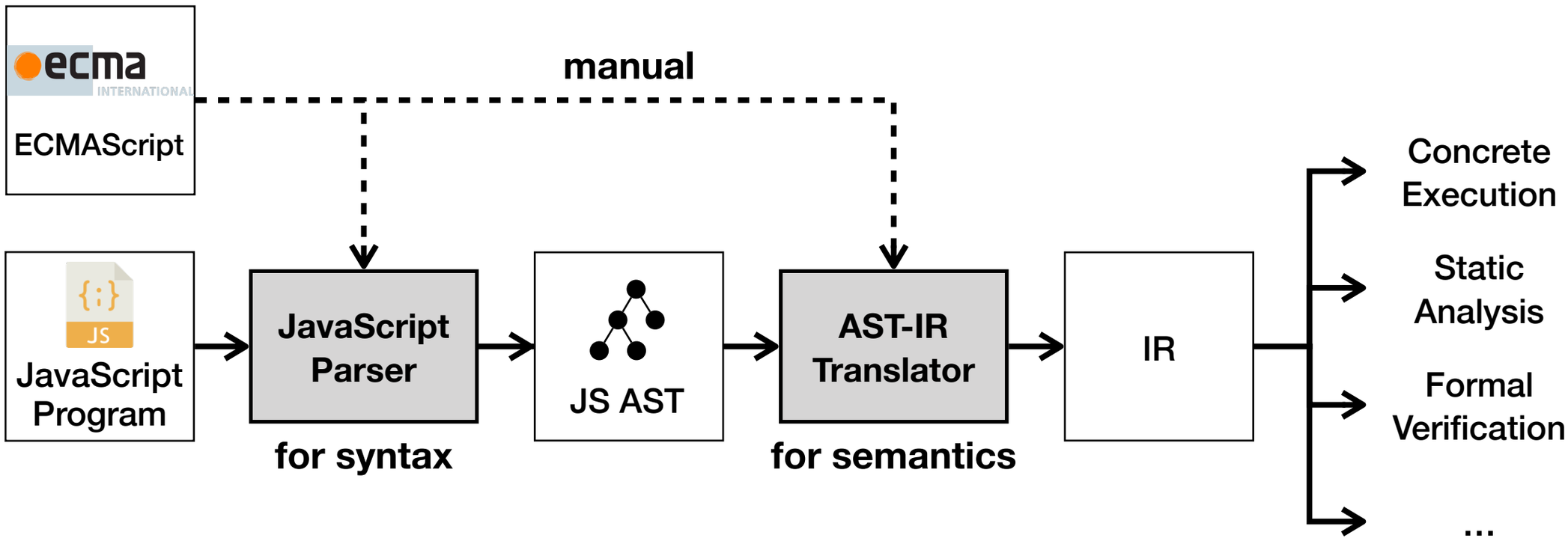
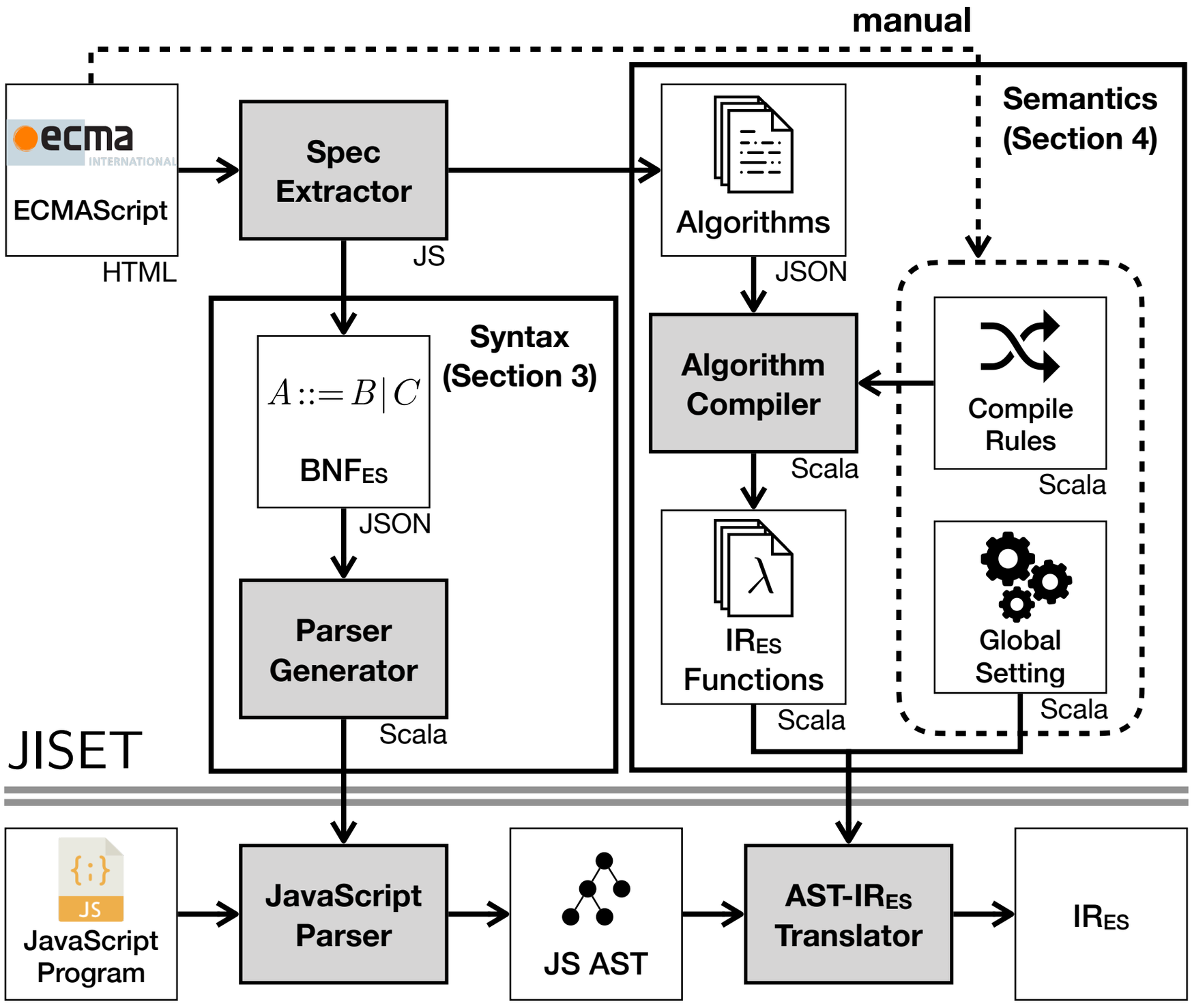

ecma
INTERNATIONAL
ECMAScript
manual
JS
JavaScript
Program
JavaScript
Parser
for syntax
JS AST
AST-IR
Translator
for semantics
IR
Concrete
Execution
Static
Analysis
Formal
Verification
...


manual
ECMAScript
HTML
Spec Extractor
JS
Algorithms
JSON
Semantics (Section 4)
Syntax (Section 3)
$A ::= B \mid C$
BNF$_{ES}$
JSON
Parser Generator
Scala
Algorithm Compiler
Scala
Compile Rules
Scala
IR$_{ES}$ Functions
Scala
Global Setting
Scala
JISET
JavaScript Program
JavaScript Parser
JS AST
AST-IR$_{ES}$ Translator
IR$_{ES}$

ArrayLiteral`[Yield, Await]` :

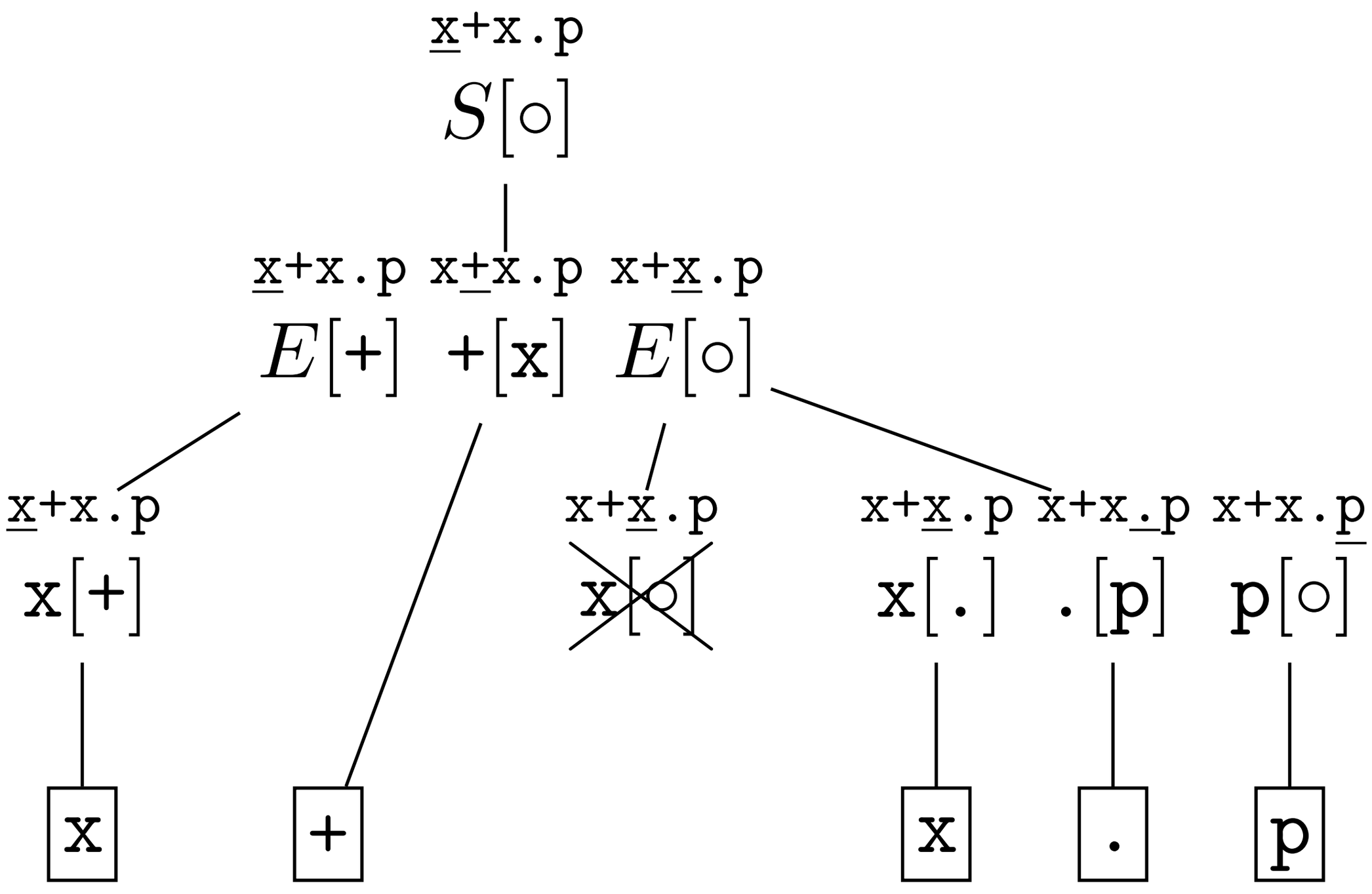
`[` *Elision*`opt` `]`

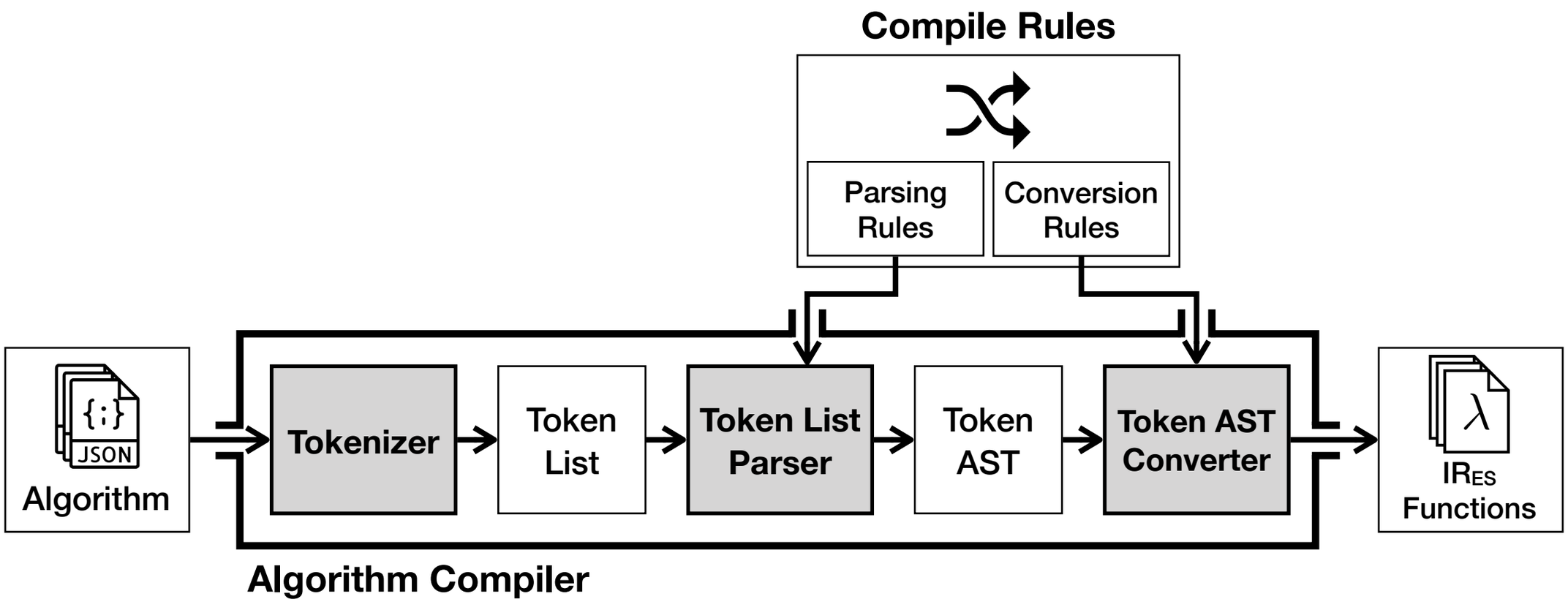
`[` *ElementList*`[?Yield, ?Await]` `]`

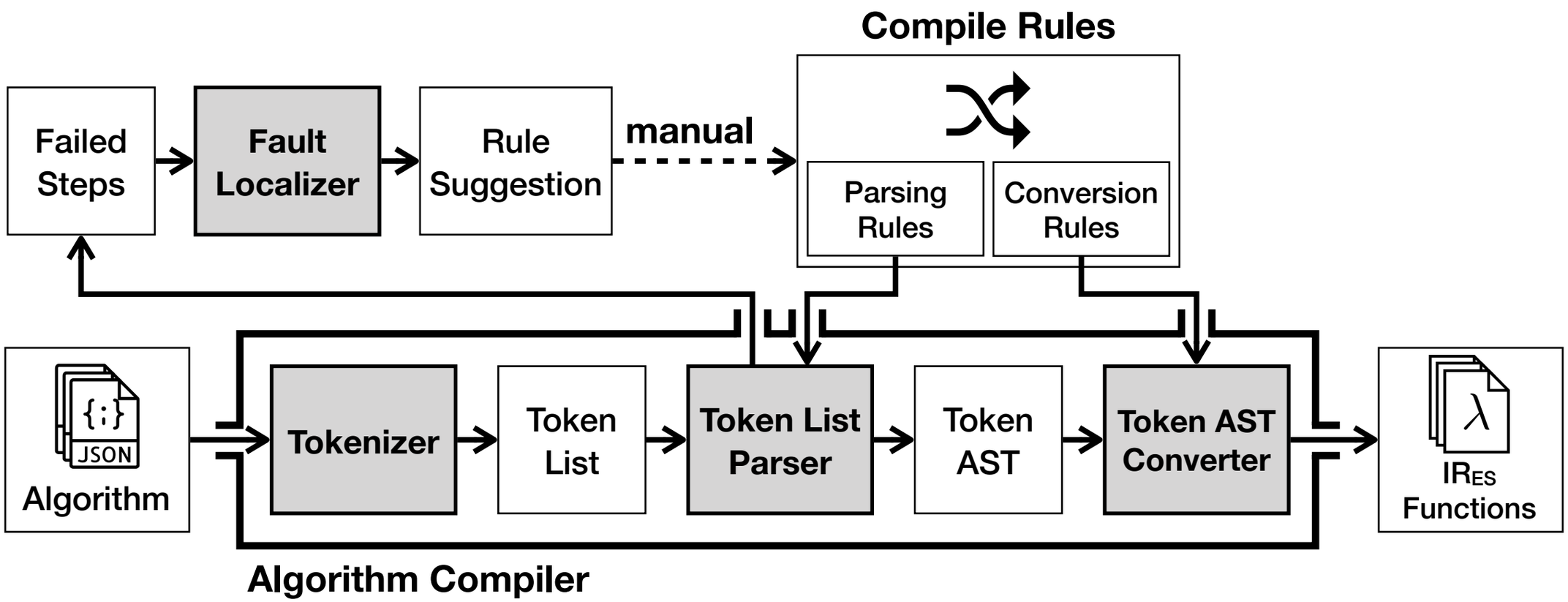
`[` *ElementList*`[?Yield, ?Await]` `,` *Elision*`opt` `]`

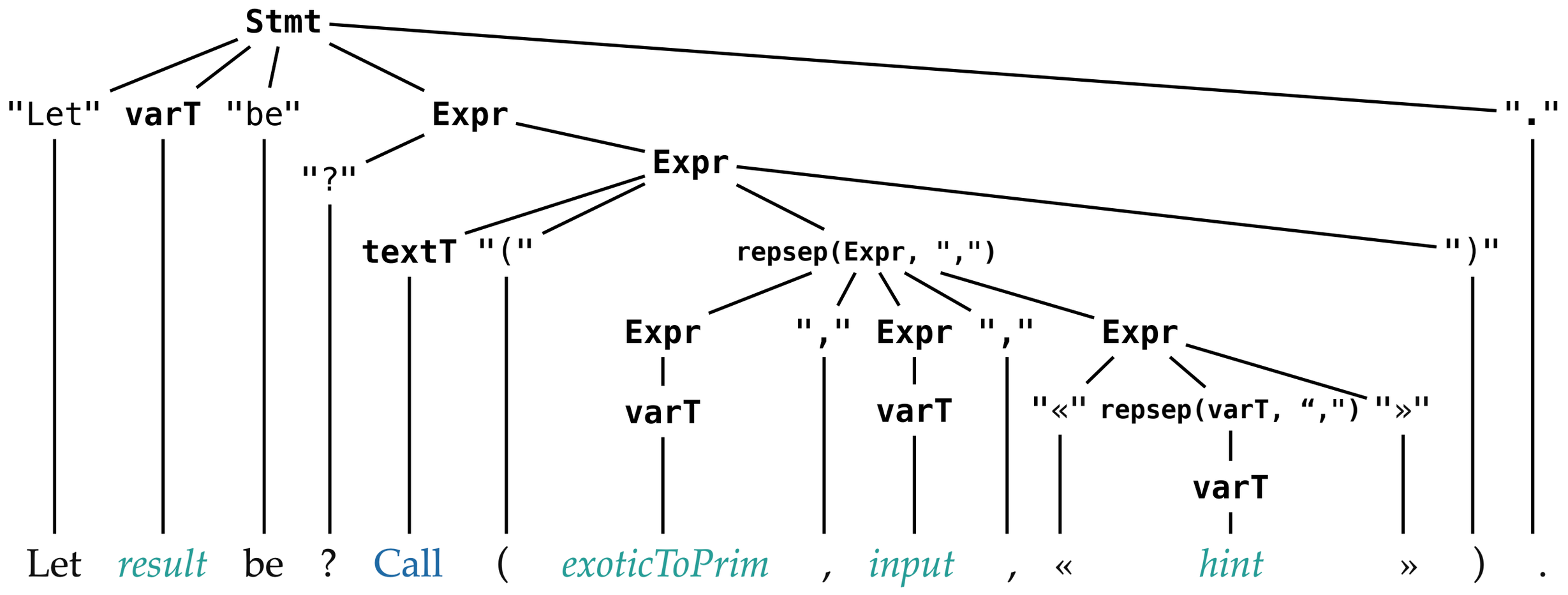
ArrayLiteral : `[` *ElementList* `,` *Elision*$_{\texttt{opt}}$ `]`

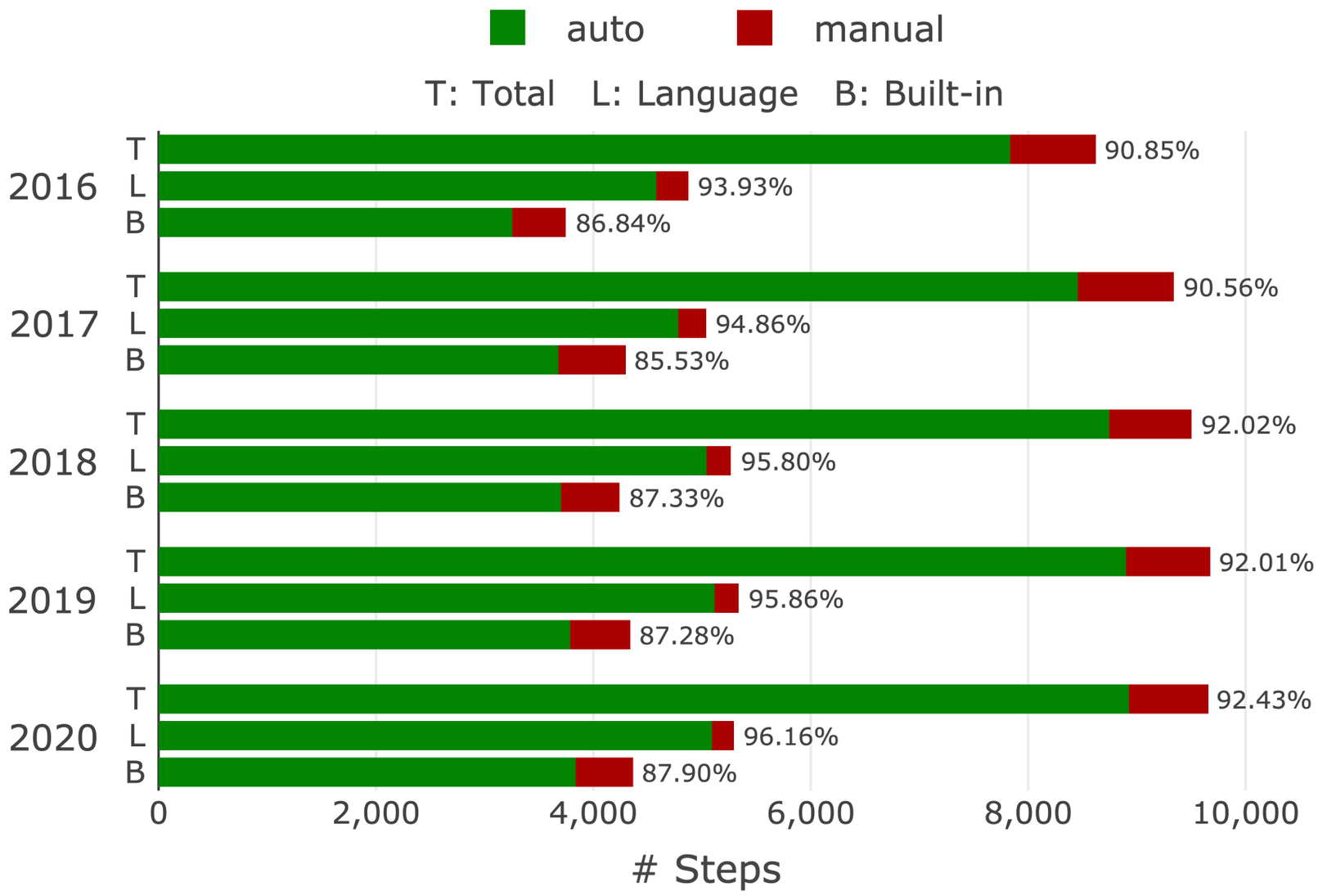
1. Let *array* be ! ArrayCreate(0).
2. Let *len* be the result of performing ArrayAccumulation for *ElementList* with arguments *array* and 0.
3. ReturnIfAbrupt(*len*).
4. Let *padding* be the ElisionWidth of *Elision*; if *Elision* is not present, use the numeric value zero.
5. Perform Set(*array*, `"length"`, ToUint32(*padding* + *len*), **false**).
6. NOTE: The above Set cannot fail because of the nature of the object returned by ArrayCreate.
7. Return *array*.

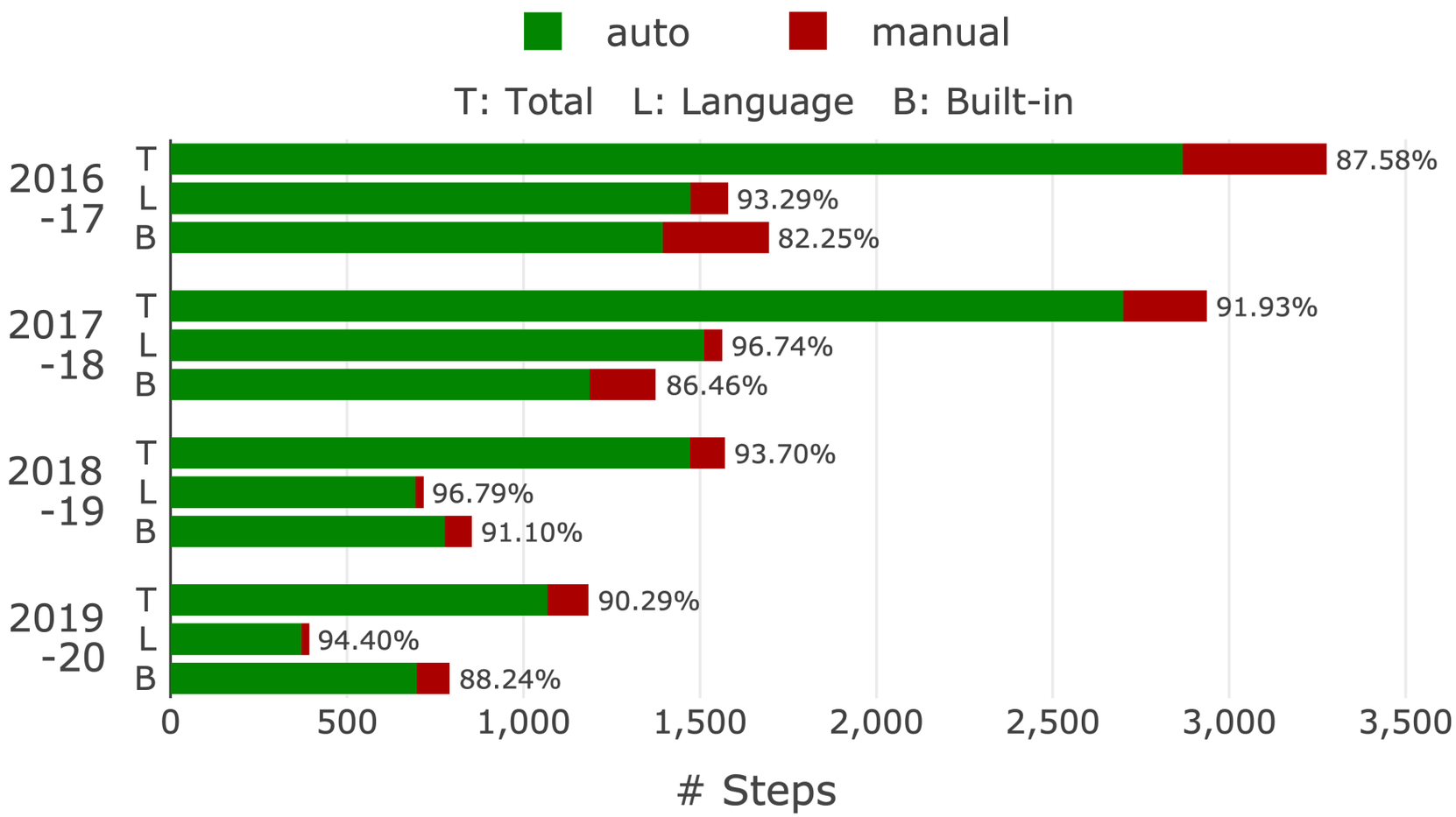

x+x.p
S[○]
x+x.p x+x.p x+x.p
E[+] +[x] E[○]
x+x.p x+x.p x+x.p x+x.p x+x.p x+x.p
x[+] x[○] x[.] .[p] p[○]
x + x . p

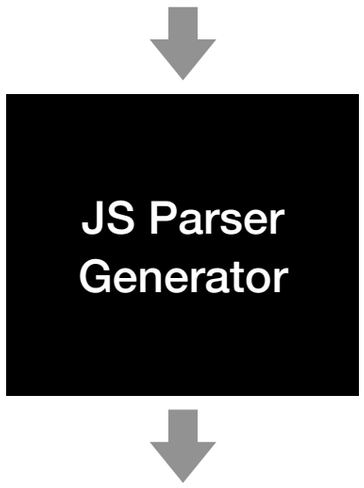

Compile Rules
Parsing Rules
Conversion Rules
Algorithm
Tokenizer
Token List
Token List Parser
Token AST
Token AST Converter
IRES Functions
Algorithm Compiler


Compile Rules
Failed Steps
Fault Localizer
Rule Suggestion
manual
Parsing Rules
Conversion Rules
JSON
Algorithm
Tokenizer
Token List
Token List Parser
Token AST
Token AST Converter
IRES Functions
Algorithm Compiler

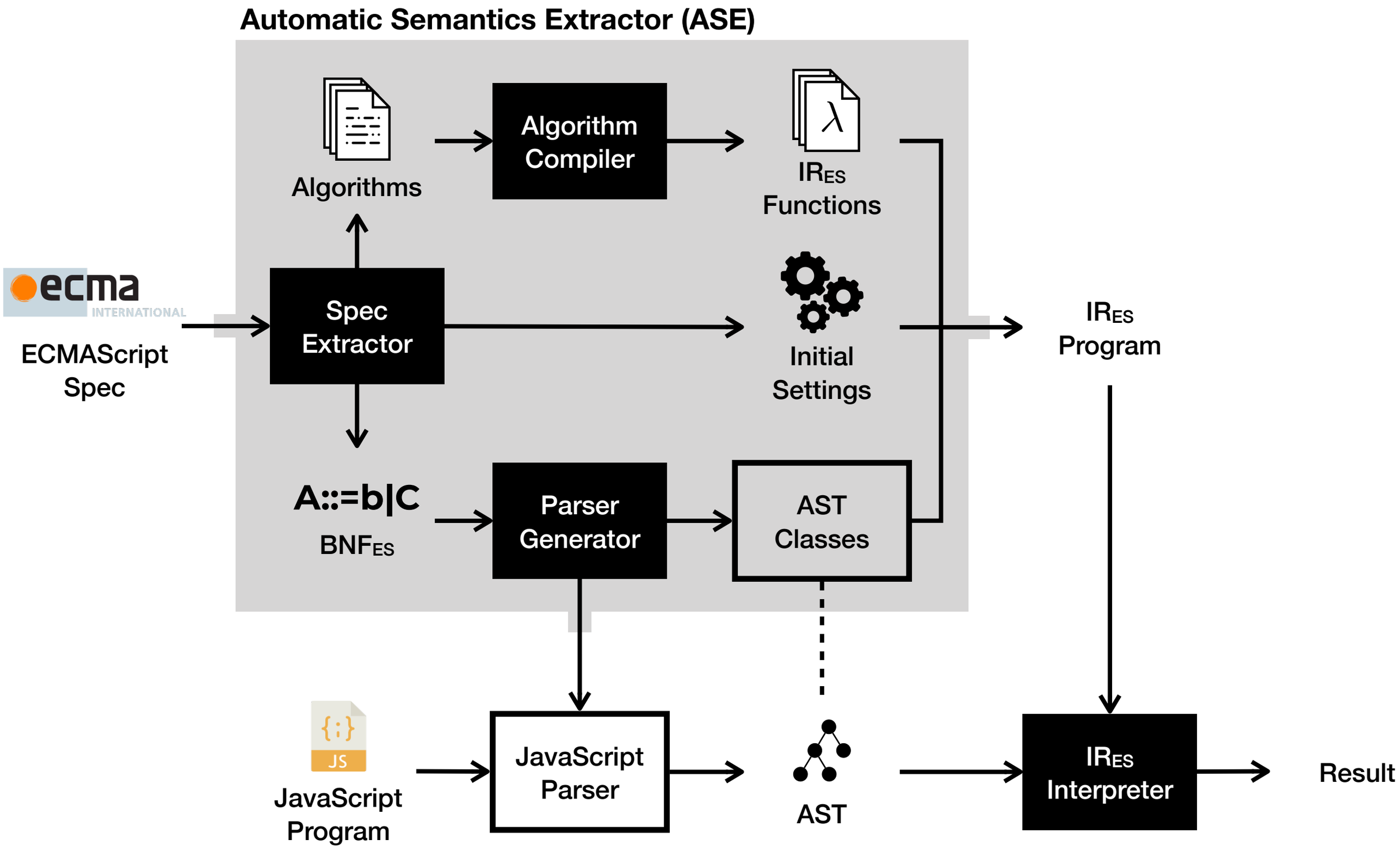

Stmt
"Let"
varT
"be"
Expr
"."
"?"
Expr
textT
"("
repsep(Expr, ",")
")"
Expr
","
Expr
","
Expr
varT
varT
"«"
repsep(varT, “,”)
"»"
varT
Let
result
be
?
Call
(
exoticToPrim
,
input
,
«
hint
»
)
.


auto
manual
T: Total L: Language B: Built-in
2016
2017
2018
2019
2020
T
L
B
90.85%
93.93%
86.84%
90.56%
94.86%
85.53%
92.02%
95.80%
87.33%
92.01%
95.86%
87.28%
92.43%
96.16%
87.90%
0
2,000
4,000
6,000
8,000
10,000
Steps


auto
manual
T: Total L: Language B: Built-in
2016 -17
T
L
B
87.58%
93.29%
82.25%
2017 -18
T
L
B
91.93%
96.74%
86.46%
2018 -19
T
L
B
93.70%
96.79%
91.10%
2019 -20
T
L
B
90.29%
94.40%
88.24%
0
500
1,000
1,500
2,000
2,500
3,000
3,500
Steps


JS Parser
Generator


Algorithm
Compiler


Automatic Semantics Extractor (ASE)
Algorithms
Algorithm Compiler
IRES Functions
ECMAScript Spec
Spec Extractor
Initial Settings
IRES Program
A::=b|C
BNFES
Parser Generator
AST Classes
JavaScript Program
JavaScript Parser
AST
IRES Interpreter
Result

ECMAScript Spec

Spec Extractor

manual

A::=b|C

BNF$_{ES}$

Algorithms

Section 3

JS Parser Generator

Section 4

Algorithm Compiler

Compile Rules

Initial Setting

IR$_{ES}$ Functions

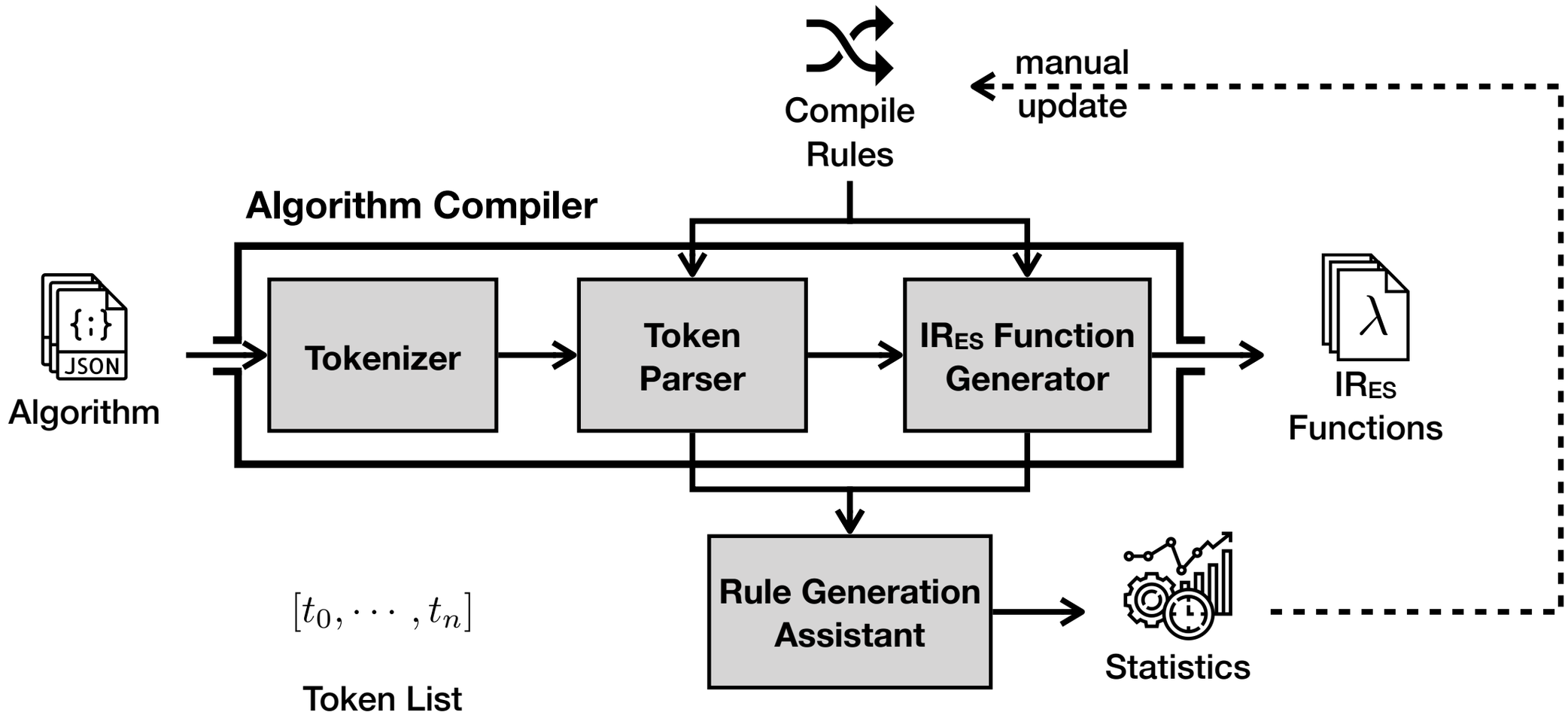
JavaScript Program

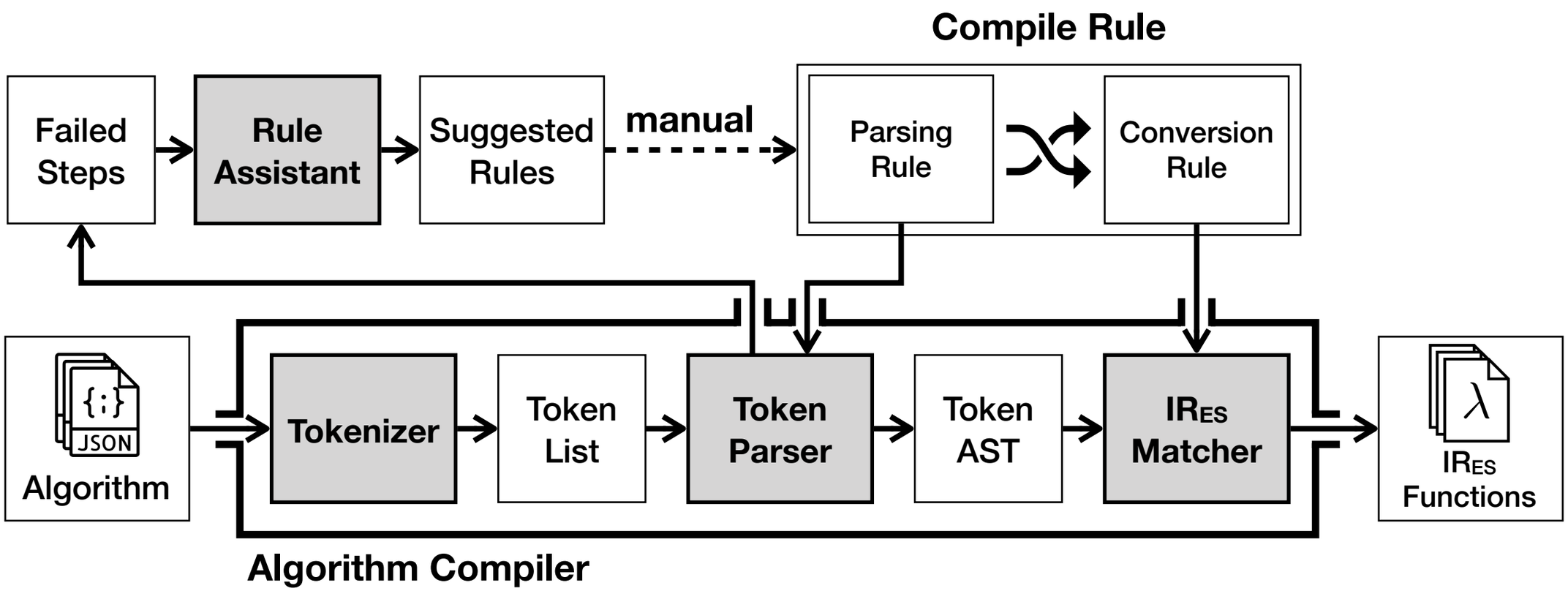
JavaScript Parser

JS AST

IR$_{ES}$ Interpreter

Evaluation Result


Compile
Rules
manual
update
Algorithm Compiler
JSON
Algorithm
Tokenizer
Token
Parser
IRES Function
Generator
IRES
Functions
$[t_0, \cdots, t_n]$
Token List
Rule Generation
Assistant
Statistics


Compile Rule
Failed Steps
Rule Assistant
Suggested Rules
manual
Parsing Rule
Conversion Rule
Algorithm
Tokenizer
Token List
Token Parser
Token AST
IRES Matcher
IRES Functions
Algorithm Compiler