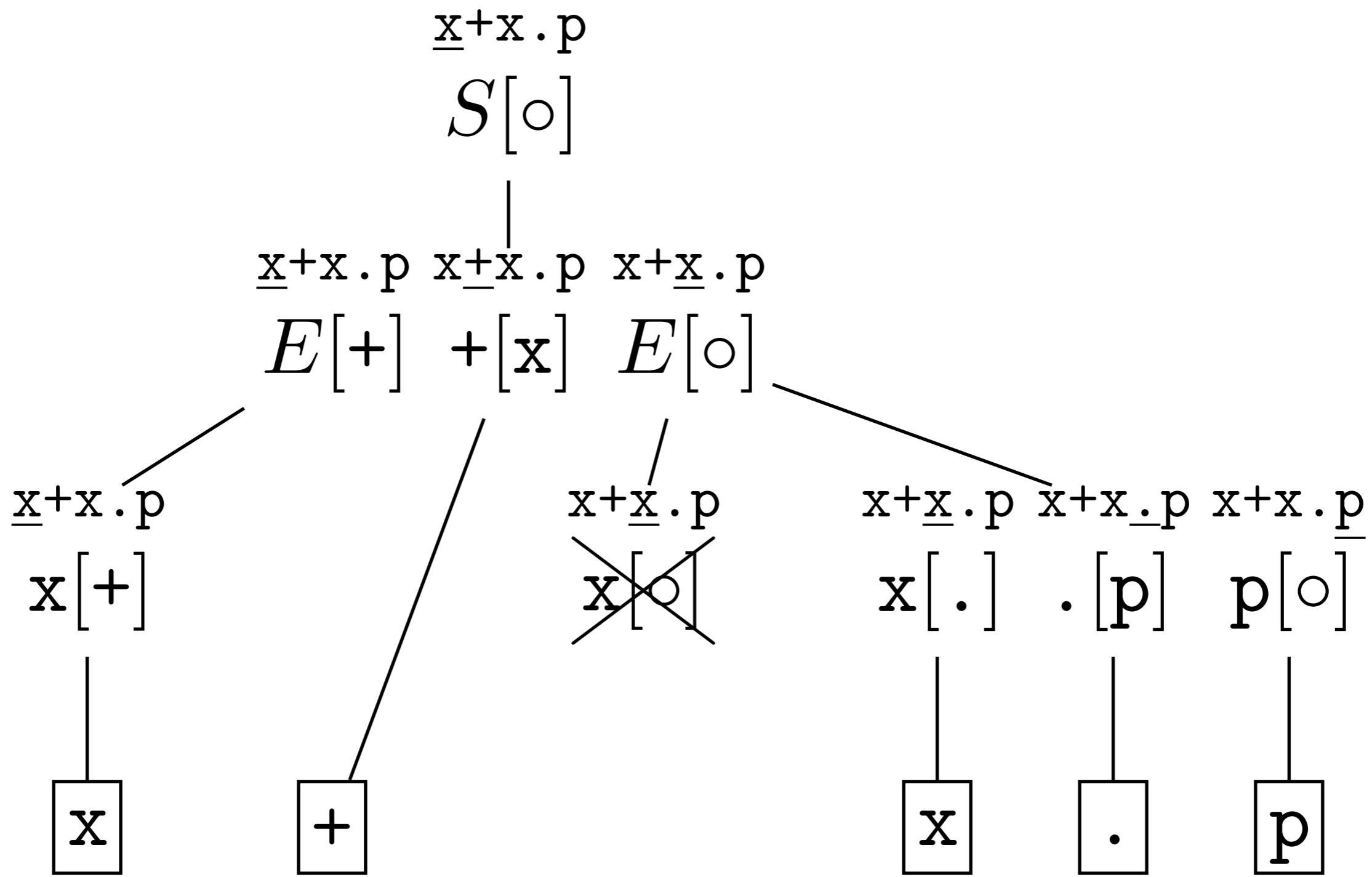
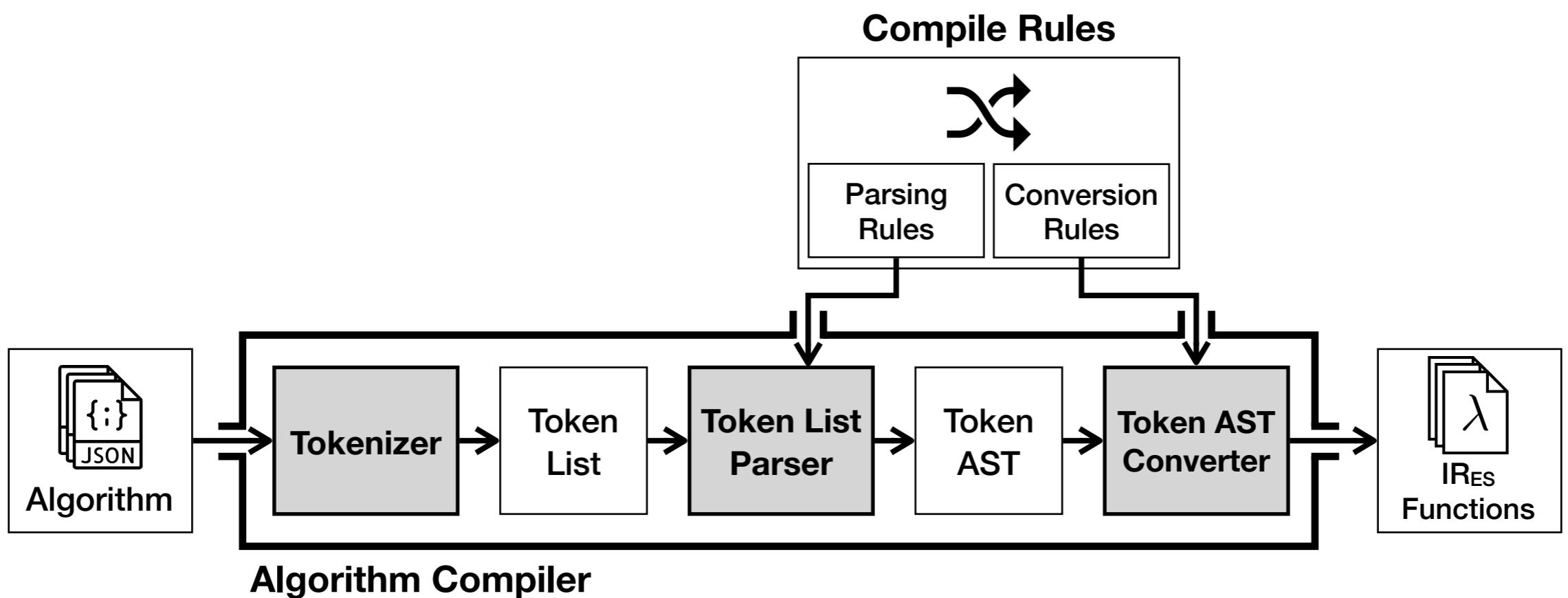


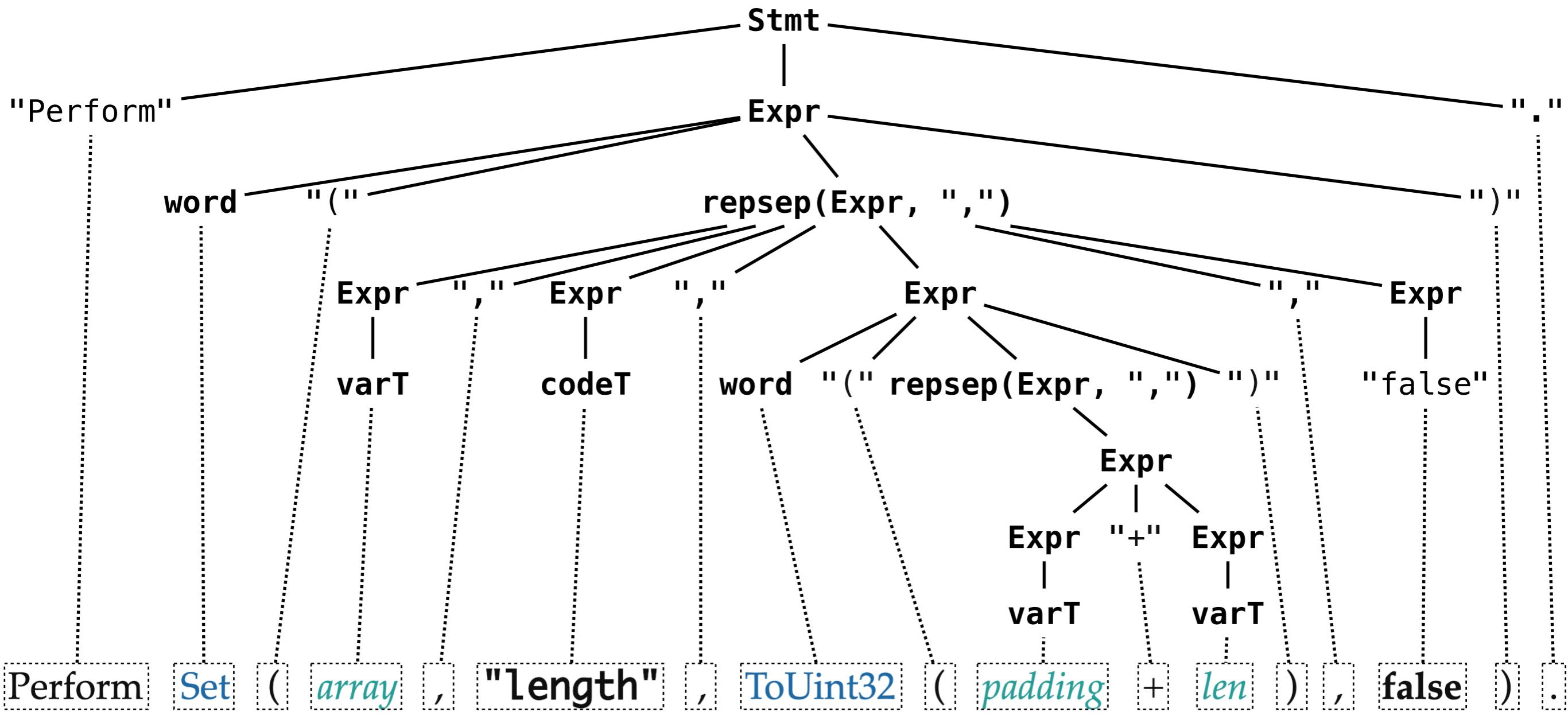
ArrayLiteral [*Yield*, *Await*] :
[*Elision*_{opt}]
[*ElementList* [?*Yield*, ?*Await*]]
[*ElementList* [?*Yield*, ?*Await*] , *Elision*_{opt}]

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

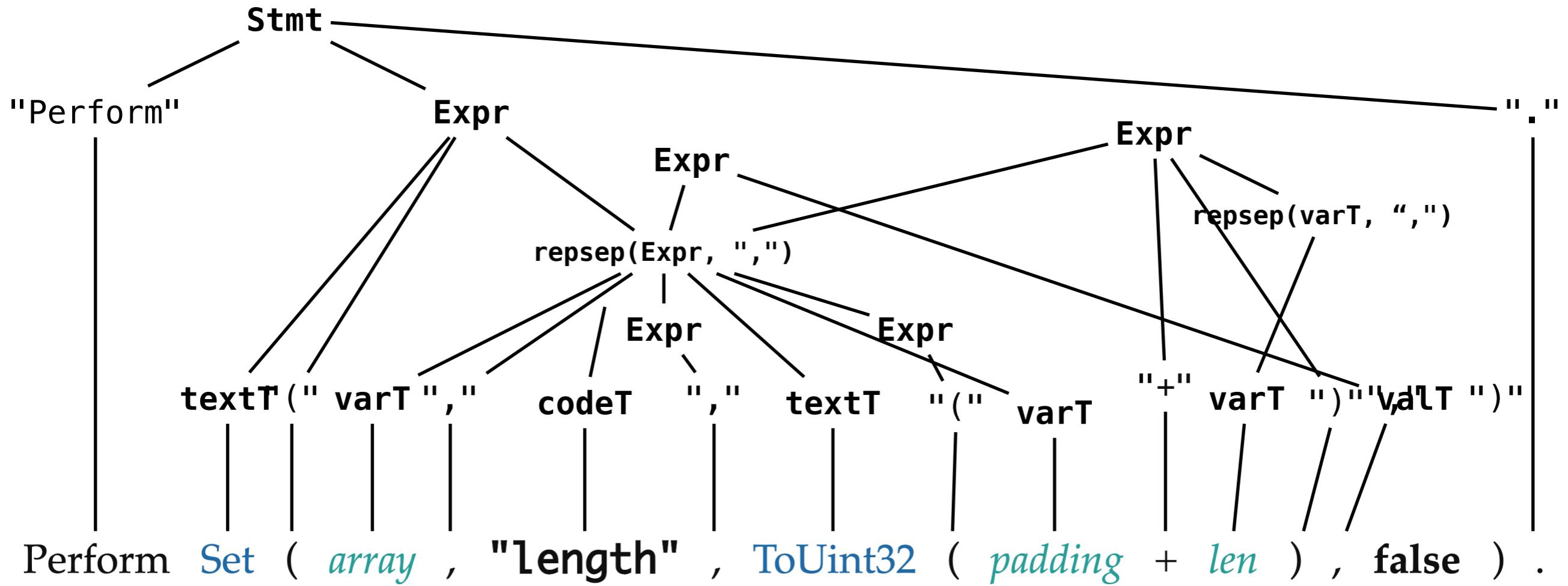
1. Let *array* be ! **ArrayCreate**(0).
2. Let *len* be the result of performing **ArrayAccumulation** for *ElementList* with arguments *array* and 0.
3. **ReturnIfAbrupt**(*len*).
4. Let *padding* be the **ElisionWidth** of *Elision*; if *Elision* is not present, use the numeric value zero.
5. Perform **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*.



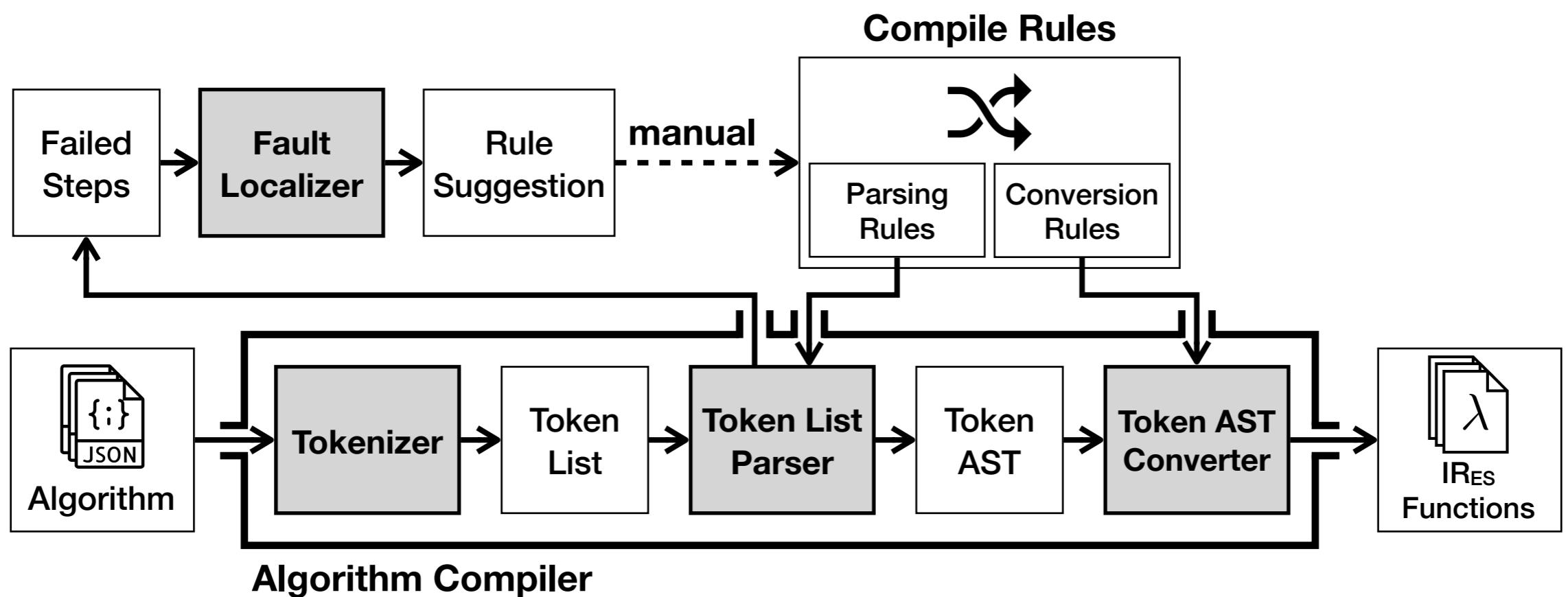


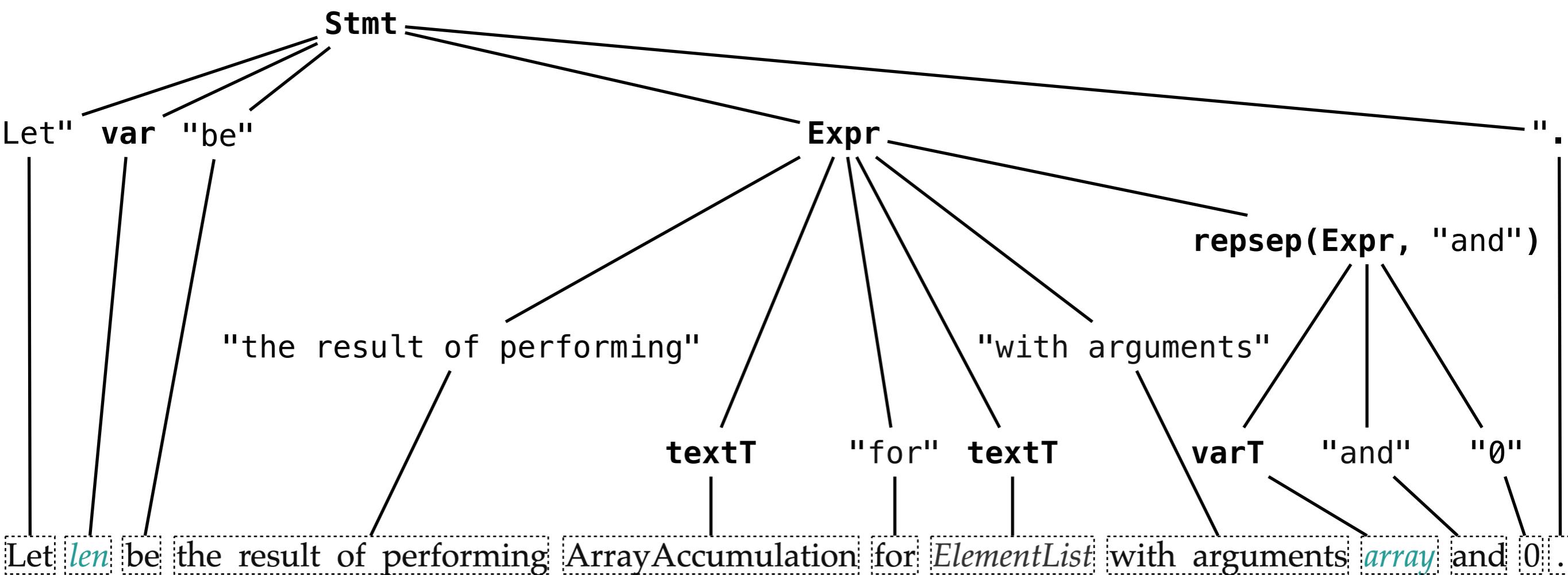


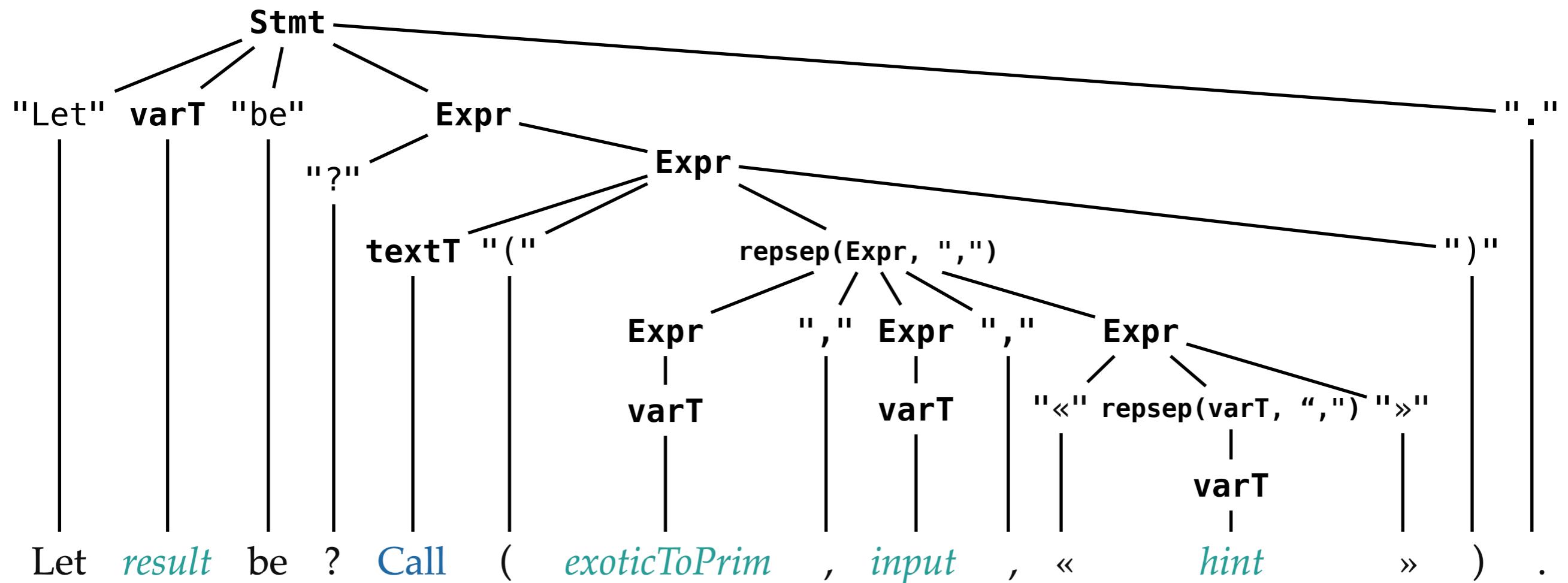
Perform Set(*array*, "length", ToUInt32(*padding* + *len*), false).

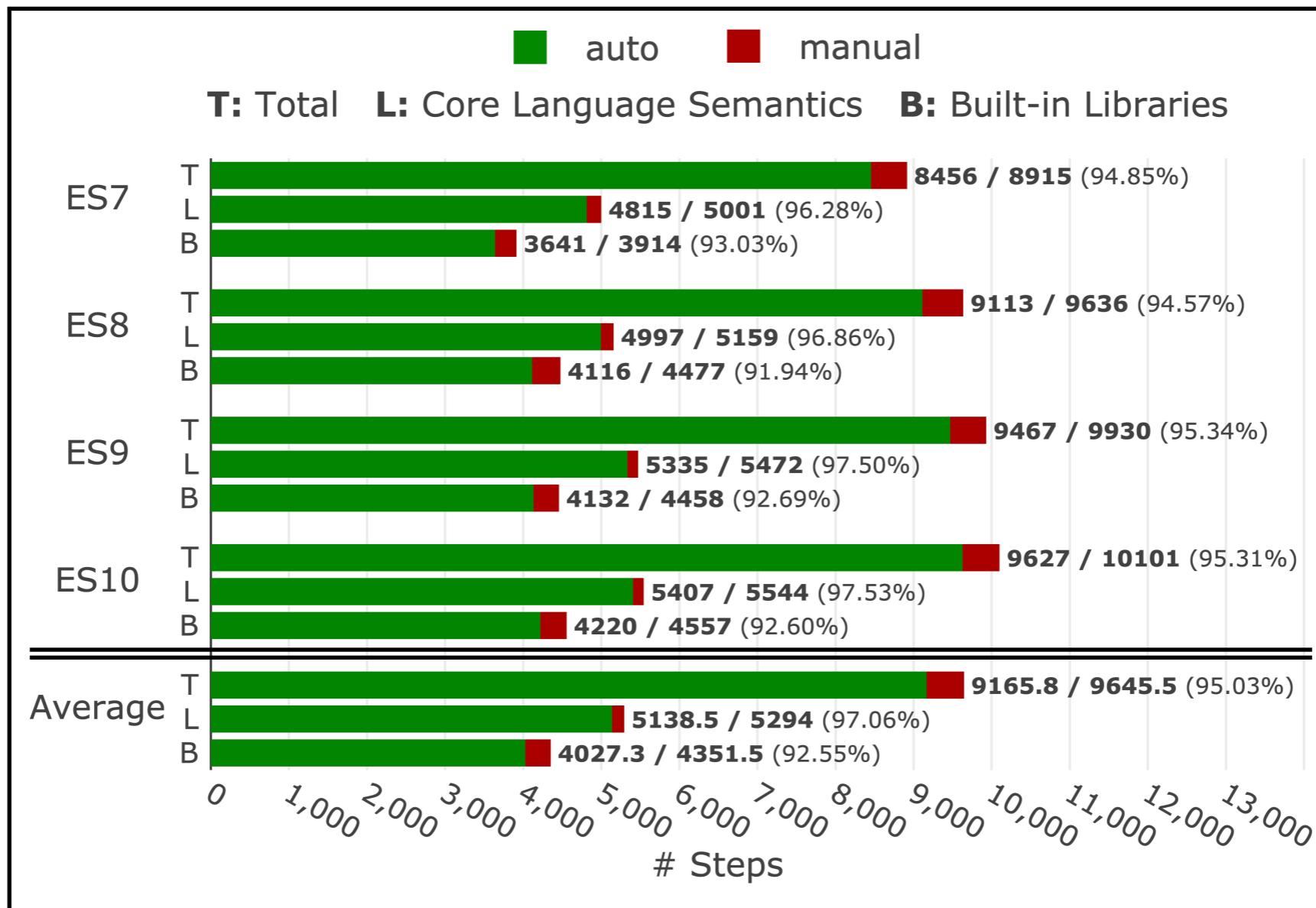


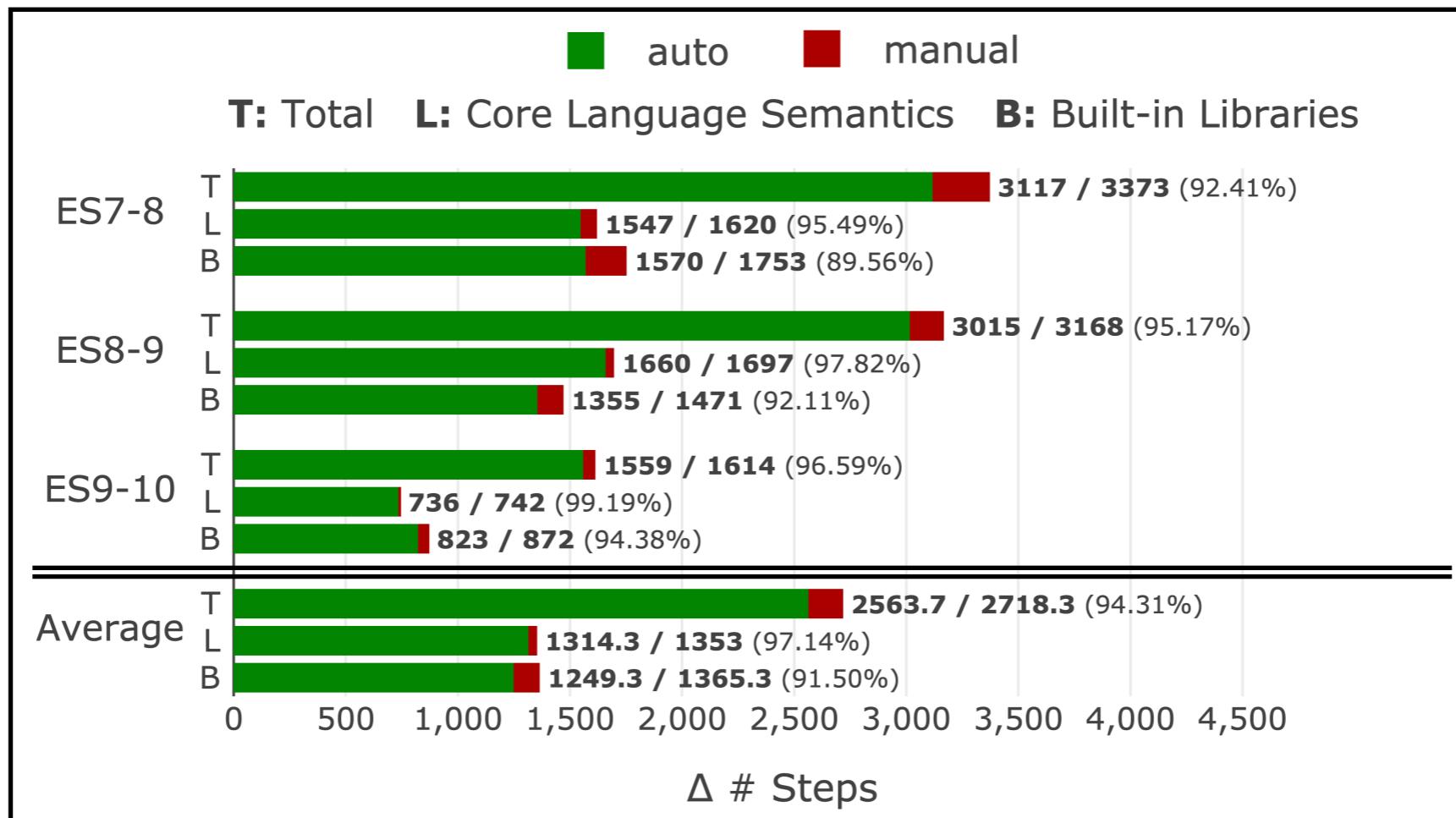
Let *result* be ? Call (*exoticToPrim* , *input* , «





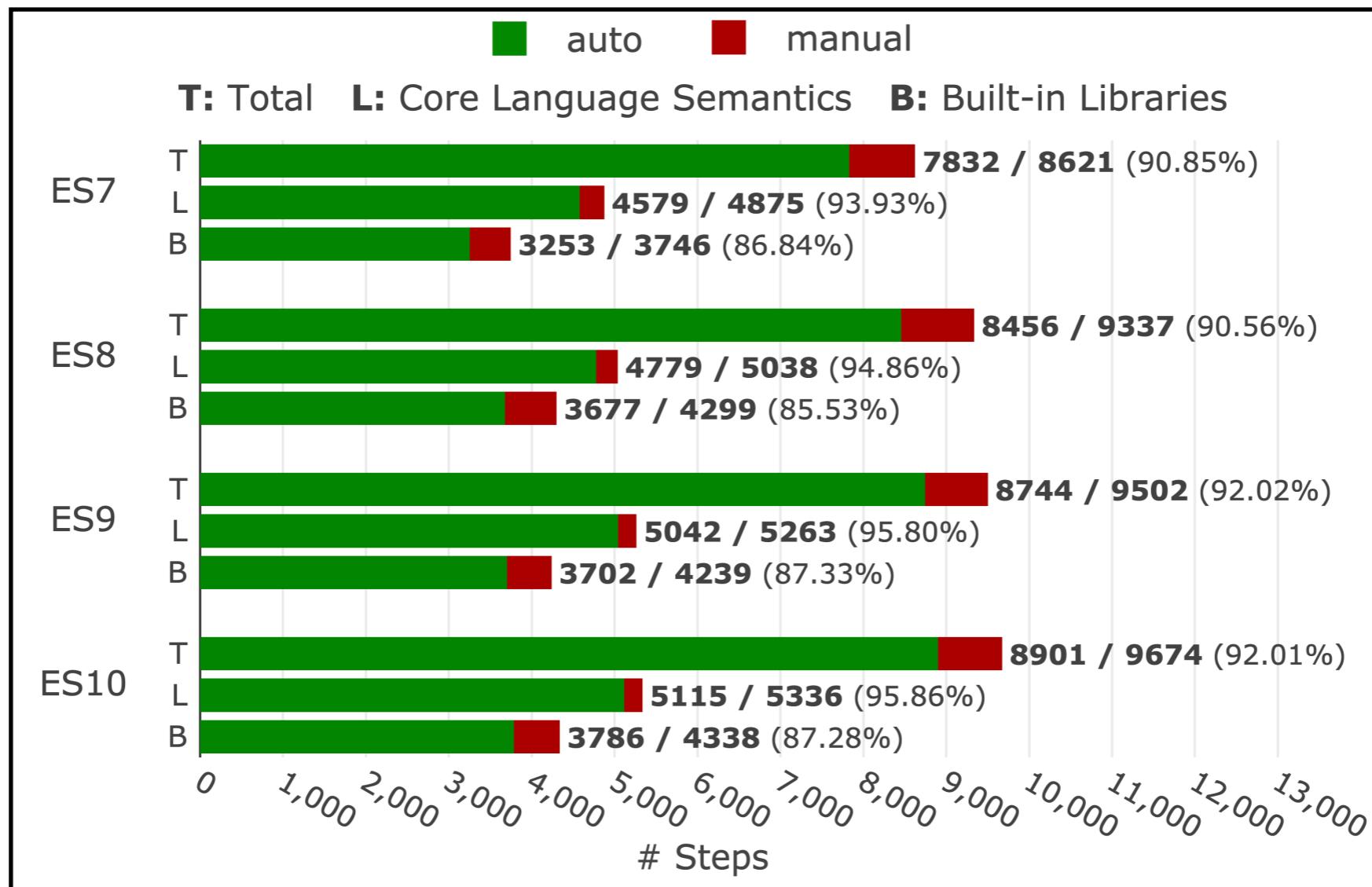


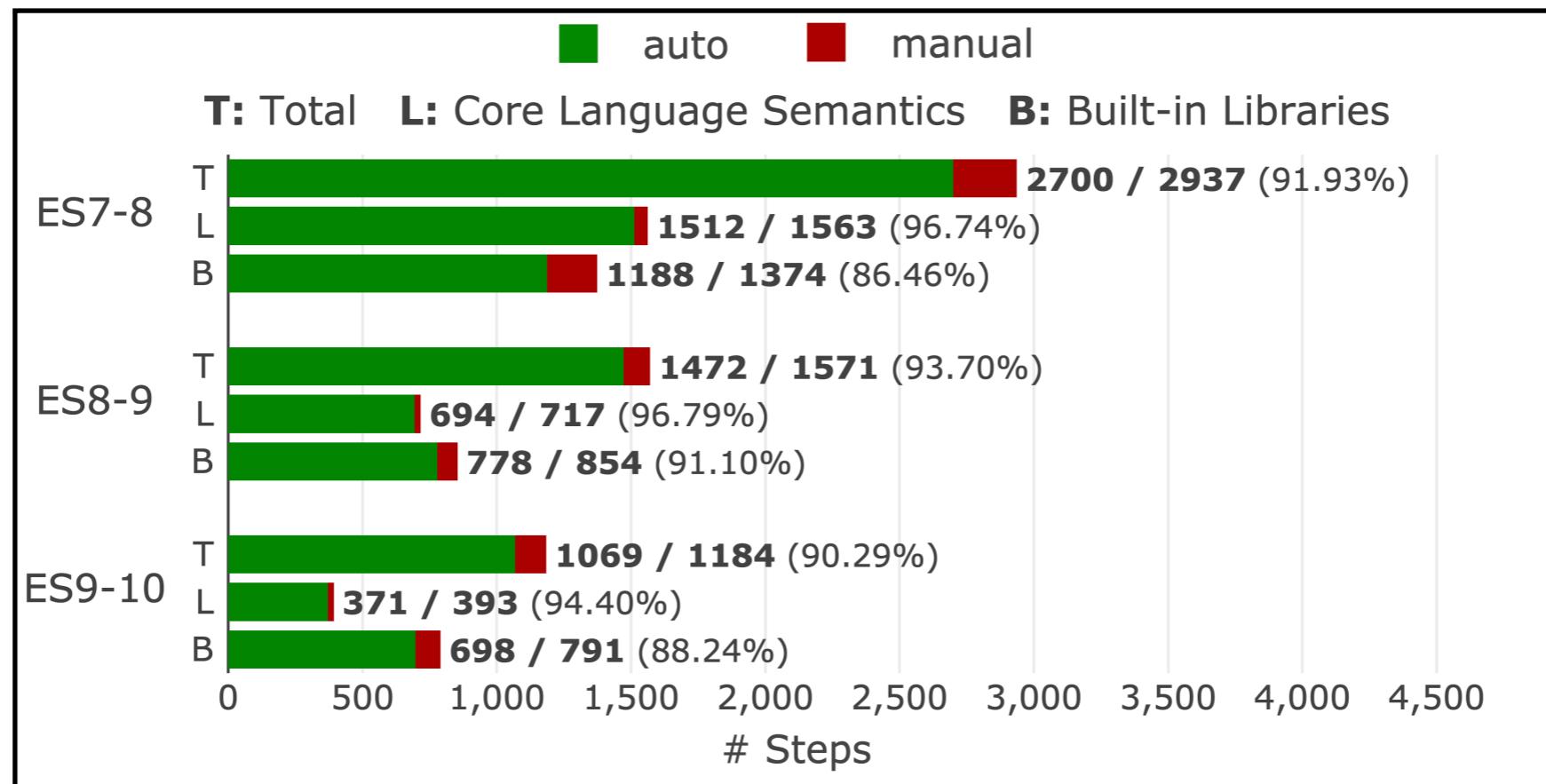




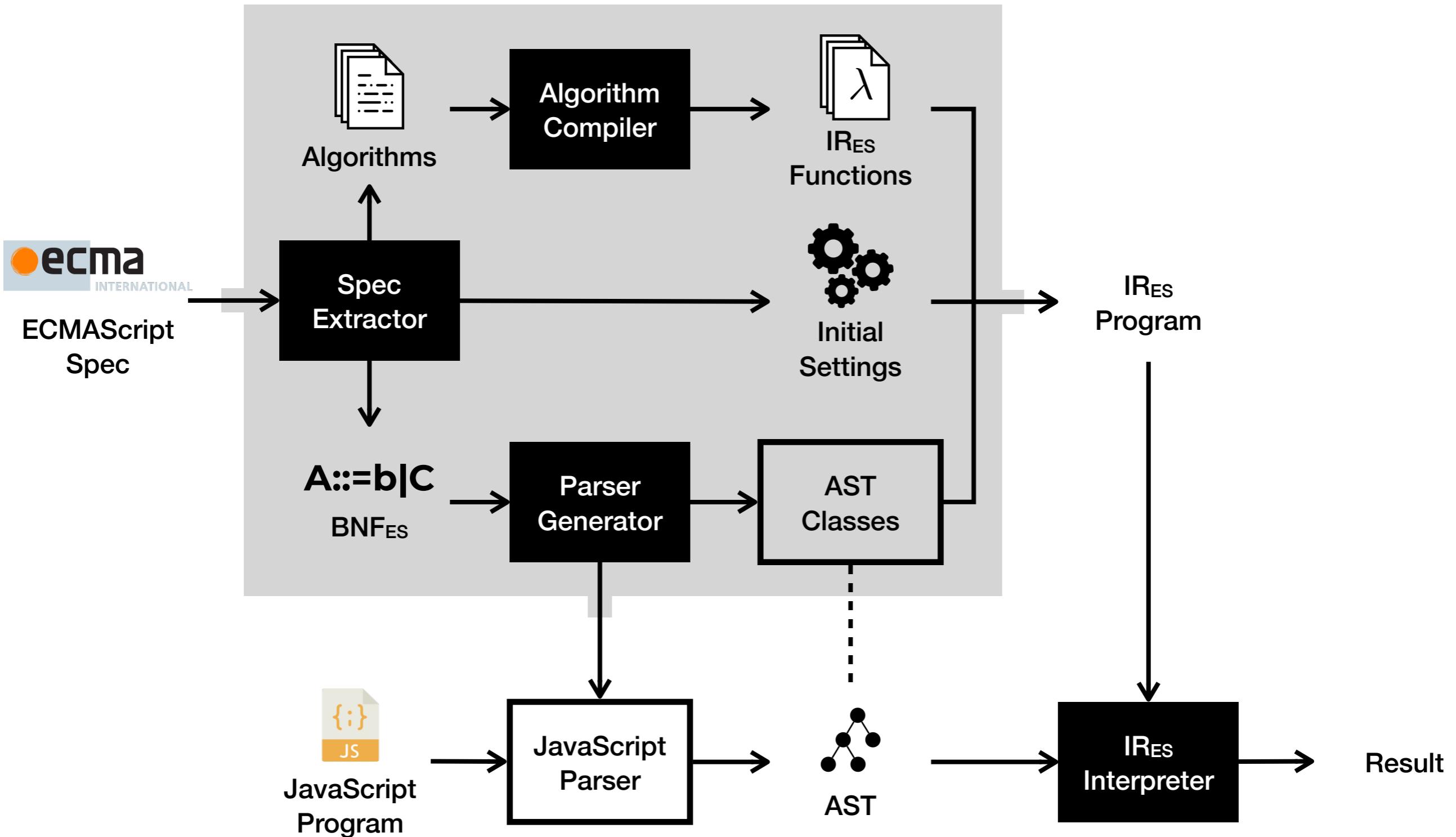
25.6.4.5.1 PromiseResolve (C , x)

1. **Assert:** Type(C) is Object.
2. If IsPromise(x) is true, then
 - a. Let $xConstructor$ be ? Get(x , "constructor").
 - b. If SameValue($xConstructor$, C) is true, return x .
3. Let $promiseCapability$ be ? NewPromiseCapability(C).
4. Perform ? Call($promiseCapability$.[[Resolve]], **undefined**, « x »).
5. Return $promiseCapability$.[[Promise]].





Automatic Semantics Extractor (ASE)





ECMAScript
Spec

Spec
Extractor

manual

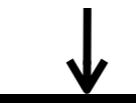
A::=b|C
BNF_{ES}

Section 3

JS Parser
Generator



Algorithms



Compile
Rules

Section 4

Algorithm
Compiler



IR_{ES}
Functions



Initial
Setting

