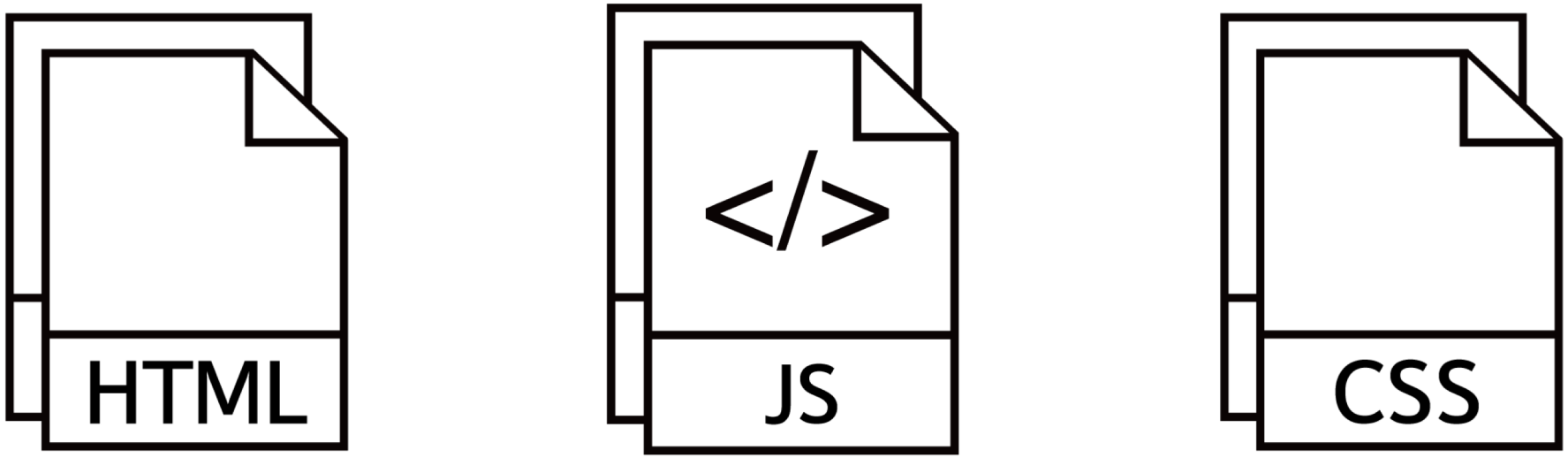
CodeAlchemist:
Semantics-Aware Code Generation to Find Vulnerabilities in JavaScript Engines

HyungSeok Han, DongHyeon Oh, Sang Kil Cha

KAIST

https://daramg.gift



https://daramg.gift

```
# id
uid=0(root) gid=0(root) groups=0(root)
# _
```

JS bugs are critical!

How to Find JS Bugs?

1. Analyzing JS Engine Code

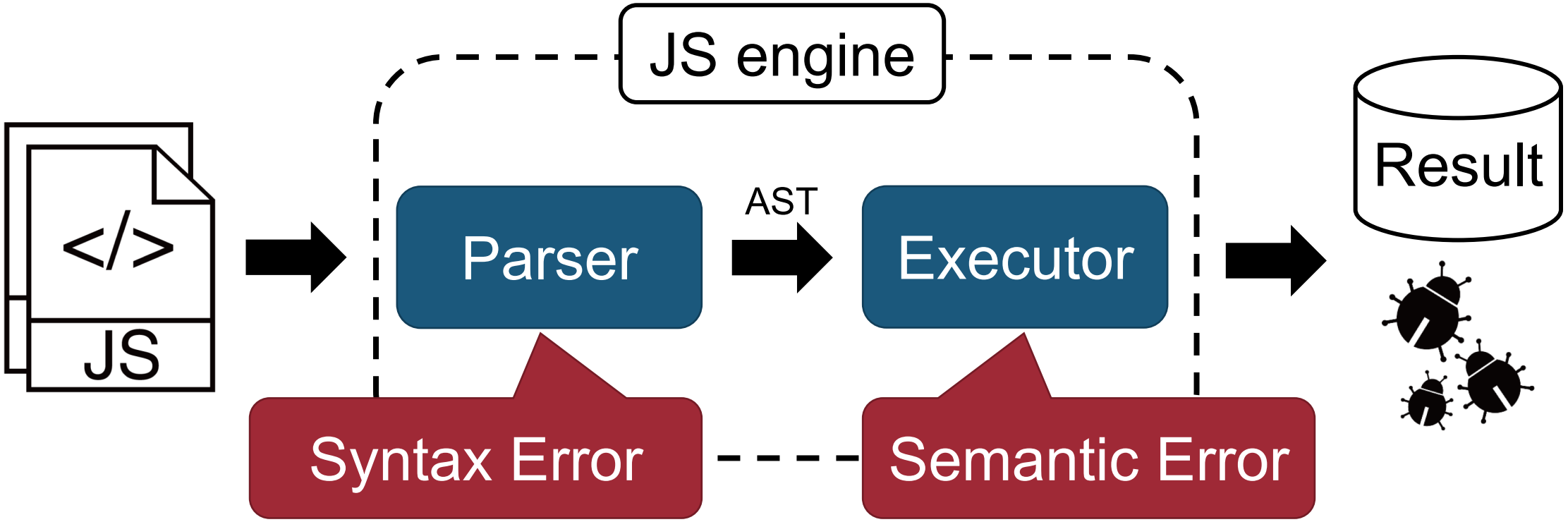
2. Fuzzing

How to Find JS Bugs?

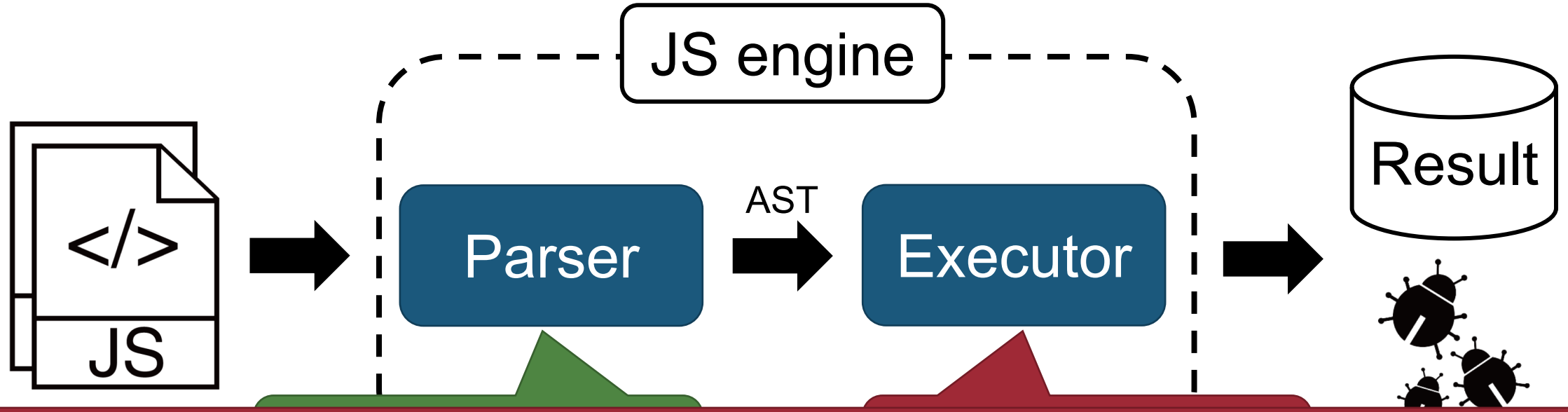
1. ~~Analyzing JS Engine Code~~

2. **Fuzzing**

Structure of JS Engine

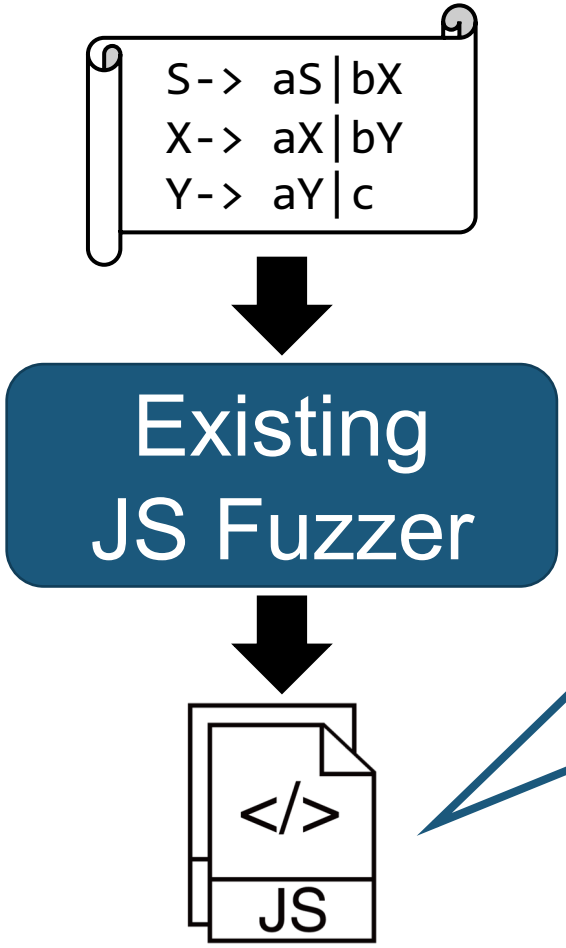


Previous Work: Grammar-based Fuzzer



***Semantics*-unaware**

Semantics-Unawareness




```
// ReferenceError
var arr = new Array (n);

// TypeError
var num = 10; num ();
```

Previous Work: Grammar-based Fuzzer

- jsfunfuzz
  - A state-of-the-art **generation-based** fuzzer developed by ***Mozilla***
  - Found **2,800** bugs since 2006

- LangFuzz
  - A state-of-the-art **mutation-based** fuzzer appeared at ***USENIX'12***
  - A parent of IFuzzer and TreeFuzz
  - Found **2,300** bugs since 2011

Semantics-Unawareness of jsfunfuzz

jsfunfuzz

JS code

100,000 JS code snippets

Suppress semantic errors

```
try {
  var n = 42, buf = [x, 2, 3];
  …
} catch(e) {}
```

Semantics-Unawareness of jsfunfuzz

jsfunfuzz

JS code

100,000 JS code snippets

Suppress semantic errors

```
// try {
  var n = 42, buf = [x, 2, 3];
  …
// } catch(e) {}
```

Semantics-Unawareness of jsfunfuzz

Kind	# of Occurrences			
				
Syntax Error	18,200	17,429	17,998	17,135
Range Error	310	285	328	308
Reference Error	78,294	79,116	78,401	78,935
Type Error	3,196	3,169	3,273	3,507
URI Error	0	0	0	0
Custom Error	0	1	0	115
Total Count	100,000	100,000	100,000	100,000

Previous Work: Grammar-based Fuzzer

- jsfunfuzz
  - A state-of-the-art **generation-based** fuzzer developed by ***Mozilla***
  - Found **2,800** bugs since 2006

- LangFuzz
  - A state-of-the-art **mutation-based** fuzzer appeared in ***USENIX'12***
  - A parent of IFuzzer and TreeFuzz
  - Found **2,300** bugs since 2011

Semantics-Unawareness of LangFuzz

```
var arr = new Array (0x100);
for(let i = 0; i < 0x100; i++){
  // i is only available here
  arr[i] = i;
}
```

Semantics-Unawareness of LangFuzz

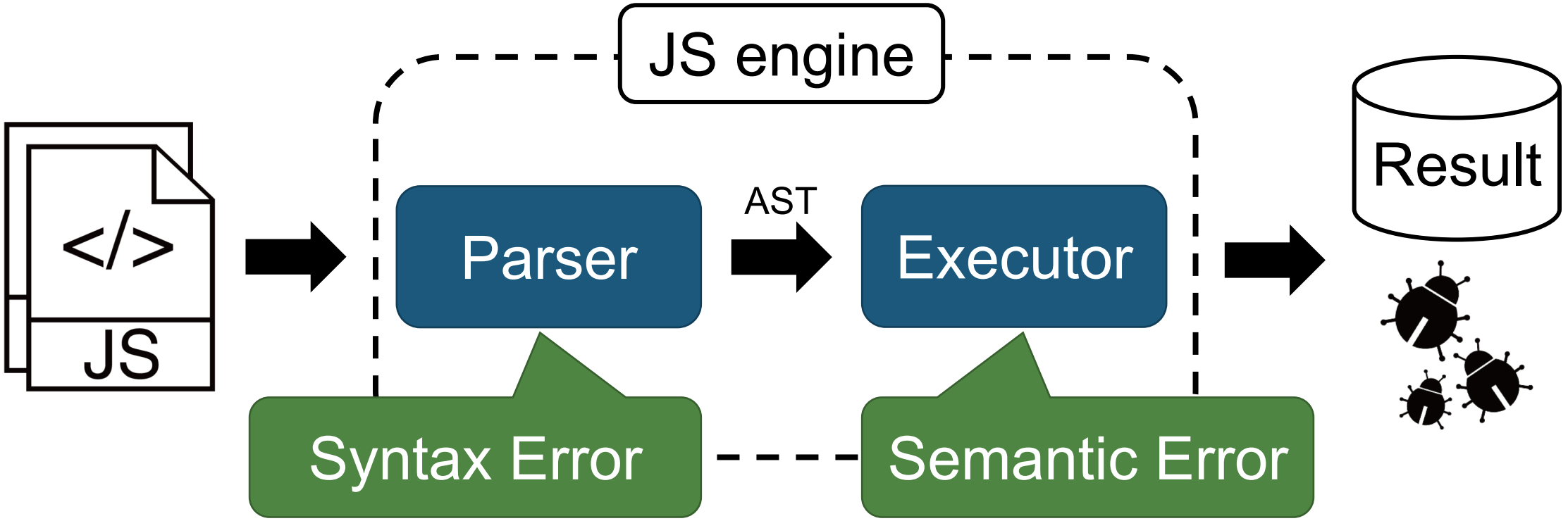
```
var x = new String (y);
for(let i = 0; i < 0x100; i++){
  // i is only available here
  arr[i] = i;
}
```

Semantics-Unawareness of LangFuzz

```
var arr = new String (i);
for(let i = 0; i < 0x100; i++){
  // i is only available here
  arr[i] = i;
}
```

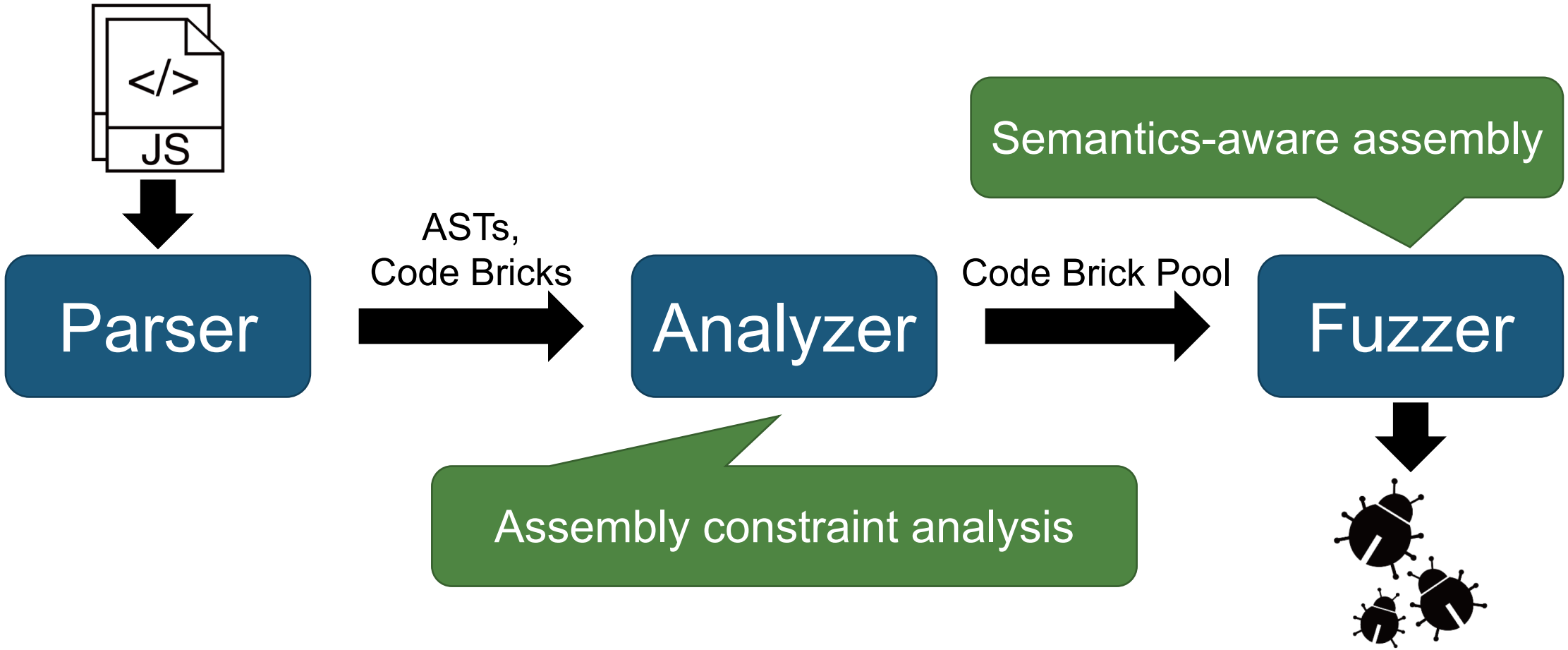
```
ReferenceError: i is not defined
```

Our Goal: Be Semantics-Aware

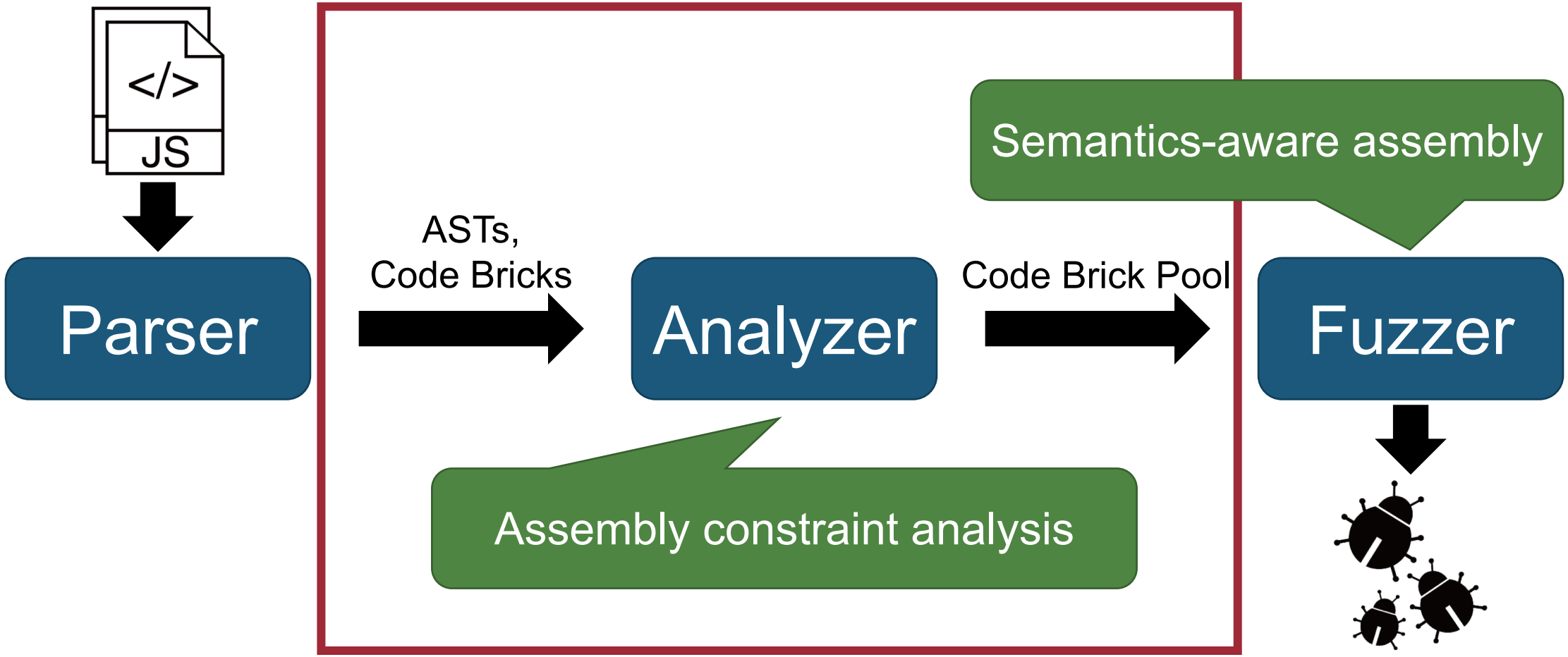


Intuition: Assemble Code Bricks by Assembly Constraints

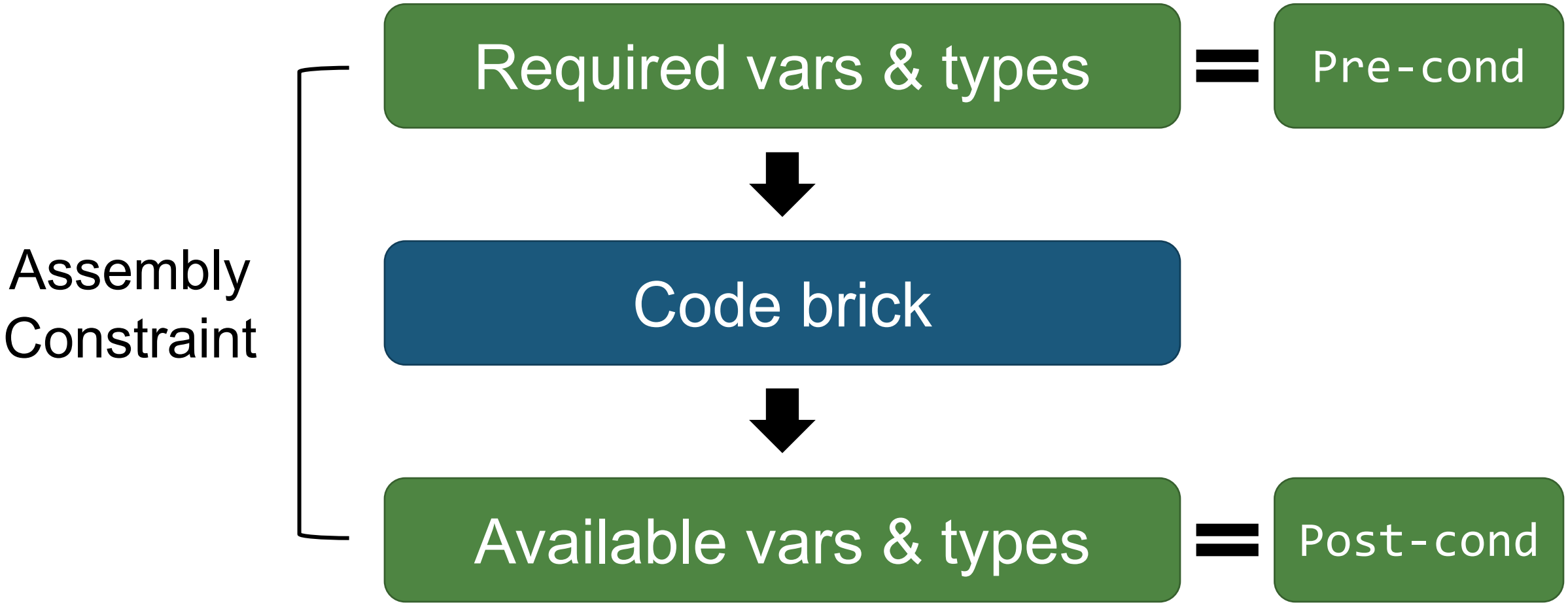
CodeAlchemist: Semantics-Aware Code Generation for Fuzzing



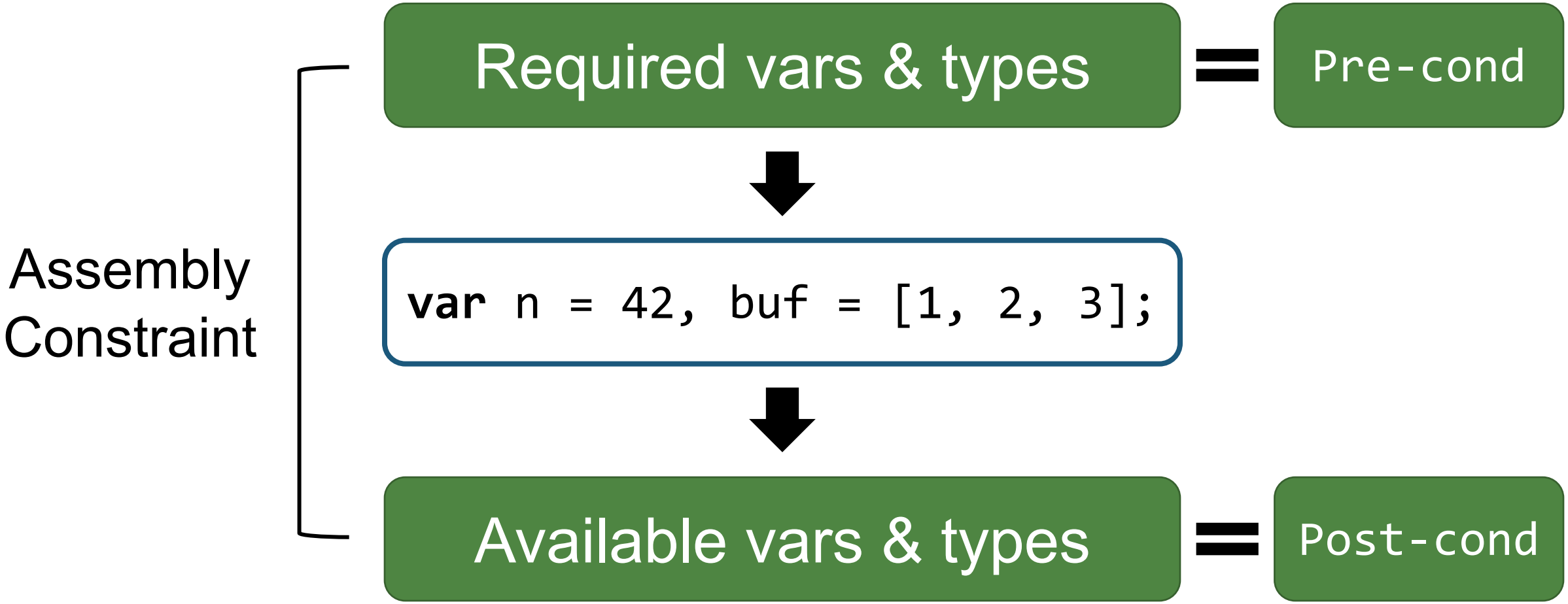
How to Analyze Assembly Constraints?



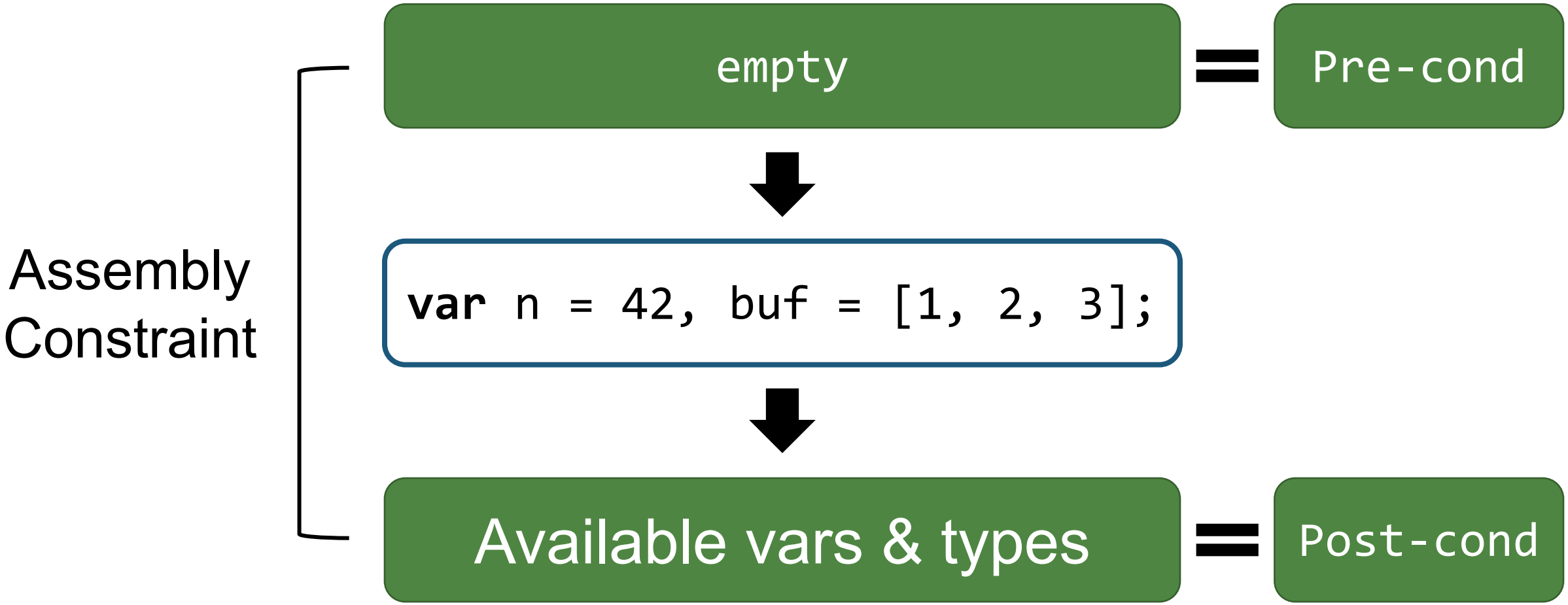
Assembly Constraint



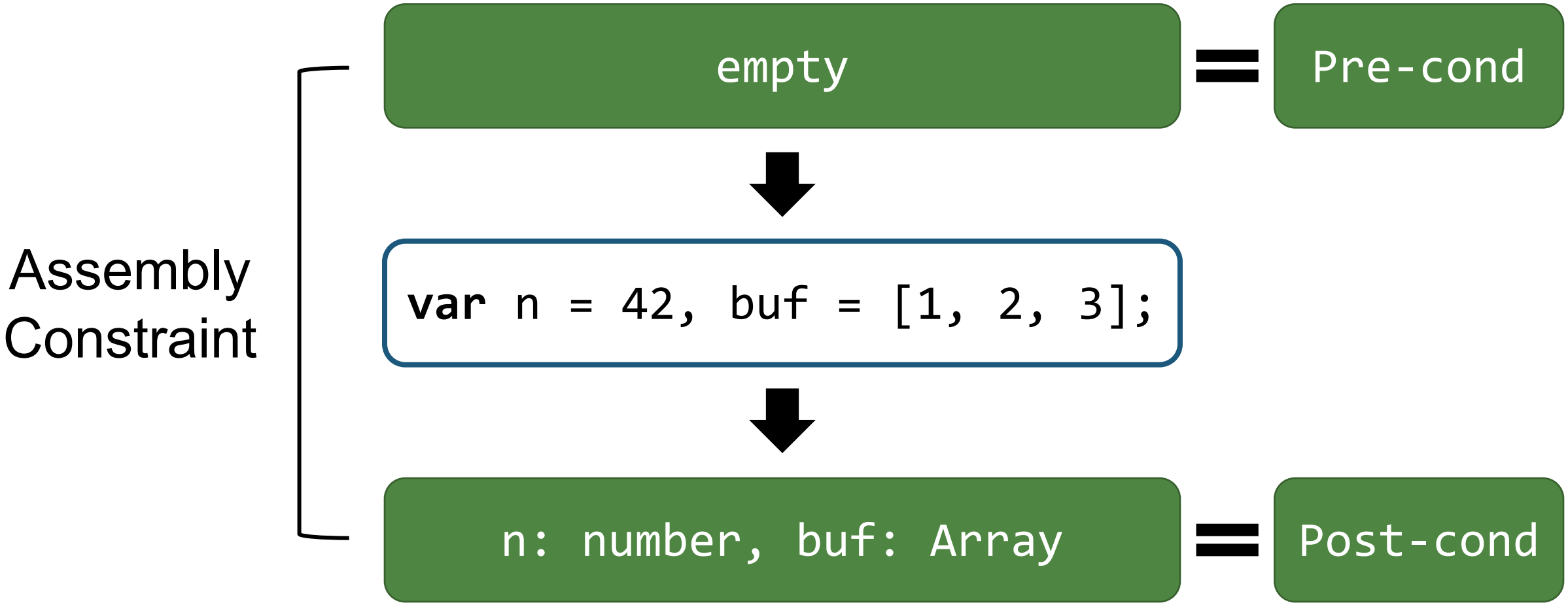
Assembly Constraint



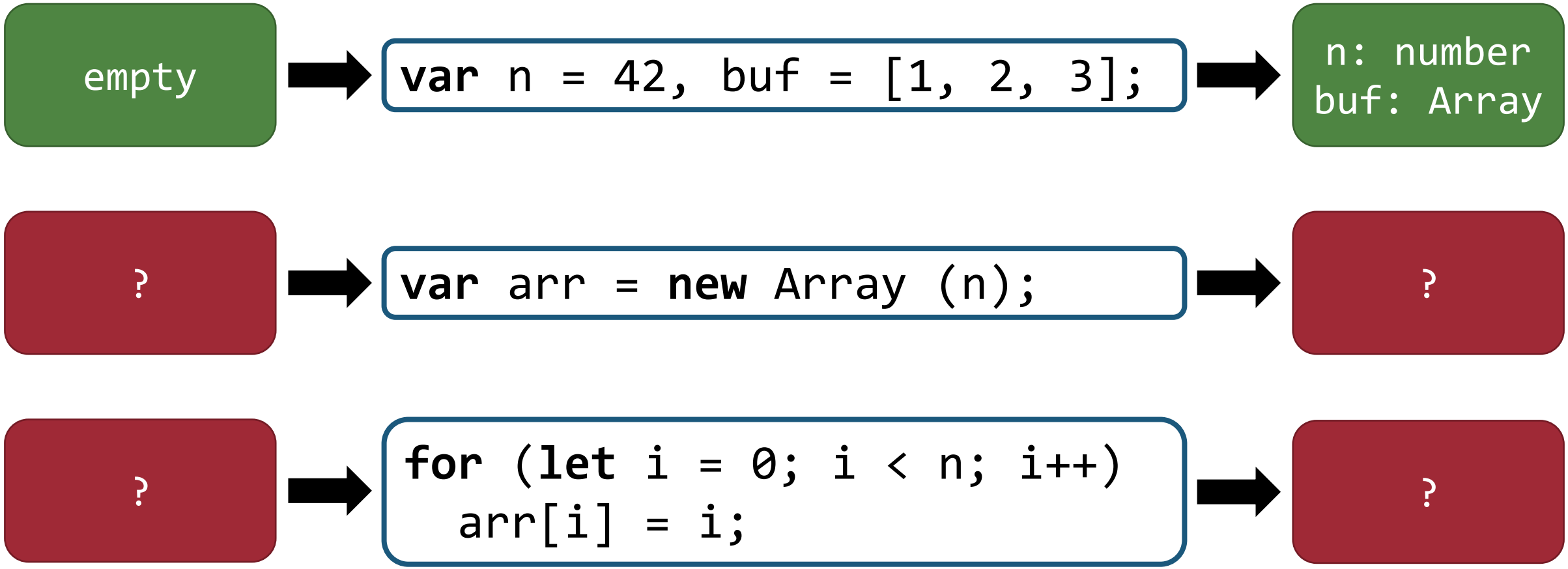
Assembly Constraint



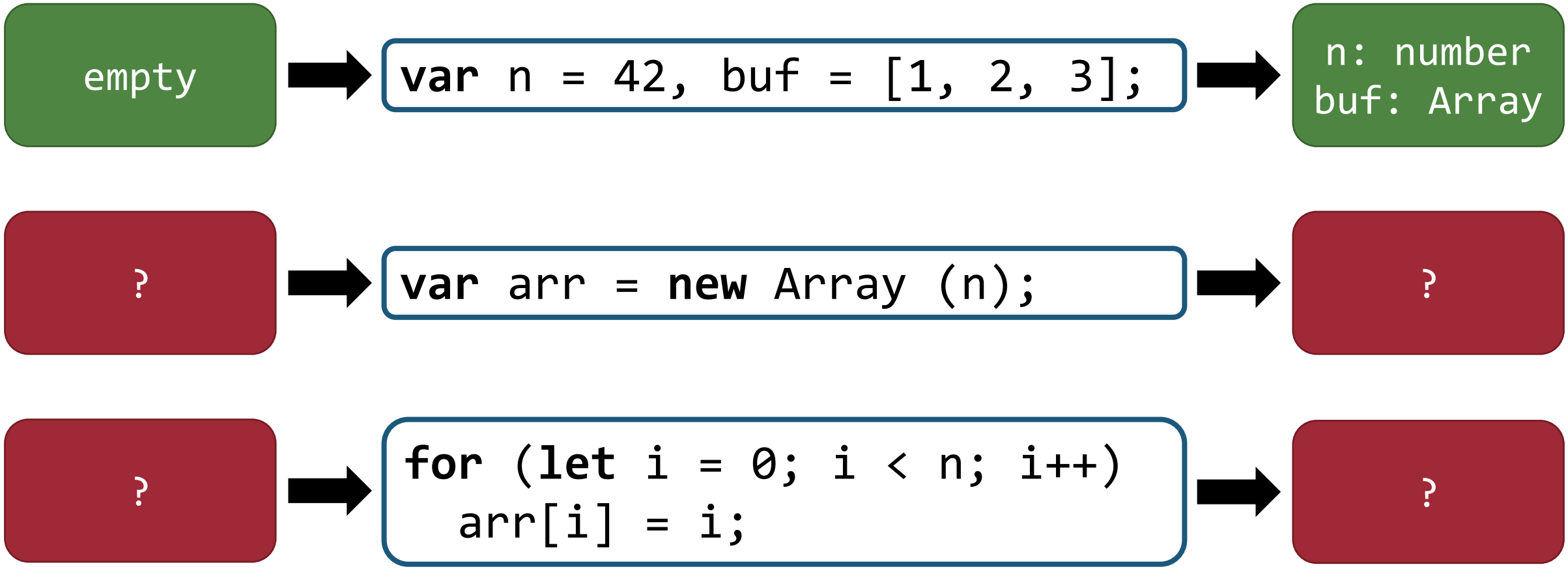
Assembly Constraint



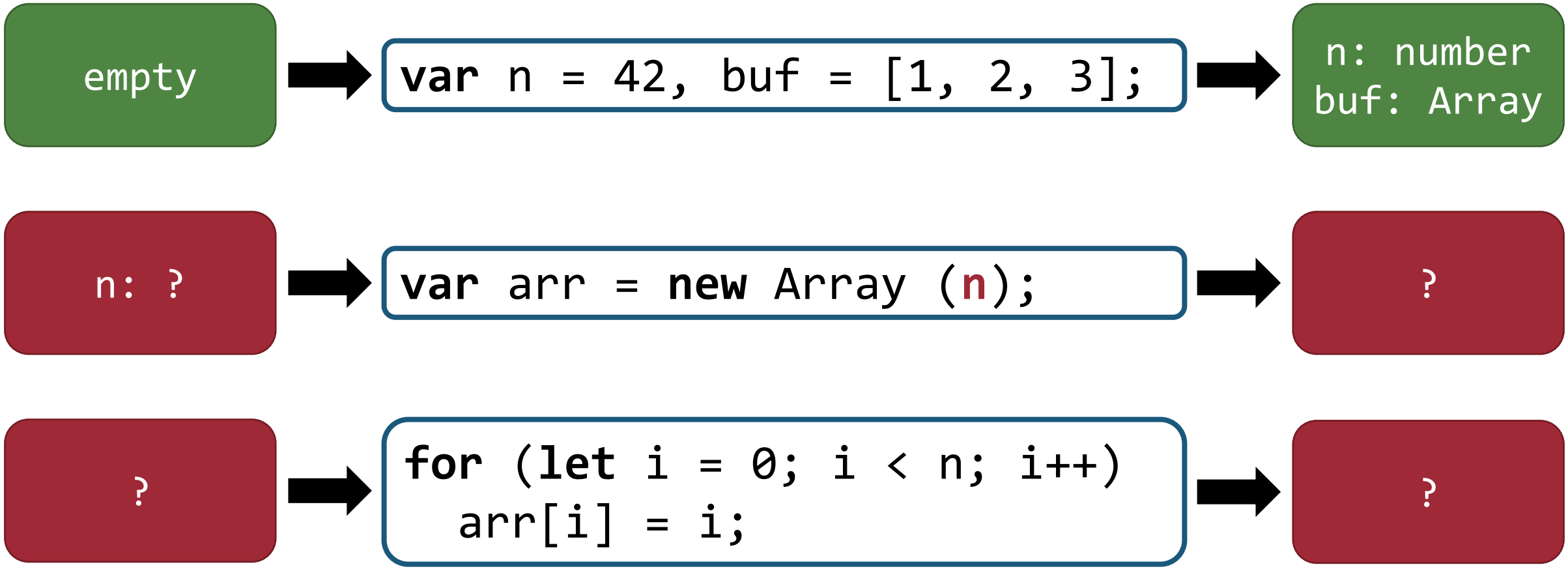
Assembly Constraint Analysis



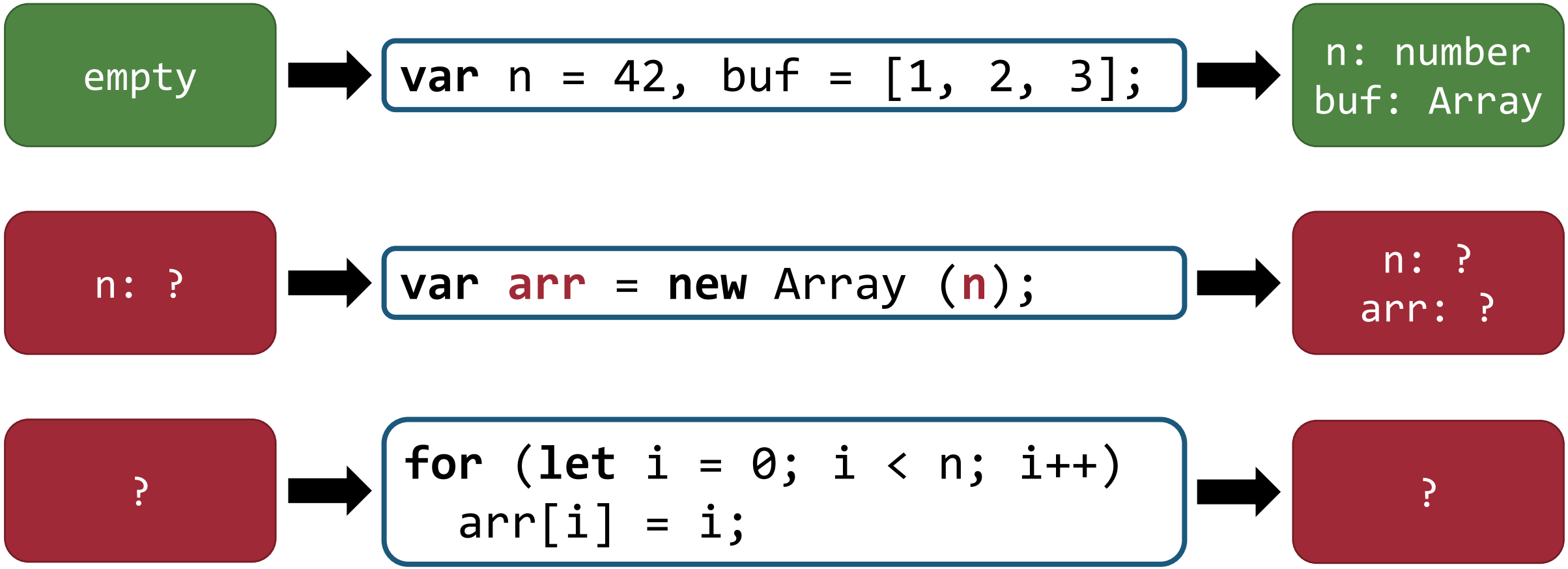
Data-flow Analysis



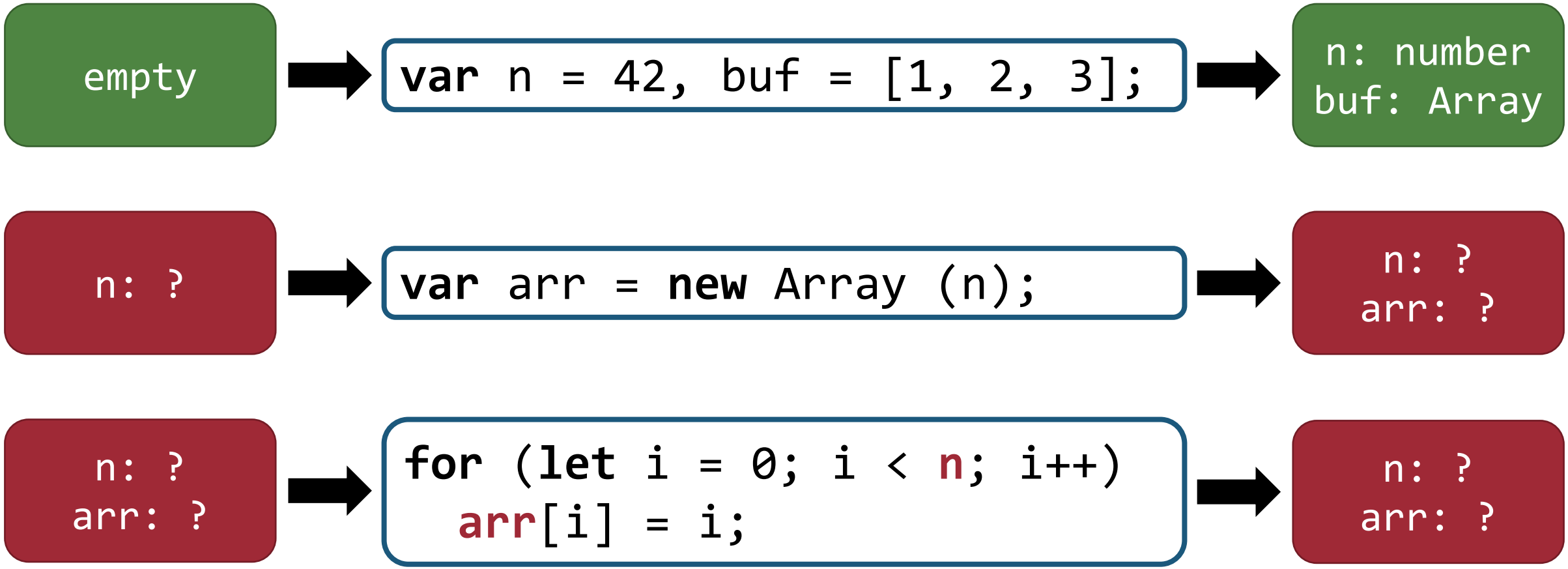
Data-flow Analysis



Data-flow Analysis

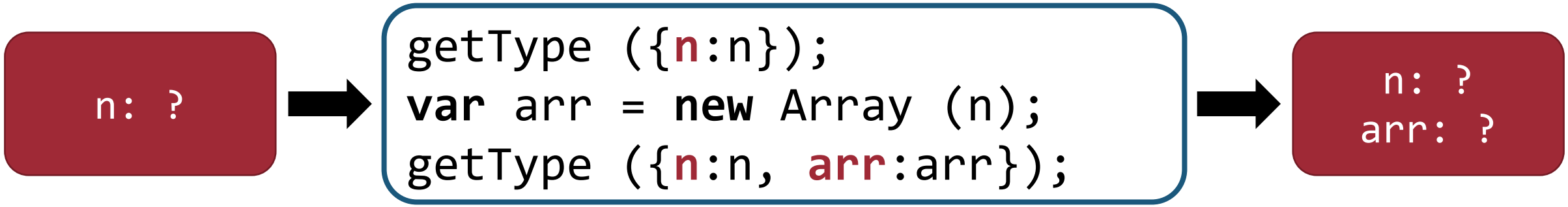


Data-flow Analysis

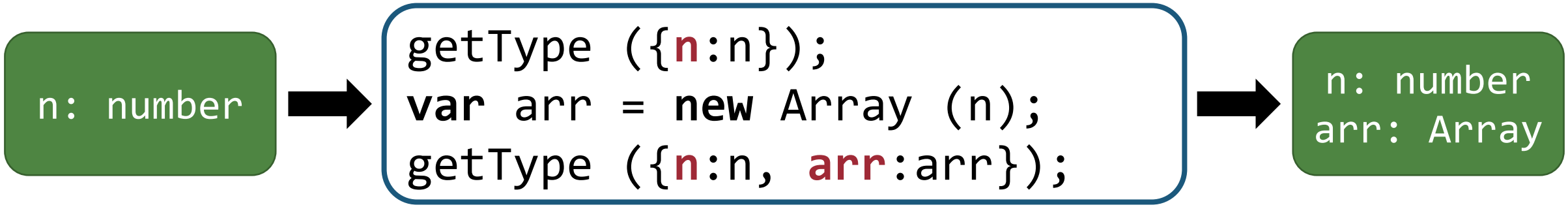


Dynamic Type Analysis

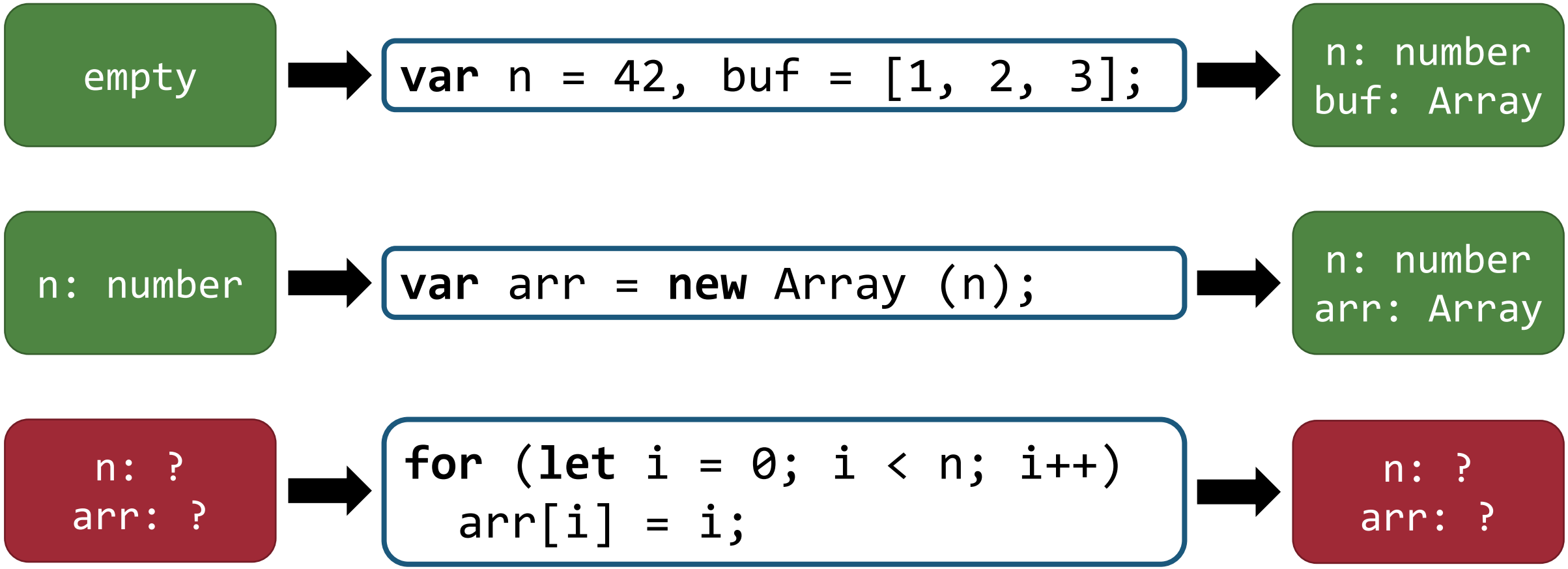
Dynamic Type Analysis



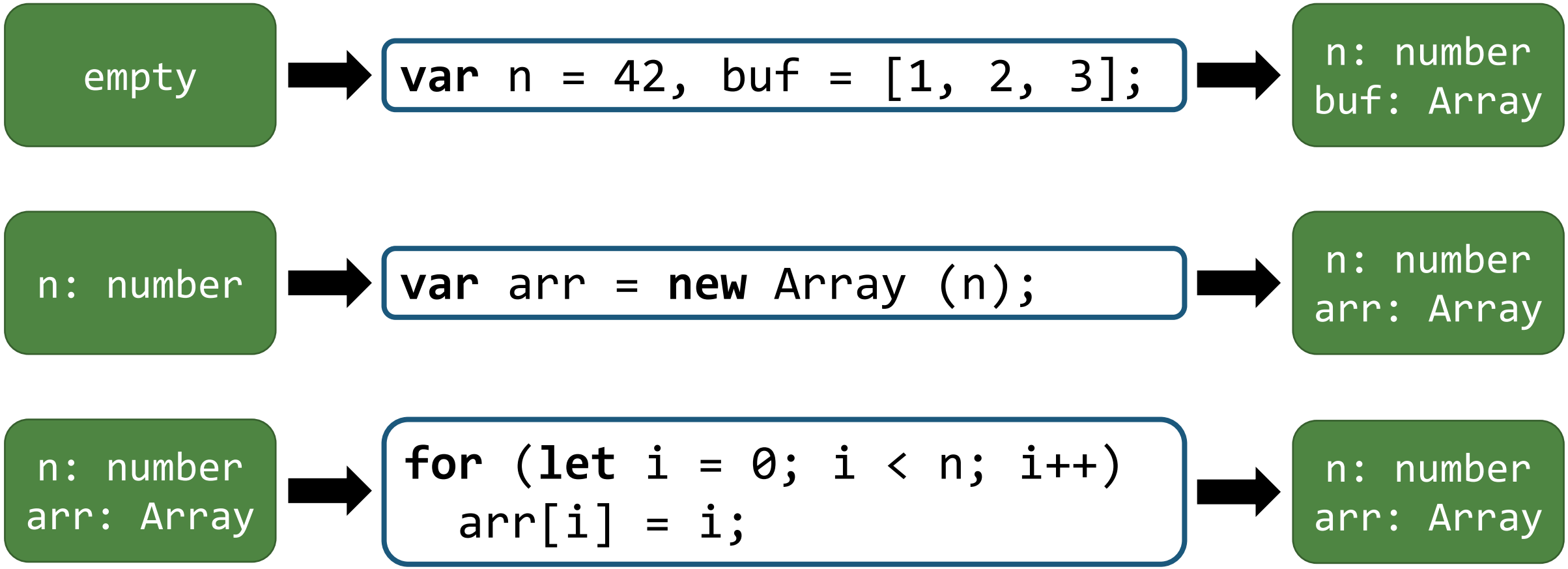
Dynamic Type Analysis



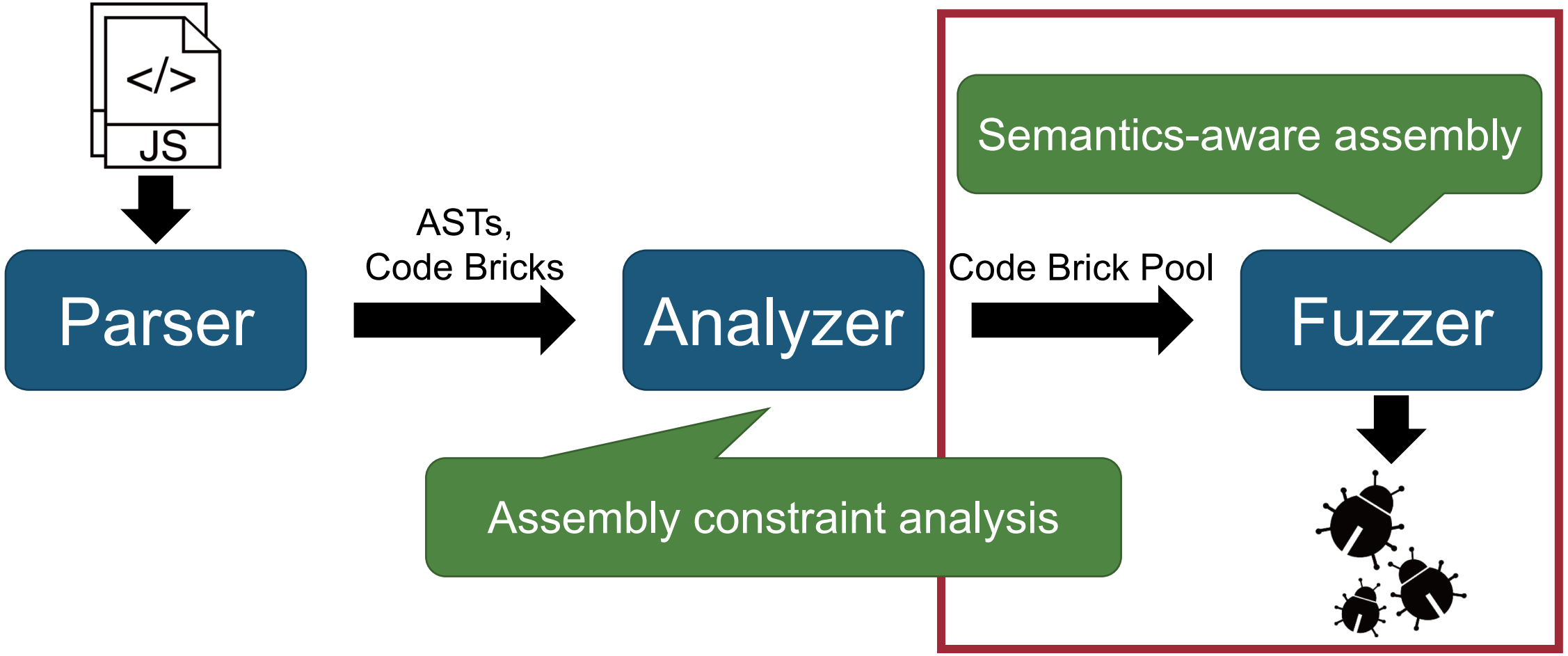
Dynamic Type Analysis



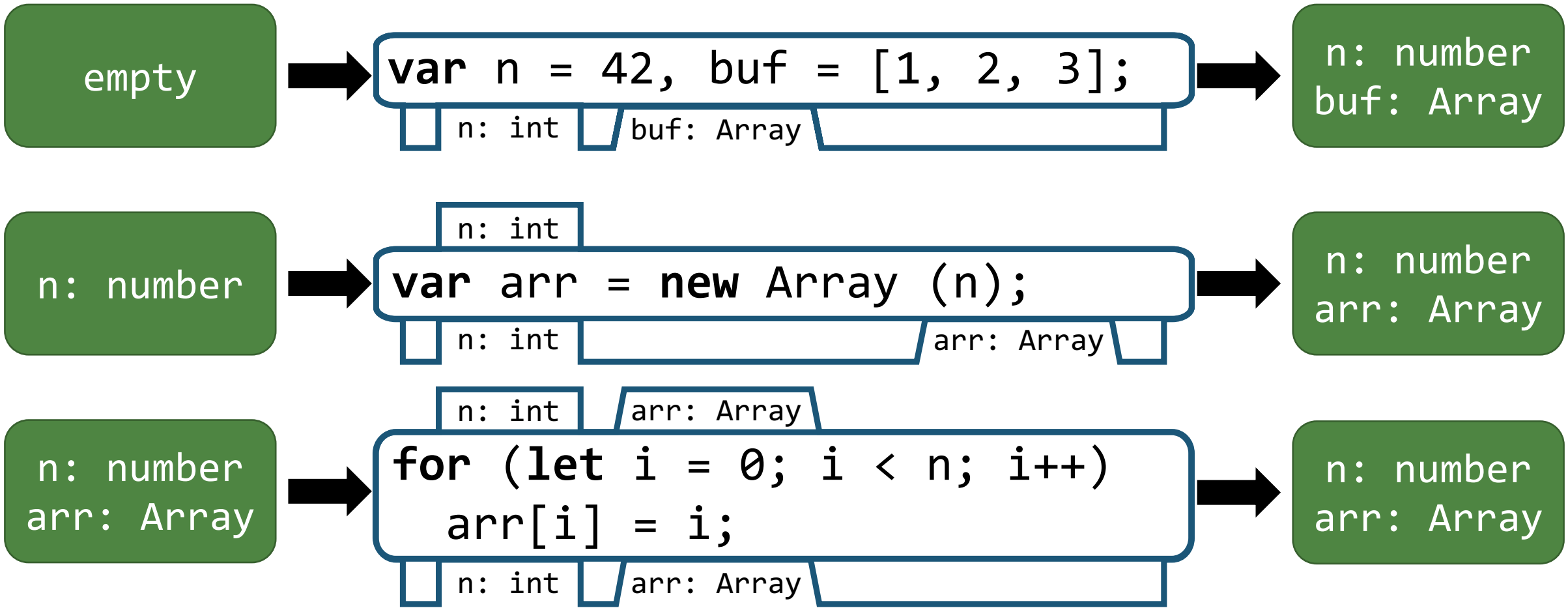
Assembly Constraint Analysis



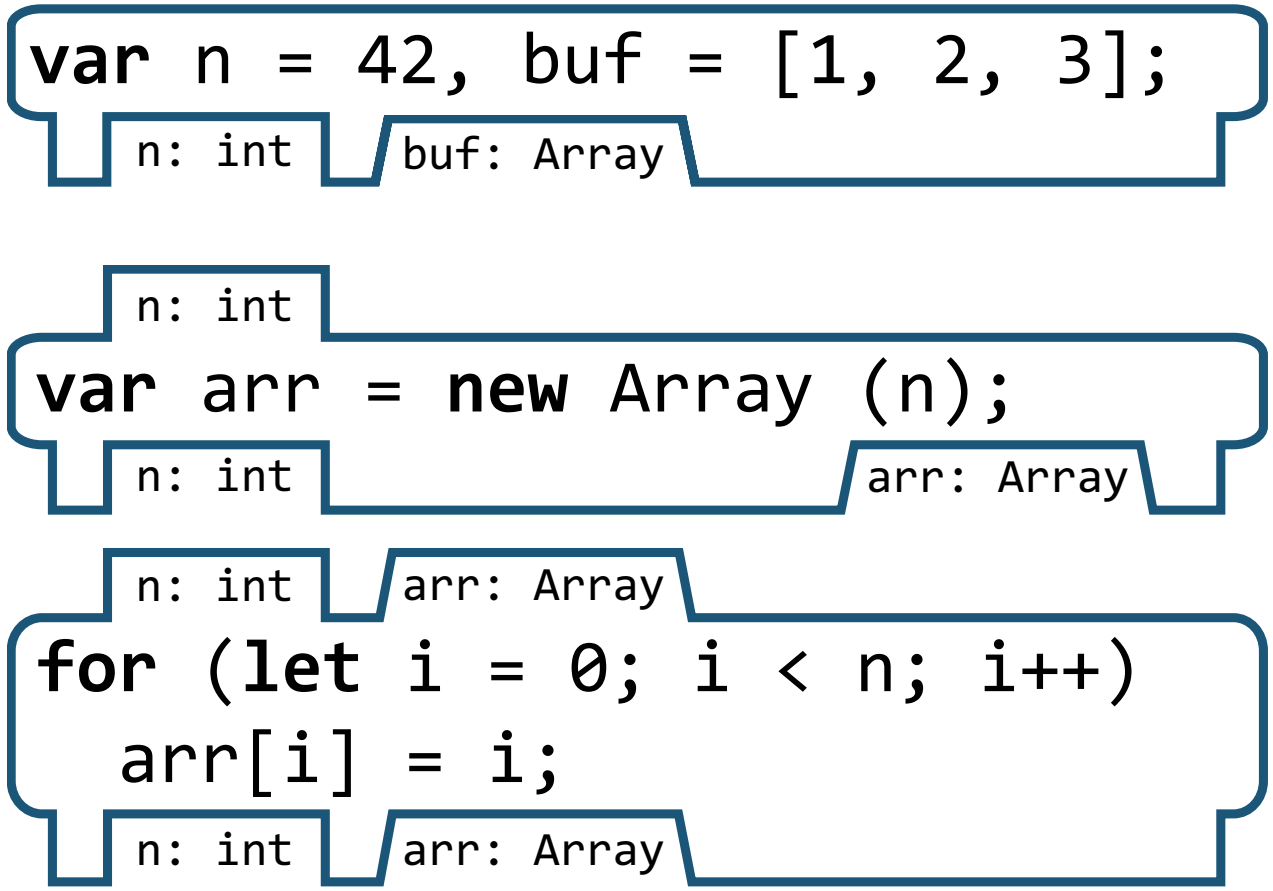
How to Assemble Code Bricks?



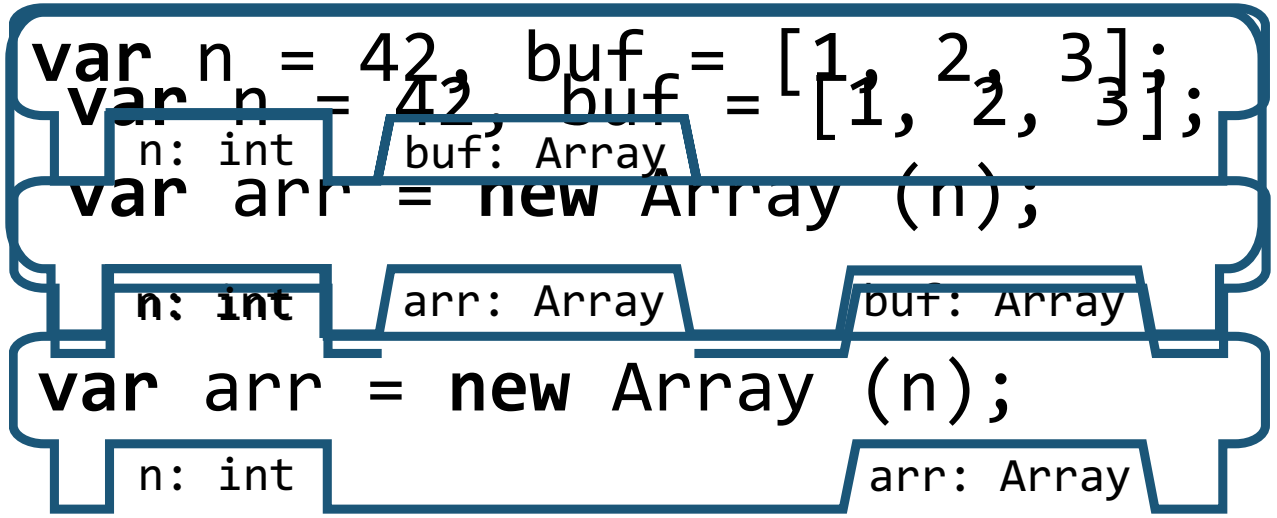
Code Bricks with Teeth & Holes



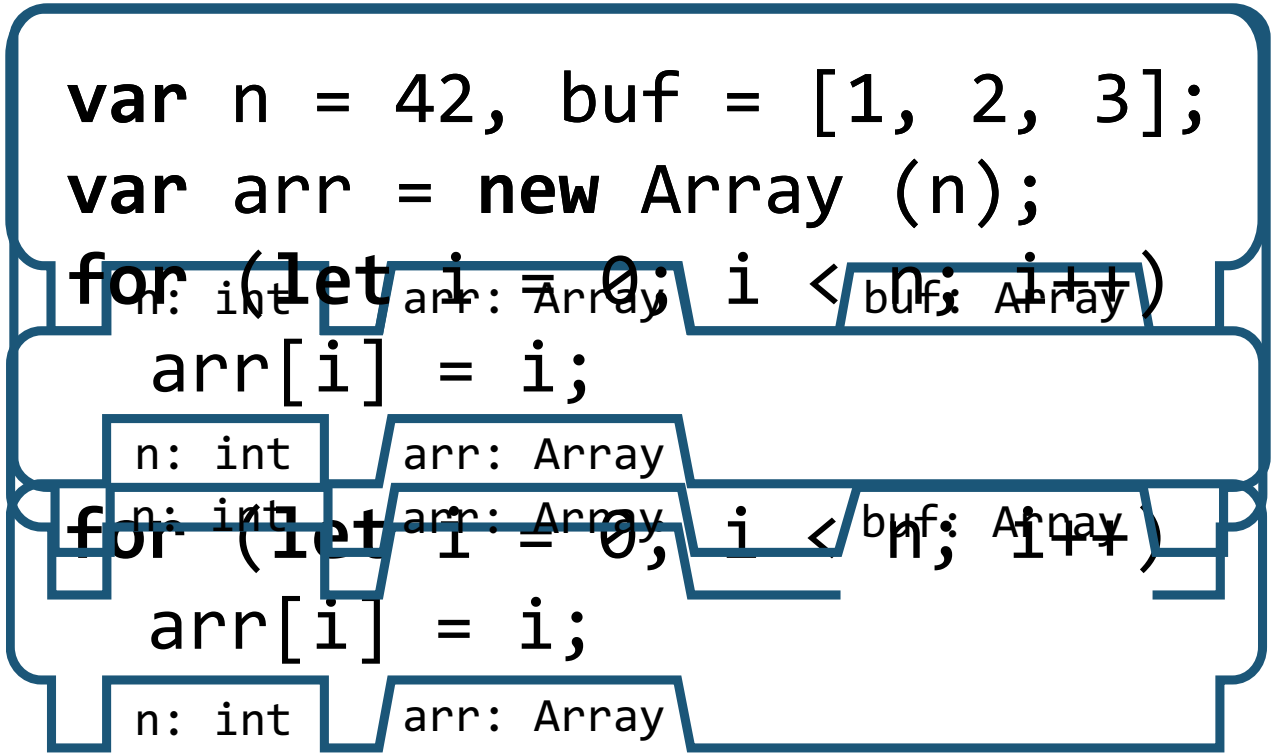
Semantics-Aware Assembly



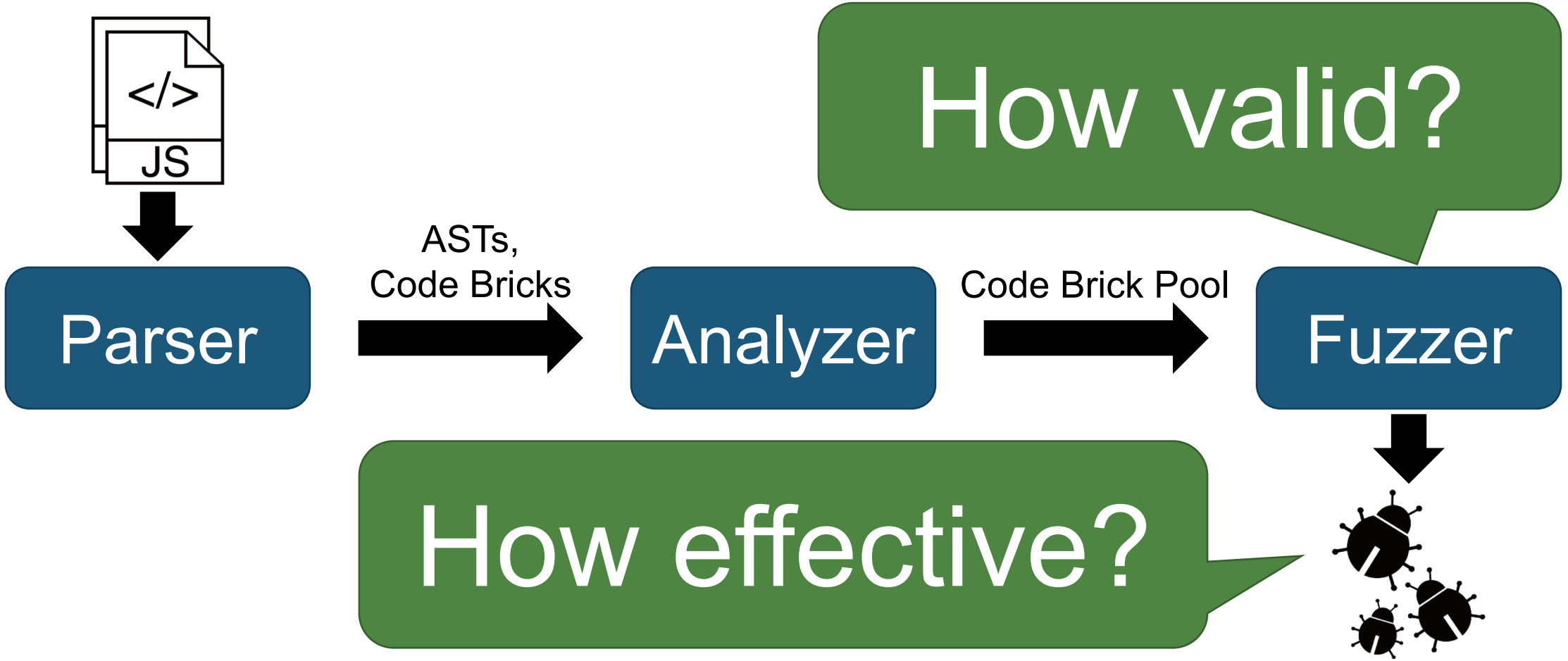
Semantics-Aware Assembly



Semantics-Aware Assembly



Evaluation

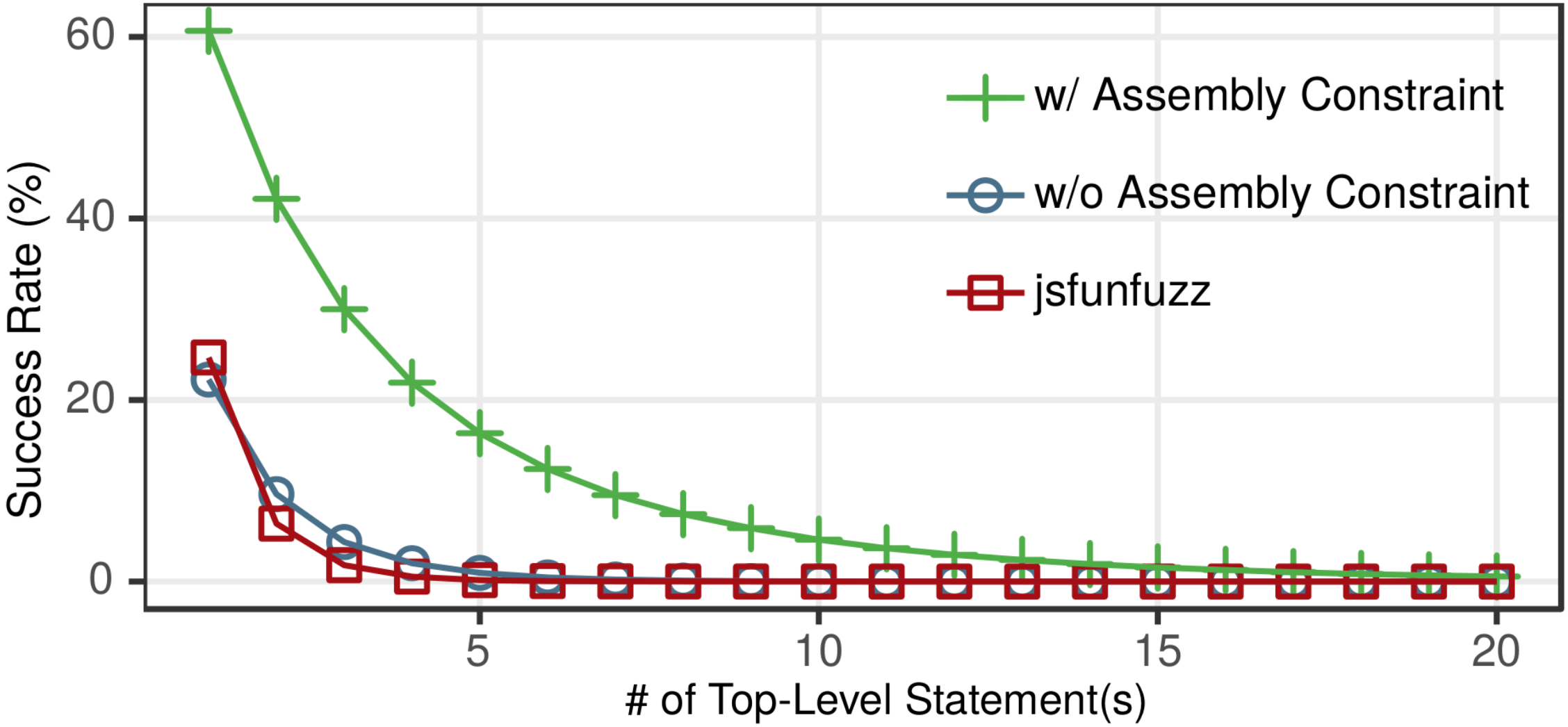

JS
Parser
ASTs,
Code Bricks
Analyzer
Code Brick Pool
Fuzzer
How valid?
How effective?

Experiment Setup

- Collect about **63,000** JS code snippets
  - Regression tests in four major JS engines
  - Test code snippets in Test262
  - PoC exploits for previous security bugs

- The latest JS engines as of July 10th, 2018
  - ChakraCore 1.10.1
  - V8 6.7.288.46
  - JavaScriptCore 2.20.3
  - SpiderMonkey 61.0.1

Validity of Generated JS



vs. State-of-the-Arts

- jsfunfuzz: A **state-of-the-art** JS fuzzer developed by **Mozilla**
- IFuzzer: A variant of LangFuzz, ***ESORICS'16***
- Running time: 24 hours x 4 engines = 96 hours

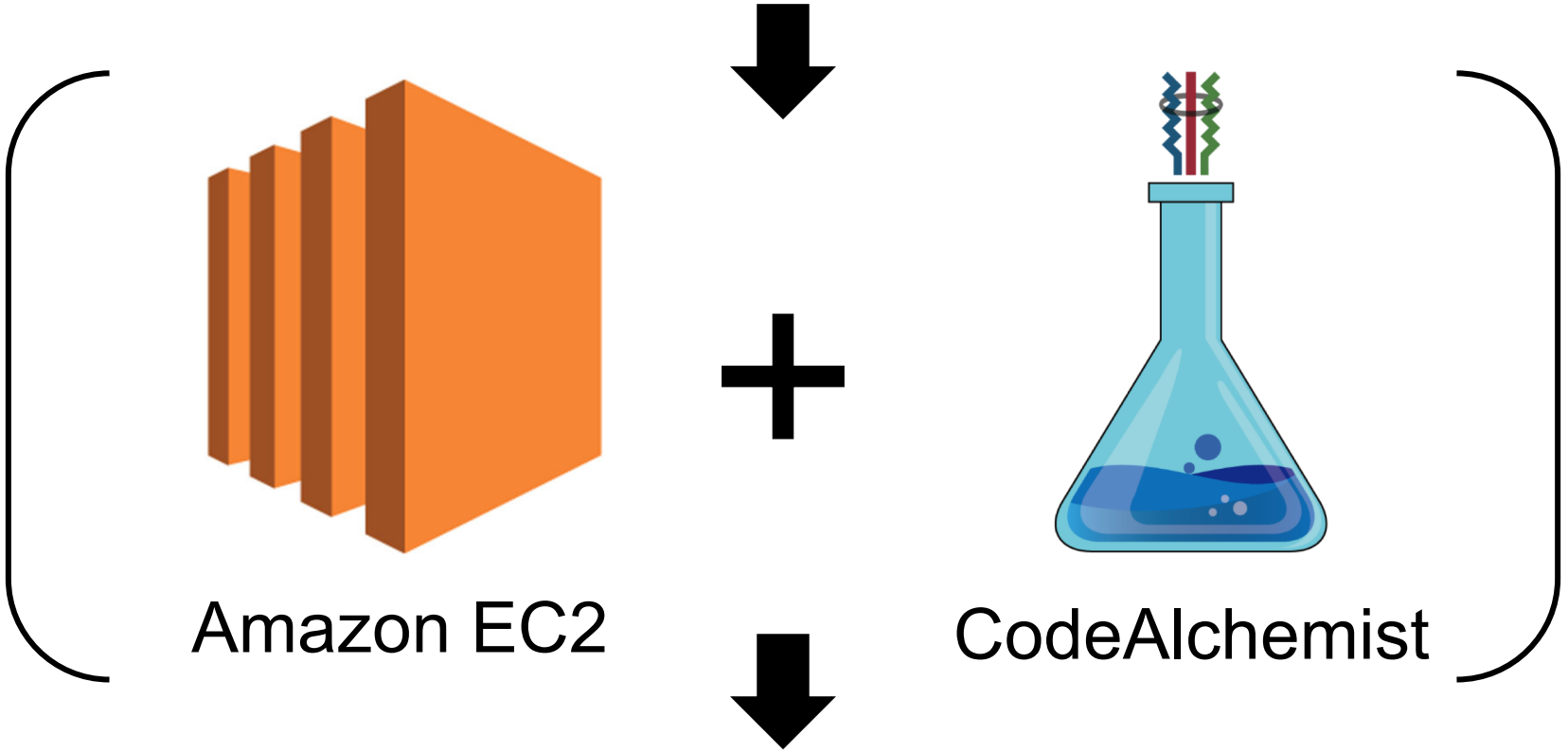
JS Engine	CodeAlchemist	jsfunfuzz	IFuzzer
ChakraCore 1.10.1	6	0	0
JavaScriptCore 2.20.3	6	3	0
V8 6.7.288.46	2	0	0
SpiderMonkey 61.0.1	0	0	0

Real-World Bug Finding

- Ran a week for the latest JS engines
  - JavaScriptCore: 2.20.3, 2.21.4 (beta)
  - V8: 6.7.288.46
  - SpiderMonkey: 61.0.1
  - ChakraCore: 1.10.0, 1.10.1
- Found 19 unique bugs
  - 11 exploitable bugs
  - 3 CVEs for us

$5.424 / hours x 24 hours x 7 days x 6 engines

= $5,500



$200,000 / exploitable bugs x 11 bugs

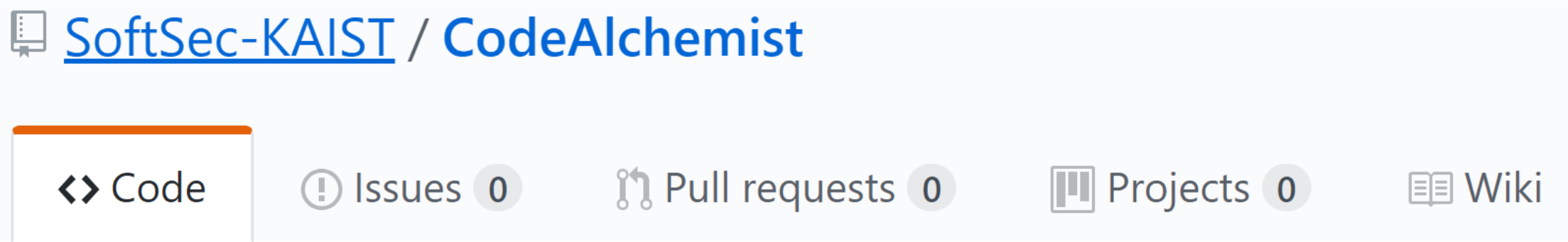
= $2,200,000

Future Research

- Seed selection
- Simple random code brick selection
- Supporting other language interpreters or compilers

Open Science

https://github.com/SoftSec-KAIST/CodeAlchemist



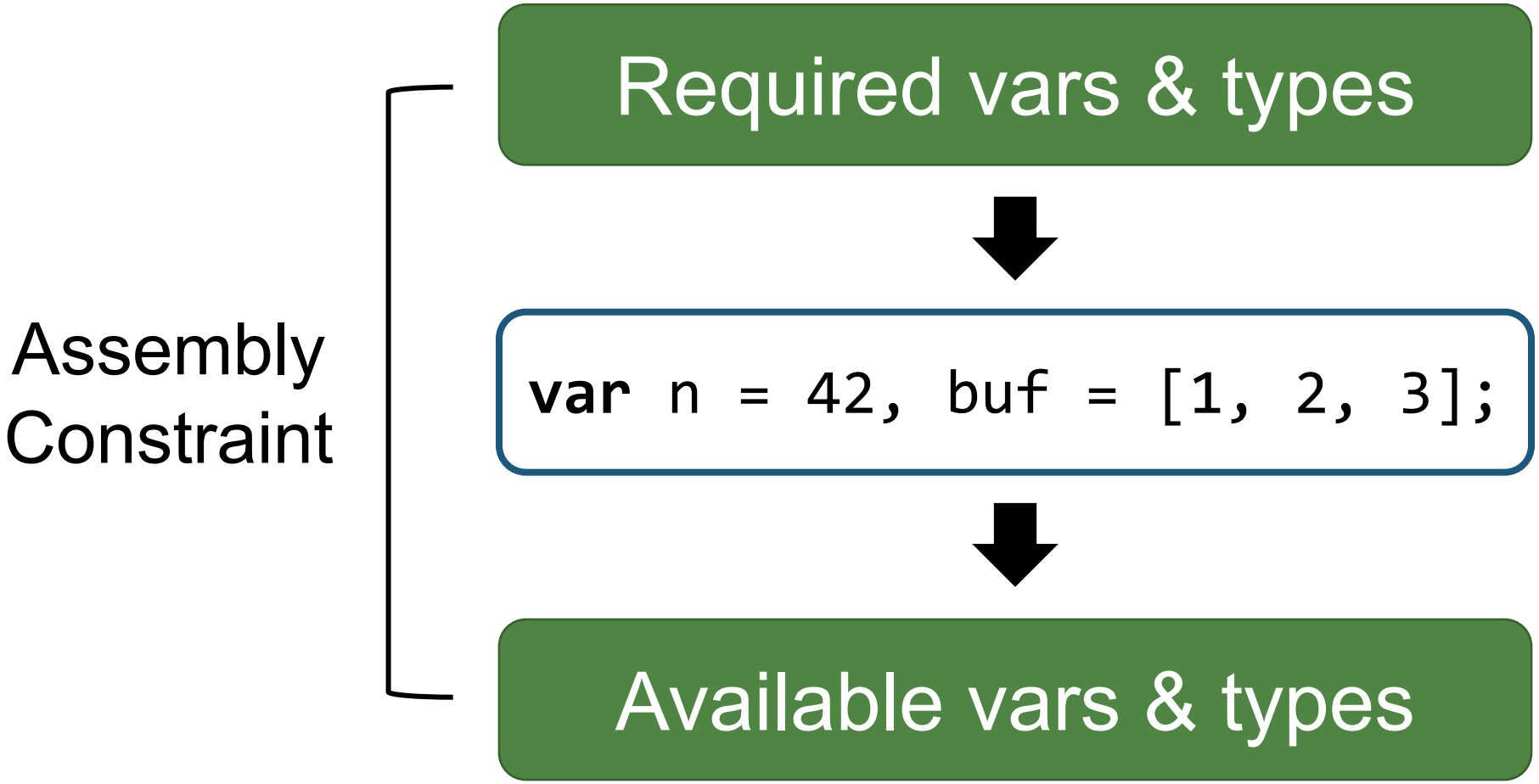
Release in March

More in the Paper

- Fragmentization
- Code brick generation
- Parameters of CodeAlchemist
- Other evaluation

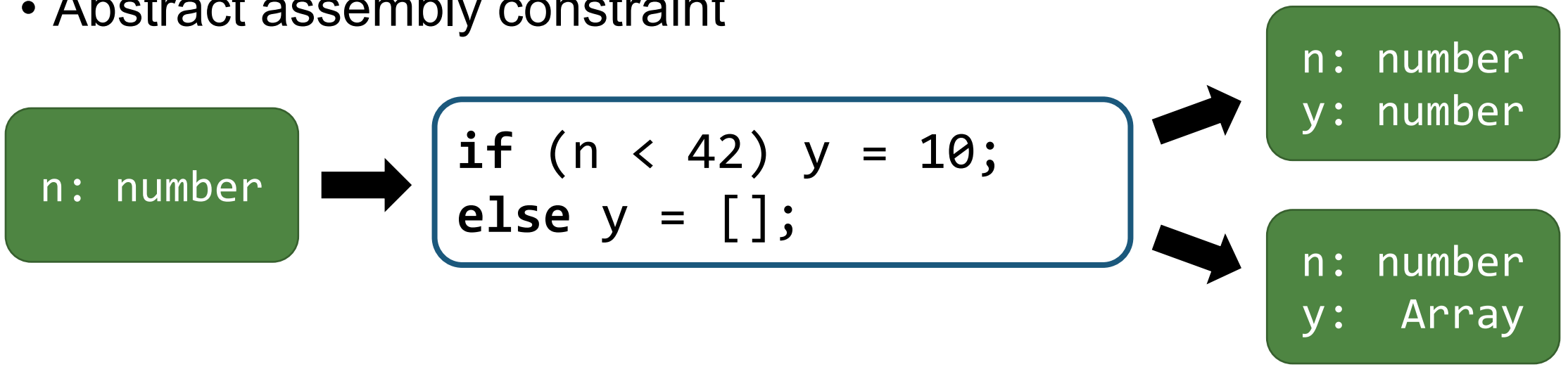
Question?

Assembly Constraint



Why not 100% Success?

- Dynamic nature of JS
- Complex and large top-level statement
- Abstract assembly constraint



JS Type CodeAlchemist handles

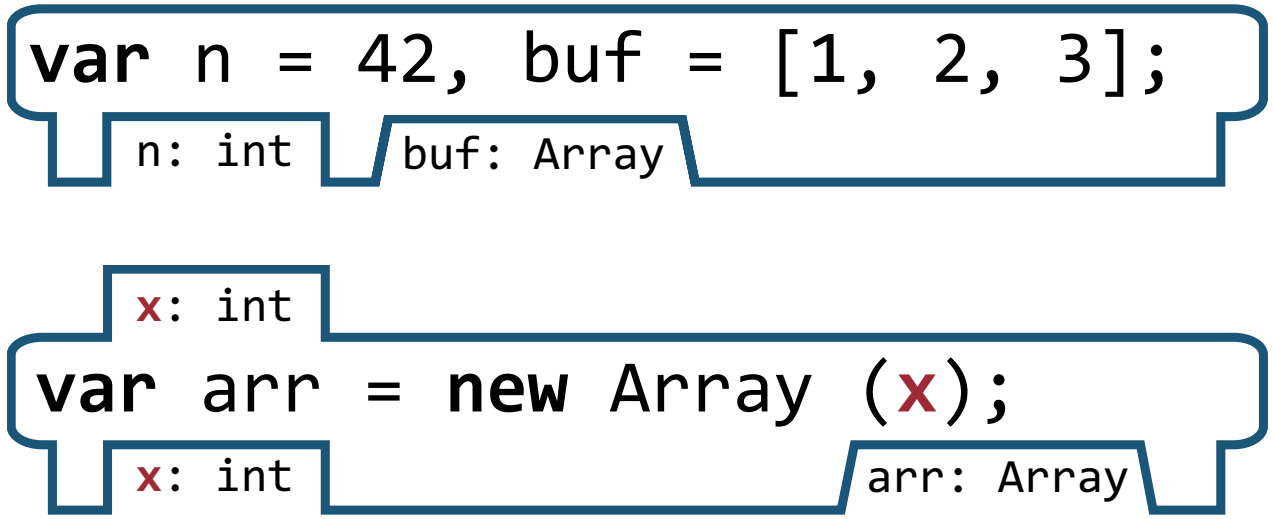
- Primitive types
  - Undefined, Null, String, Boolean, Symbol, Number, Object
- Built-in types
  - Array, ArrayBuffer, Function, …
  - Depend on JS engine

vs. State-of-the-Arts (in Previous Ver.)

- Ran 24 hours for ChakraCore 1.7.6 (Jan. 9th, 2018)
- jsfunfuzz: the latest version before Jan. 9th, 2018
- Seeds: JS snippets before Jan. 9th, 2018

	CodeAlchemist	jsfunfuzz	IFuzzer
# of Unique Crashes	7	3	0
# of Known CVEs	1	1	0

Semantics-Aware Assembly



Semantics-Aware Assembly