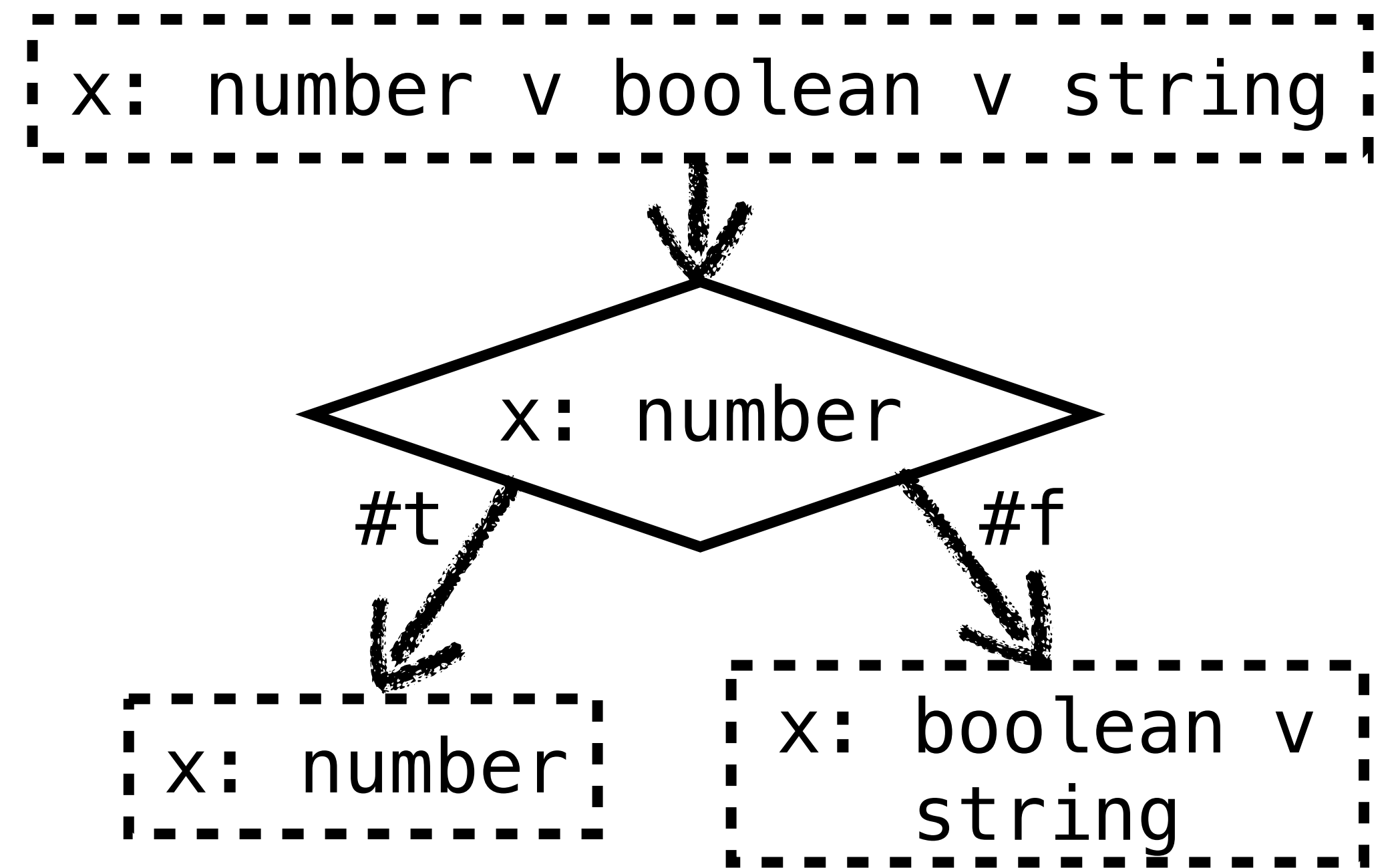
$$\text{refine}(x == e, \#f)(\sigma^\#) = \sigma^\#[x \mapsto \tau_x^\# \setminus \lfloor \tau_e^\# \rfloor]$$

$$\text{refine}(x : \tau, \#t)(\sigma^\#) = \sigma^\#[x \mapsto \tau_x^\# \sqcap \{\tau\}]$$

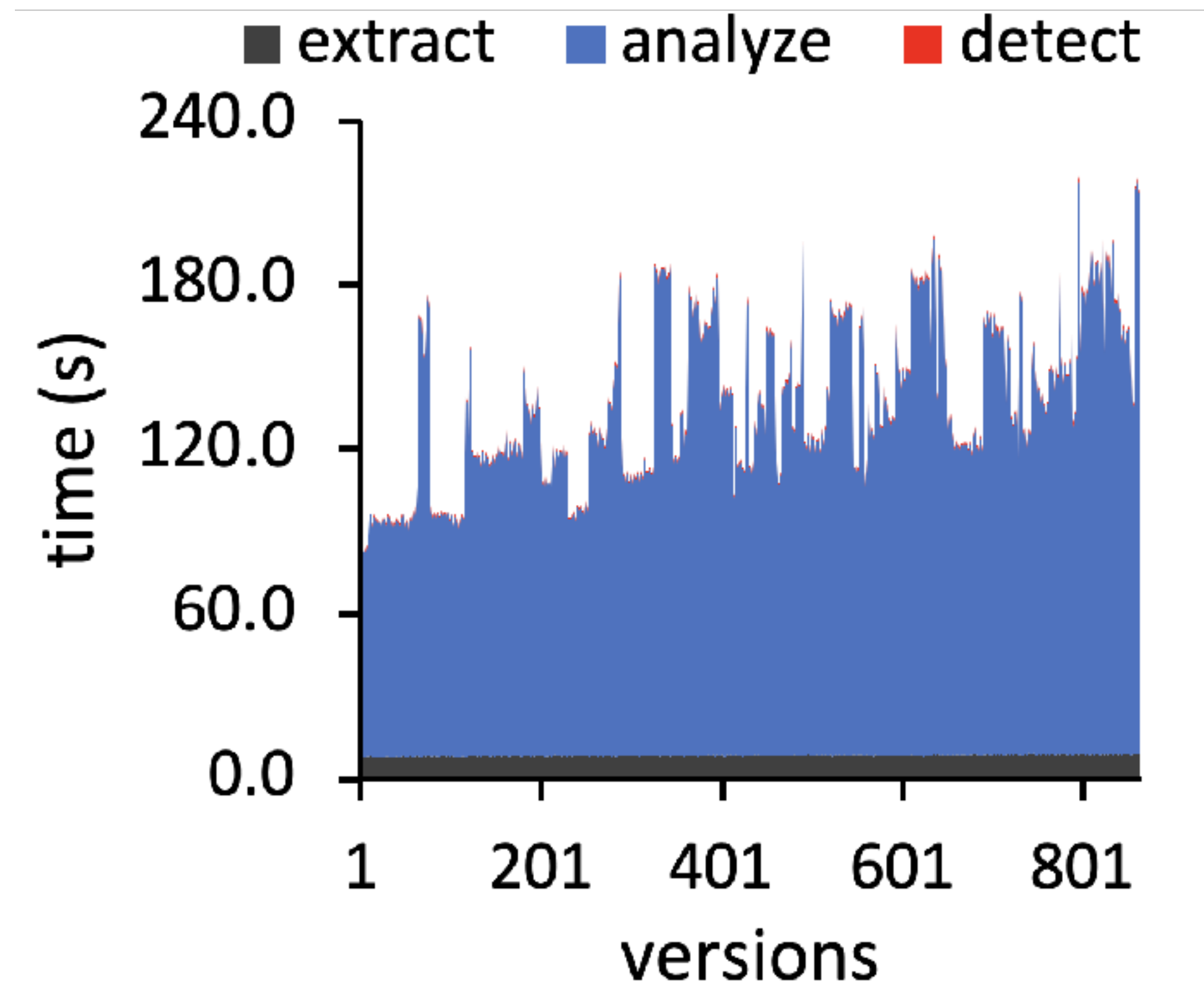
$$\text{refine}(x : \tau, \#f)(\sigma^\#) = \sigma^\#[x \mapsto \tau_x^\# \setminus \{\tau' \mid \tau' <: \tau\}]$$

$$\text{refine}(e, b)(\sigma^\#) = \sigma^\#$$

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  $\tau$ , or returns  $\emptyset$ , otherwise.



# RQ1) Performance

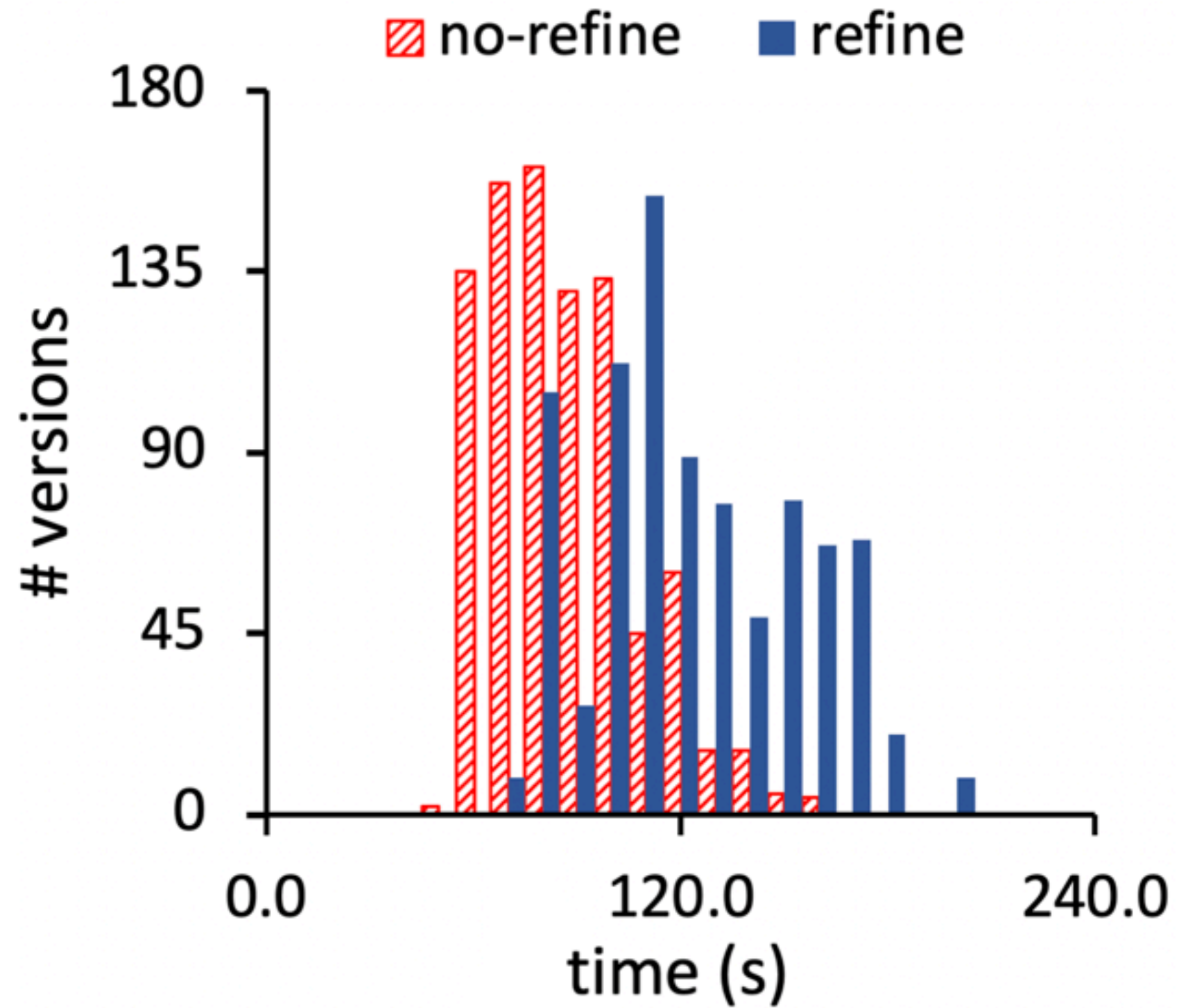


- **864 versions** of ECMAScript (Jan. 1, 2018 to Mar. 9, 2021)
- 4.2GHz Quad-Core Intel Core i7
- 32GB of RAM
- **Average Time : 137.3 s**
  - extract : 8.0 s
  - analyze: 128.5
  - detect: 0.8 s

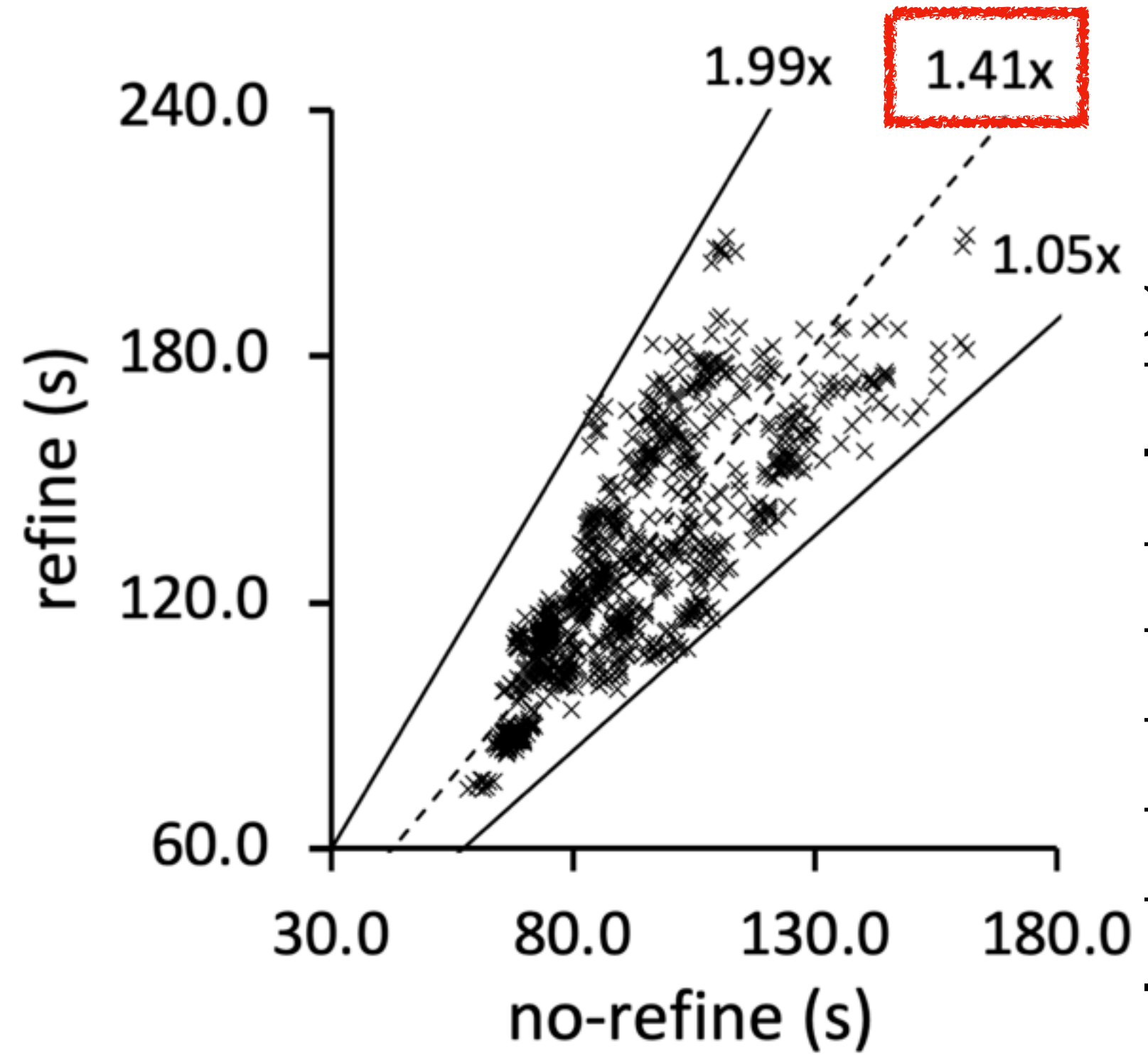
# RQ2) Precision

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
<b>Total</b>		92 / 279 (33.0%)		93 / 157 (59.2%)		+1 / -122 (+26.3%)	

# RQ3) Effectiveness of Refinement



(c) The histogram of time



(d) The ratio of time

## Detected Bugs)

$\Delta$	
+1 / -28	/ -29
	+1 / +1
/	/
/ -25	/ -25
/ -69	/ -59
	/ -10
+1 / -122	(+26.3%)

# RQ4) Detection of New Bugs

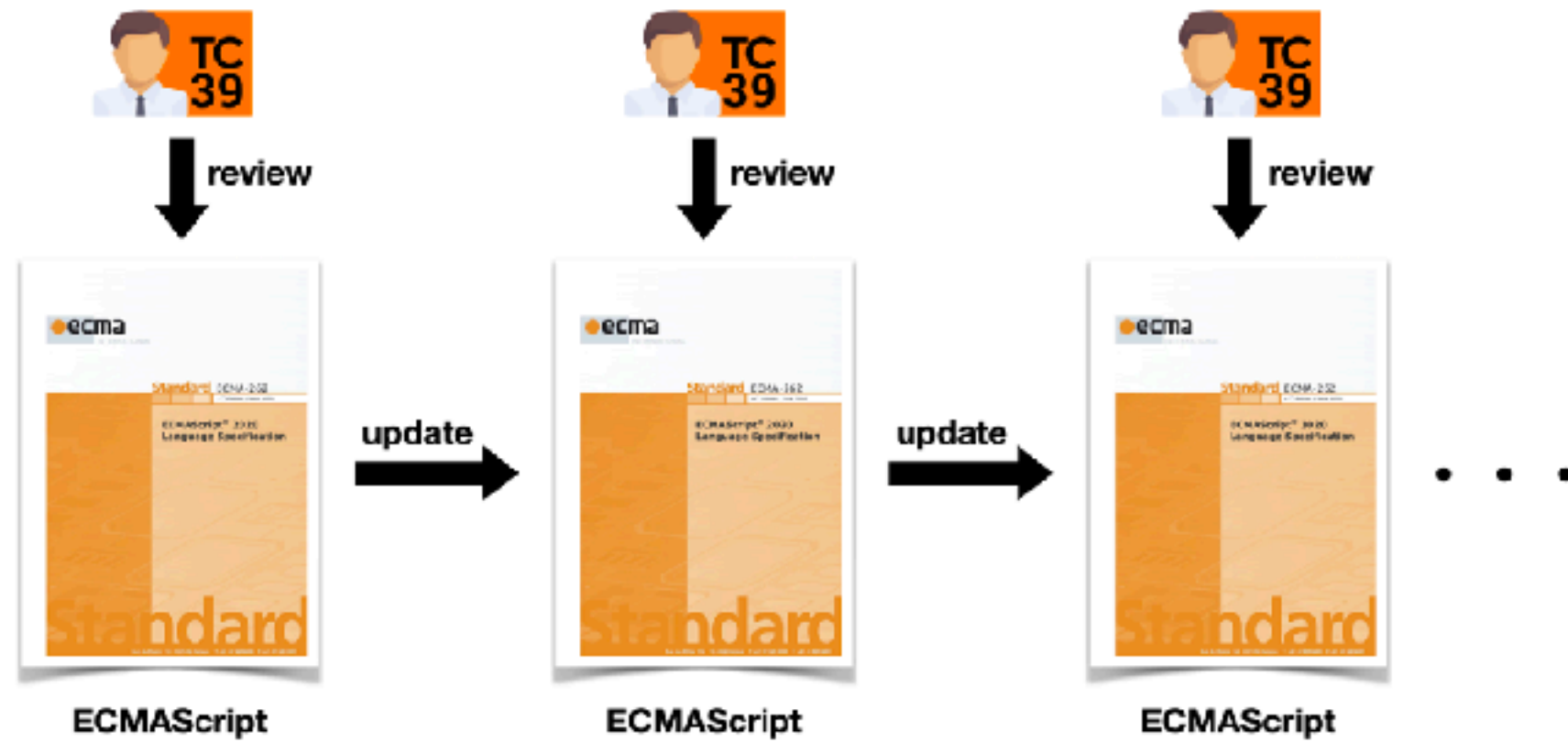
- The Latest Version: **ECMAScript 2021 (ES12)**

14 Bugs  
in Spec.

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



## Problem: Manual Review of ECMAScript



## Solution: Type Analysis for ECMAScript

20.3.2.28 `Math.round(x)`  $x$ : (String  $\vee$  Boolean  $\vee$  Number  $\vee$  Object  $\vee$  ...)

1. Let  $n$  be `?ToNumber(x)`.  $n$ : (Number)  $\wedge$  `ToNumber(x)`: (Number  $\vee$  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`.

Type Mismatch for numeric operator `>`

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

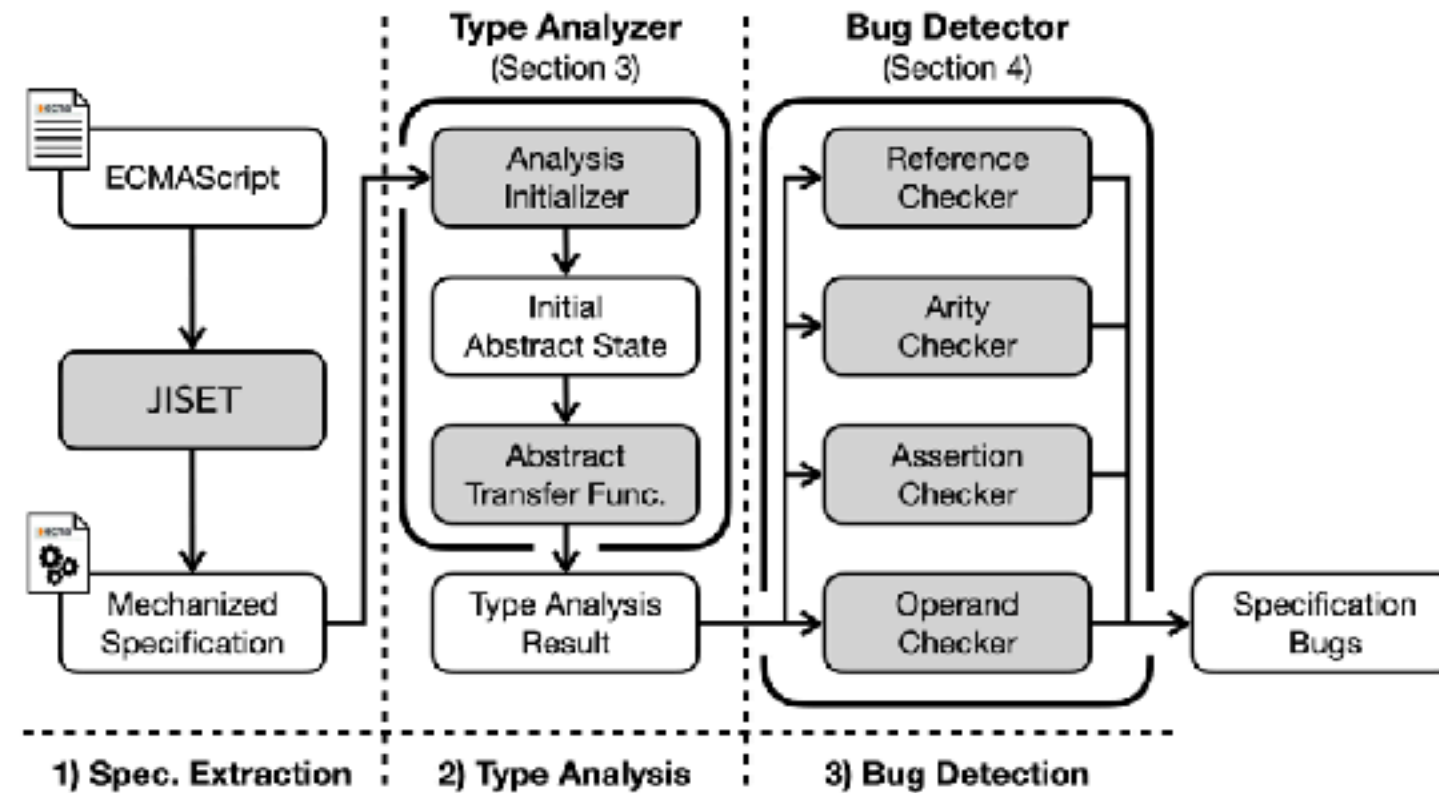
3. If  $n < 0.5$  and  $n > 0$ , return `+0`.
4. If  $n < 0$  and  $n \geq -0.5$ , return `-0`.

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

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

## Overall Structure of JSTAR

JavaScript Specification Type Analyzer using Refinement



## RQ4) Detection of New Bugs

- The Latest Version: **ECMAScript 2021 (ES12)**

14 Bugs in Spec.

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