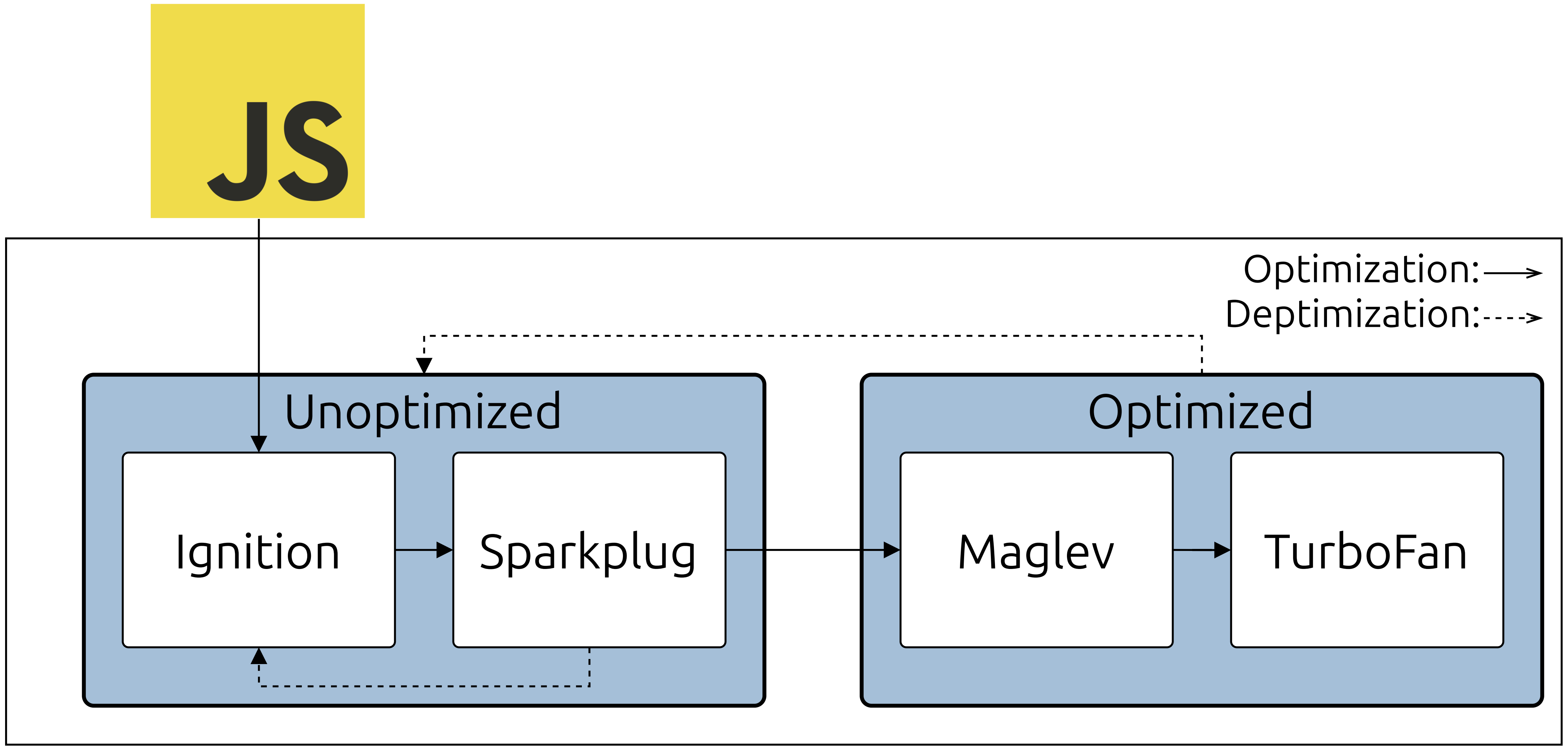
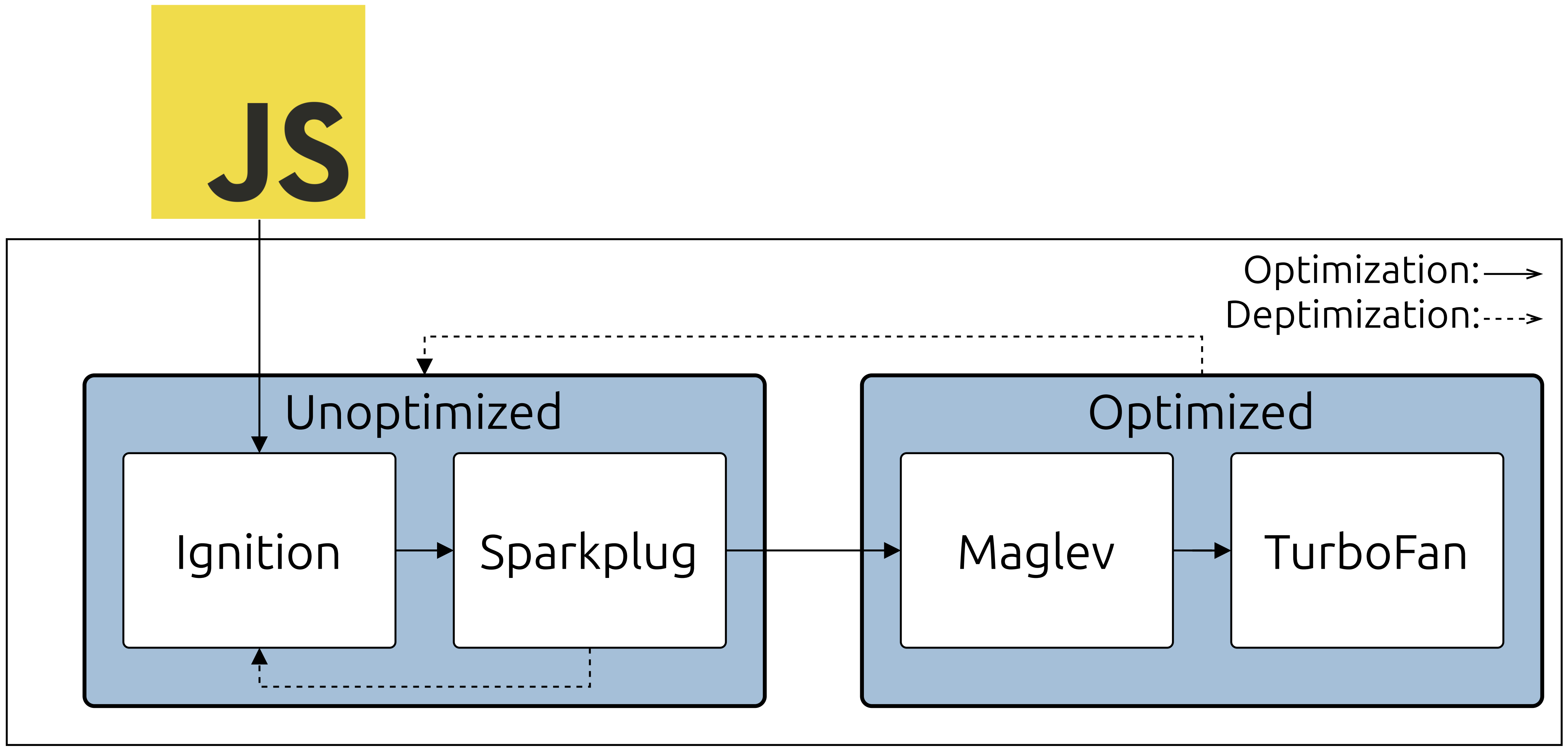
DUMPLING: FINE-GRAINED DIFFERENTIAL JAVASCRIPT ENGINE FUZZING

Liam Wachter	Asymmetric Research
Julian Gremminger	KIT
Christian Wressnegger	EPFL
Mathias Payer	RUB
Flavio Toffalini	

V8 EXECUTION TIERS



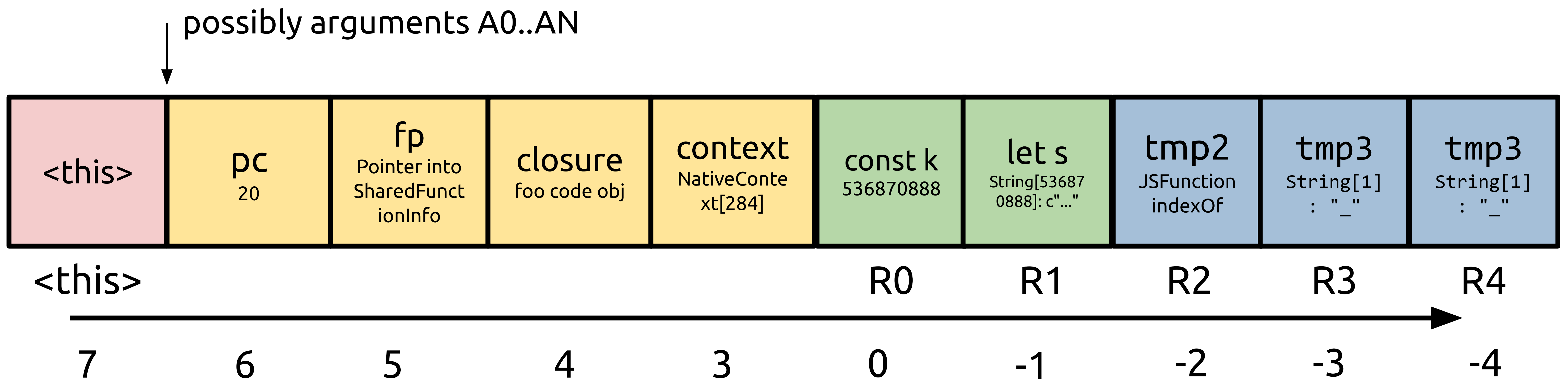
V8 EXECUTION TIERS



Even confusing -0.0 with +0.0 is enough for RCE [Röt18]

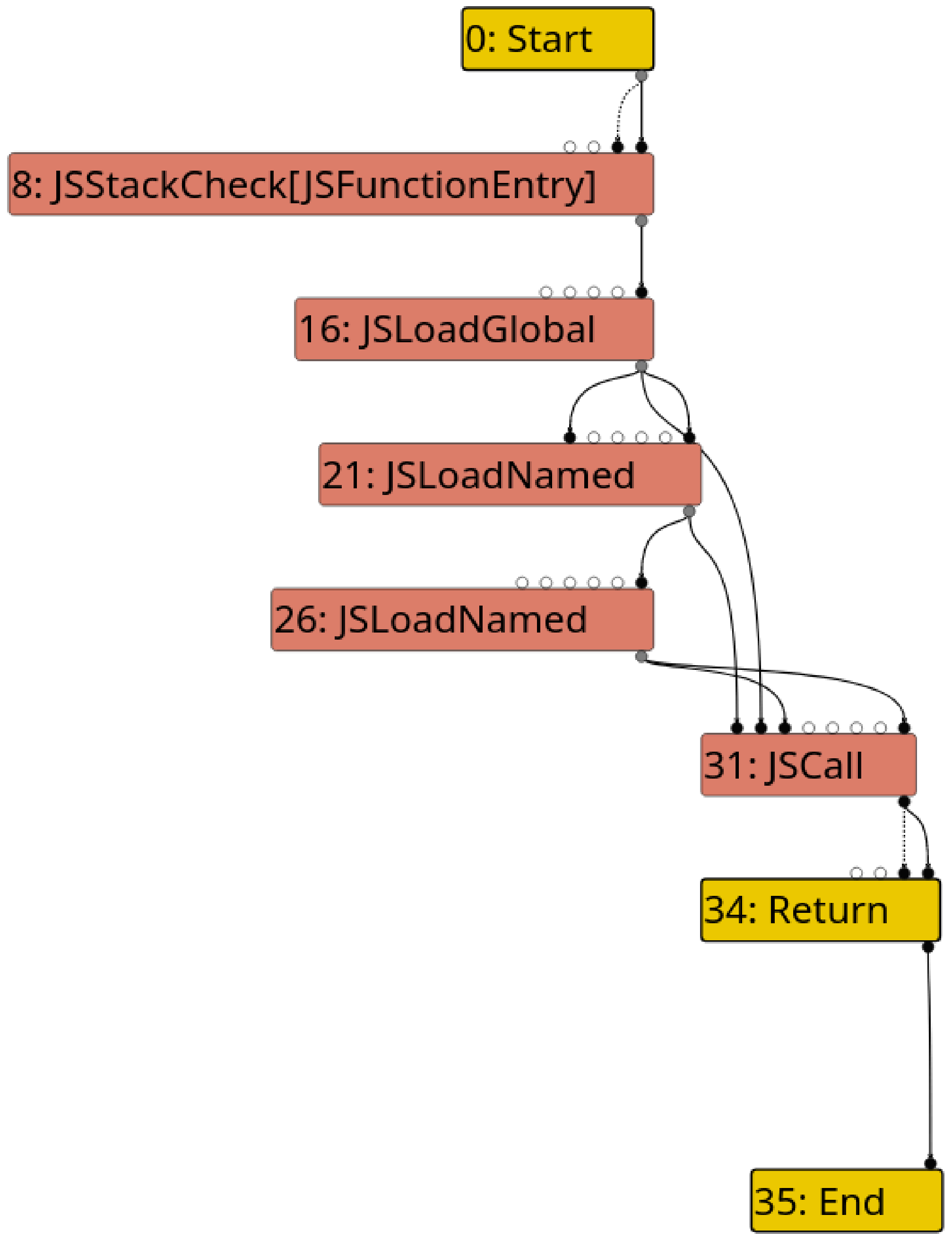
VM STATE

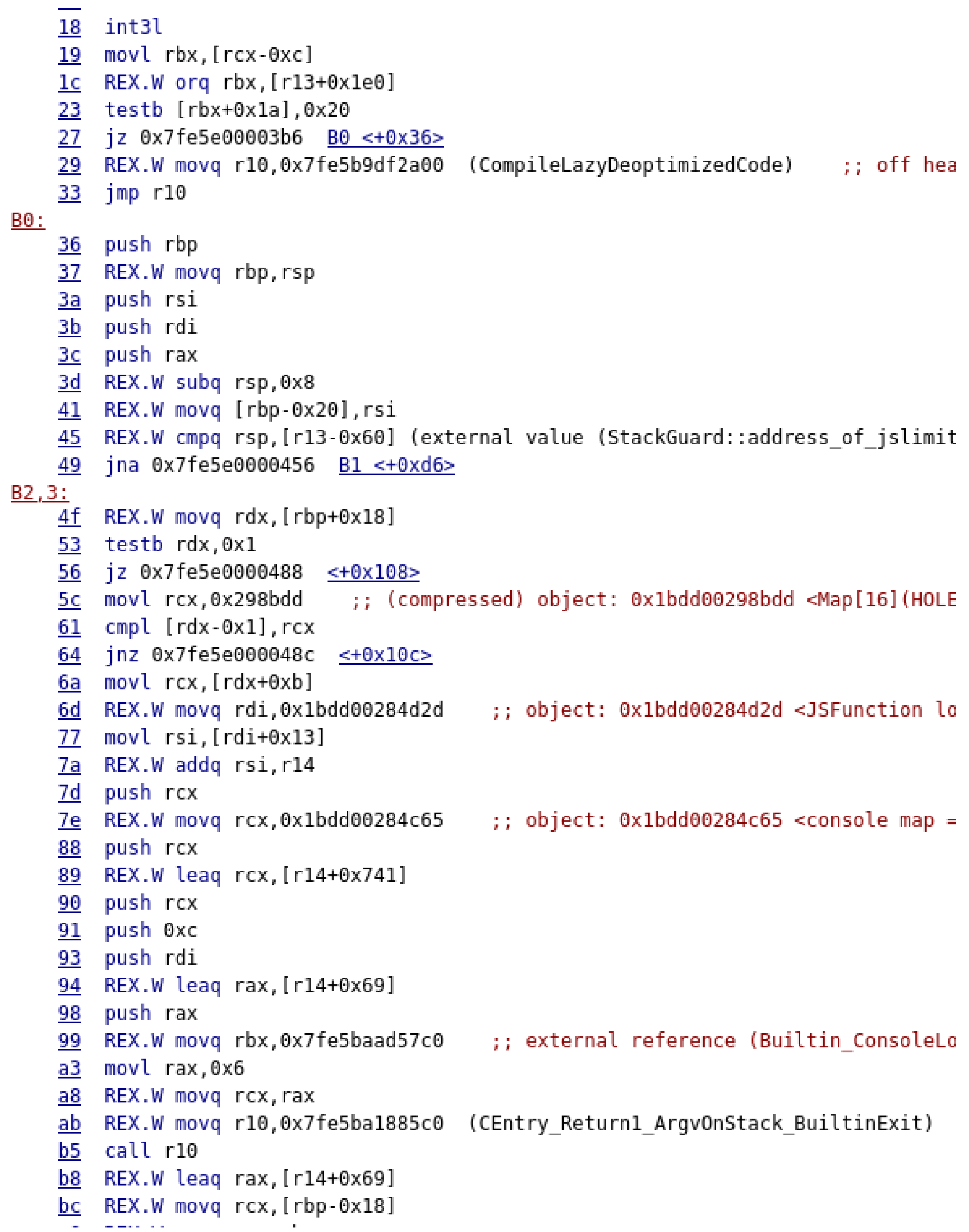
```
 0 : 01 0d e8 ff ff 1f LdaSmi.ExtraWide [536870888]
 6 : c5                Star0
 7 : 13 00             LdaConstant [0]
 9 : c2                Star3
10 : 2d f6 01 00       GetNamedProperty r3, [1], [0]
14 : c3                Star2
15 : 5e f7 f6 f9 02    CallProperty1 r2, r3, r0, [2]
20 : c4                Star1
21 : 2d f8 02 04       GetNamedProperty r1, [2], [4]
25 : c3                Star2
26 : 13 03             LdaConstant [3]
28 : c1                Star4
29 : 5f f7 f8 f5 f9 06 CallProperty2 r2, r1, r4, r0, [6]
35 : aa                Return
```



JIT COMPILATION

```
LdaConstant [0]
Star2
Mov <closure>, r3
CallRuntime [DeclareGlobals], r2-r3
LdaZero
Star1
LdaUndefined
Star0
LdaSmi.Wide [10000]
TestLessThan r1, [0]
JumpIfFalse [31] (0x4ea00040085 @ 53)
LdaGlobal [1], [1]
Star2
CreateObjectLiteral [2], [3], #41
Star3
Ldar r1
DefineNamedOwnProperty r3, [3], [4]
CallUndefinedReceiver1 r2, r3, [6]
Star0
Ldar r1
Inc [8]
Star1
JumpLoop [34], [0], [9] (0x4ea0004005f @ 15)
Ldar r0
Return
```





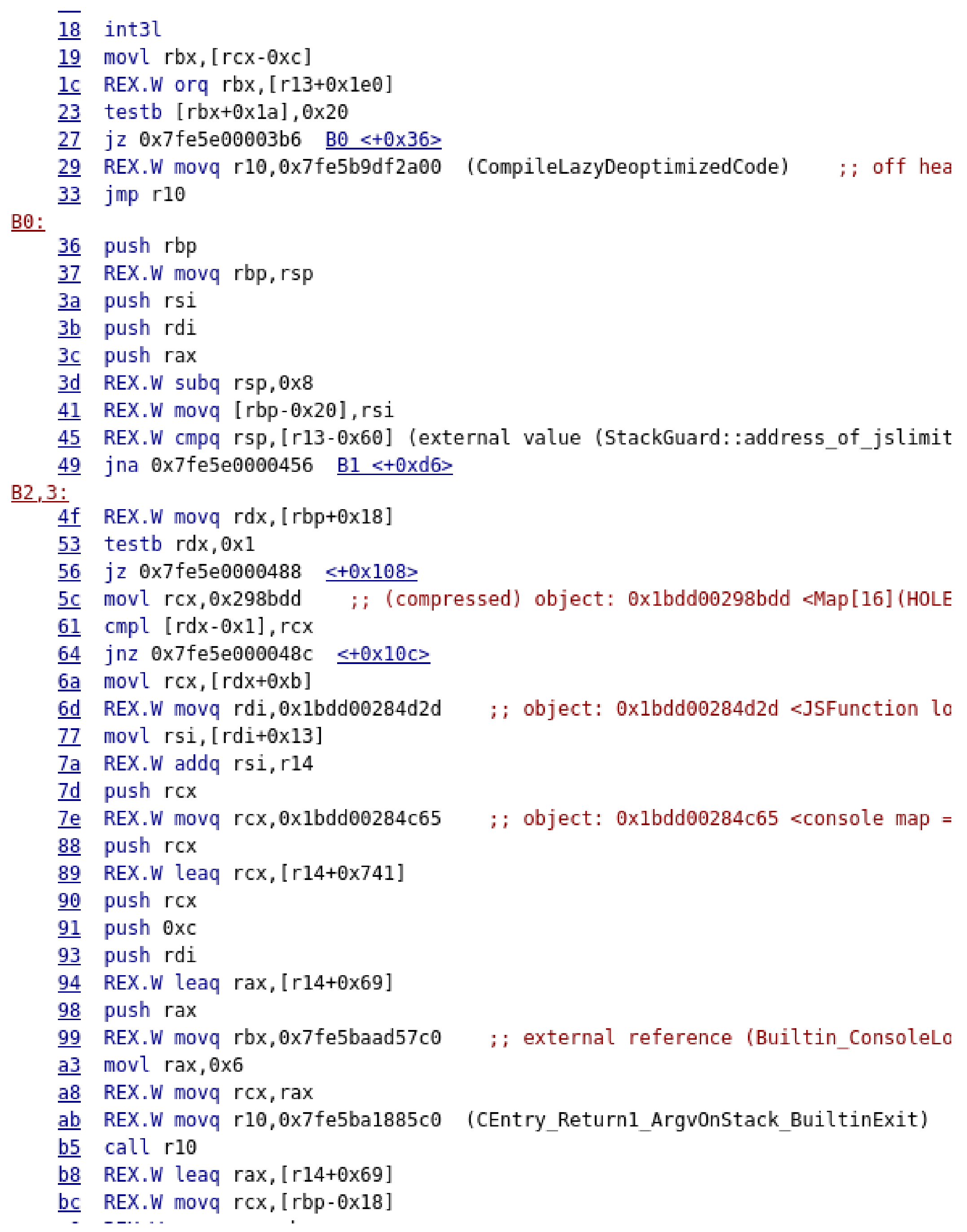
```
    18  int3l
    19  movl rbx,[rcx-0xc]
    1c  REX.W orq rbx,[r13+0x1e0]
    23  testb [rbx+0x1a],0x20
    27  jz 0x7fe5e00003b6  B0 <+0x36>
    29  REX.W movq r10,0x7fe5b9df2a00  (CompileLazyDeoptimizedCode)    ;; off hea
    33  jmp r10
B0:
    36  push rbp
    37  REX.W movq rbp,rsp
    3a  push rsi
    3b  push rdi
    3c  push rax
    3d  REX.W subq rsp,0x8
    41  REX.W movq [rbp-0x20],rsi
    45  REX.W cmpq rsp,[r13-0x60] (external value (StackGuard::address_of_jslimit
    49  jna 0x7fe5e0000456  B1 <+0xd6>
B2,3:
    4f  REX.W movq rdx,[rbp+0x18]
    53  testb rdx,0x1
    56  jz 0x7fe5e0000488  <+0x108>
    5c  movl rcx,0x298bdd    ;; (compressed) object: 0x1bdd00298bdd <Map[16](HOLE
    61  cmpl [rdx-0x1],rcx
    64  jnz 0x7fe5e000048c  <+0x10c>
    6a  movl rcx,[rdx+0xb]
    6d  REX.W movq rdi,0x1bdd00284d2d    ;; object: 0x1bdd00284d2d <JSFunction lo
    77  movl rsi,[rdi+0x13]
    7a  REX.W addq rsi,r14
    7d  push rcx
    7e  REX.W movq rcx,0x1bdd00284c65    ;; object: 0x1bdd00284c65 <console map =
    88  push rcx
    89  REX.W leaq rcx,[r14+0x741]
    90  push rcx
    91  push 0xc
    93  push rdi
    94  REX.W leaq rax,[r14+0x69]
    98  push rax
    99  REX.W movq rbx,0x7fe5baad57c0    ;; external reference (Builtin_ConsoleLo
    a3  movl rax,0x6
    a8  REX.W movq rcx,rax
    ab  REX.W movq r10,0x7fe5ba1885c0  (CEntry_Return1_ArgvOnStack_BuiltinExit)
    b5  call r10
    b8  REX.W leaq rax,[r14+0x69]
    bc  REX.W movq rcx,[rbp-0x18]
```

JIT COMPILATION

```
LdaConstant [0]
Star2
Mov <closure>, r3
CallRuntime [DeclareGlobals], r2-r3
LdaZero
Star1
LdaUndefined
Star0
LdaSmi.Wide [10000]
TestLessThan r1, [0]
JumpIfFalse [31] (0x4ea00040085 @ 53)
LdaGlobal [1], [1]
Star2
CreateObjectLiteral [2], [3], #41
Star3
Ldar r1
DefineNamedOwnProperty r3, [3], [4]
CallUndefinedReceiver1 r2, r3, [6]
Star0
Ldar r1
Inc [8]
Star1
JumpLoop [34], [0], [9] (0x4ea0004005f @ 15)
Ldar r0
Return
```

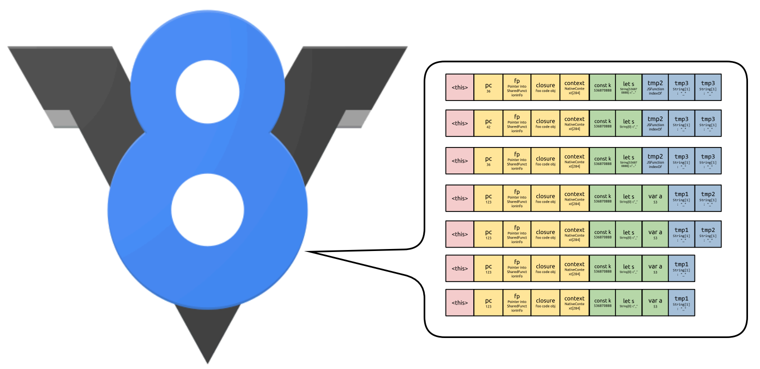
⟷

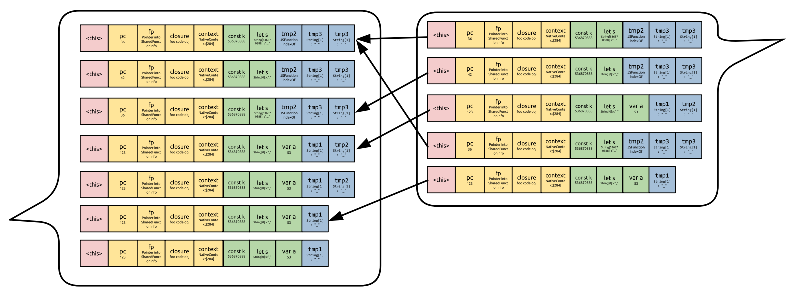
Compare VM states from unoptimized execution (left) to optimized execution (right).

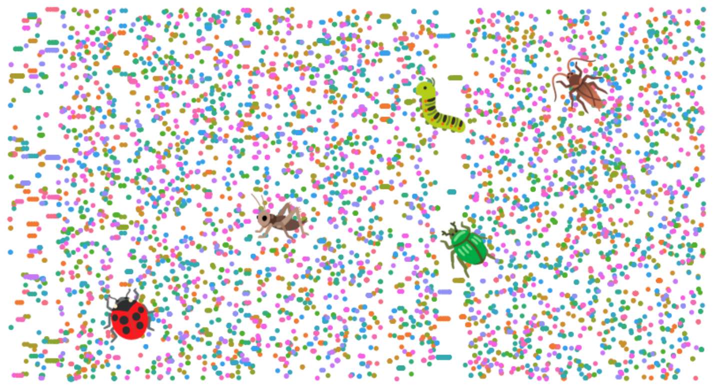


```
    18  int3l
    19  movl rbx,[rcx-0xc]
    1c  REX.W orq rbx,[r13+0x1e0]
    23  testb [rbx+0x1a],0x20
    27  jz 0x7fe5e00003b6  B0 <+0x36>
    29  REX.W movq r10,0x7fe5b9df2a00  (CompileLazyDeoptimizedCode)    ;; off hea
    33  jmp r10
B0:
    36  push rbp
    37  REX.W movq rbp,rsp
    3a  push rsi
    3b  push rdi
    3c  push rax
    3d  REX.W subq rsp,0x8
    41  REX.W movq [rbp-0x20],rsi
    45  REX.W cmpq rsp,[r13-0x60] (external value (StackGuard::address_of_jslimit
    49  jna 0x7fe5e0000456  B1 <+0xd6>
B2,3:
    4f  REX.W movq rdx,[rbp+0x18]
    53  testb rdx,0x1
    56  jz 0x7fe5e0000488  <+0x108>
    5c  movl rcx,0x298bdd    ;; (compressed) object: 0x1bdd00298bdd <Map[16](HOLE
    61  cmpl [rdx-0x1],rcx
    64  jnz 0x7fe5e000048c  <+0x10c>
    6a  movl rcx,[rdx+0xb]
    6d  REX.W movq rdi,0x1bdd00284d2d    ;; object: 0x1bdd00284d2d <JSFunction lo
    77  movl rsi,[rdi+0x13]
    7a  REX.W addq rsi,r14
    7d  push rcx
    7e  REX.W movq rcx,0x1bdd00284c65    ;; object: 0x1bdd00284c65 <console map =
    88  push rcx
    89  REX.W leaq rcx,[r14+0x741]
    90  push rcx
    91  push 0xc
    93  push rdi
    94  REX.W leaq rax,[r14+0x69]
    98  push rax
    99  REX.W movq rbx,0x7fe5baad57c0    ;; external reference (Builtin_ConsoleLo
    a3  movl rax,0x6
    a8  REX.W movq rcx,rax
    ab  REX.W movq r10,0x7fe5ba1885c0  (CEntry_Return1_ArgvOnStack_BuiltinExit)
    b5  call r10
    b8  REX.W leaq rax,[r14+0x69]
    bc  REX.W movq rcx,[rbp-0x18]
```

OVERVIEW







TRACES

Execution traces even during JIT

MATCHING

Matching algorithm to compare traces

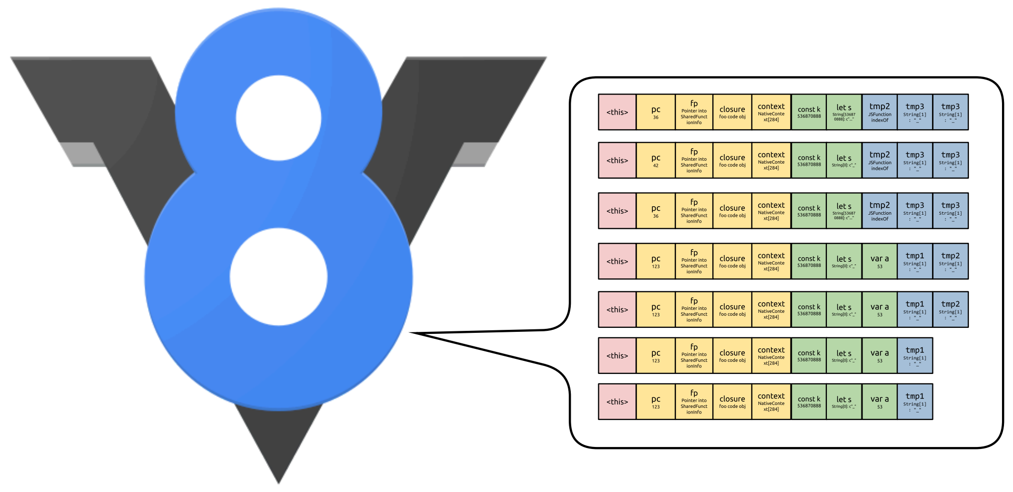
DUMPLING

Differential Fuzzer using our bug oracle

V8 BUGS

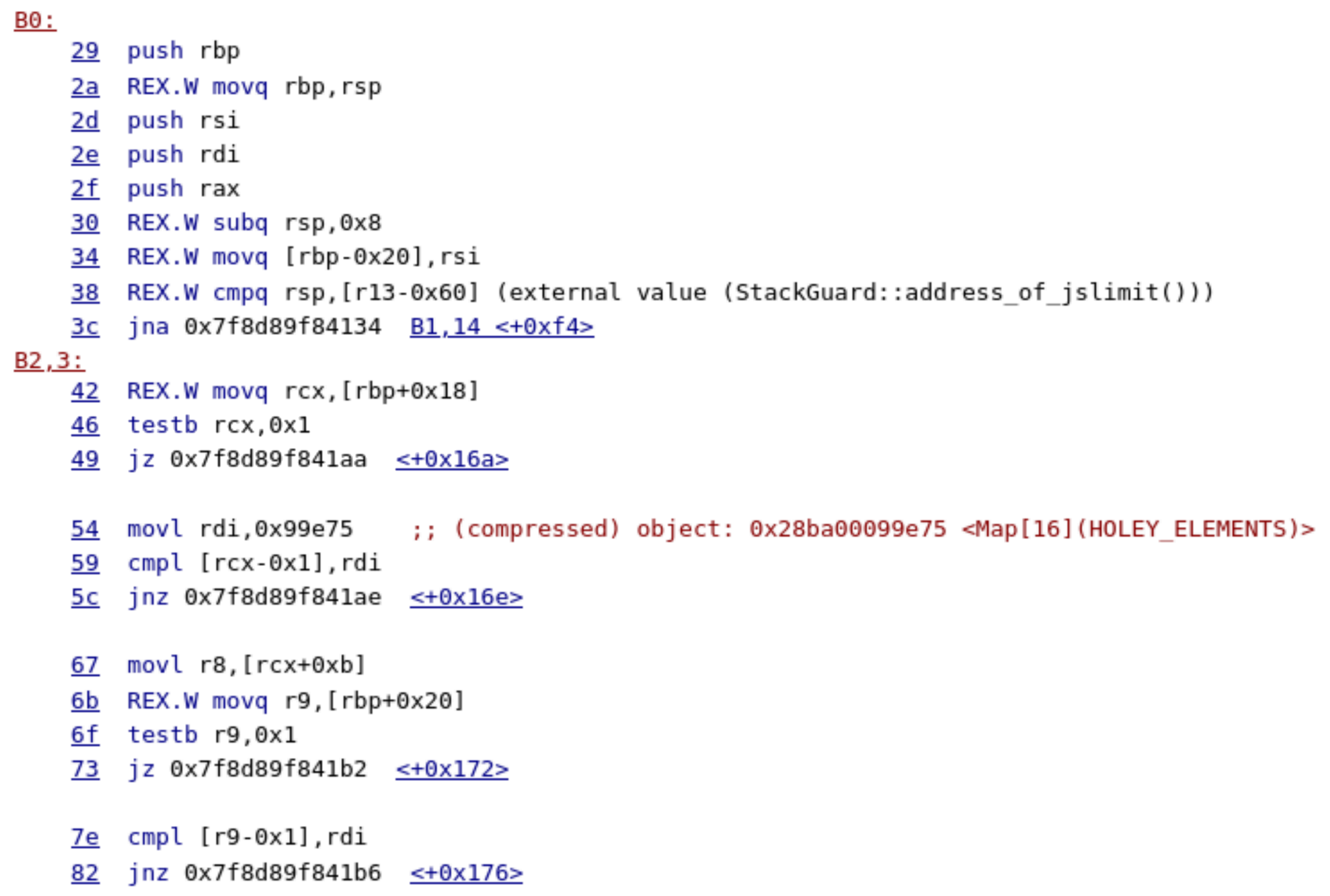
Evaluation and 8 new V8 bugs

STATE EXTRACTION



STATE EXTRACTION: JIT

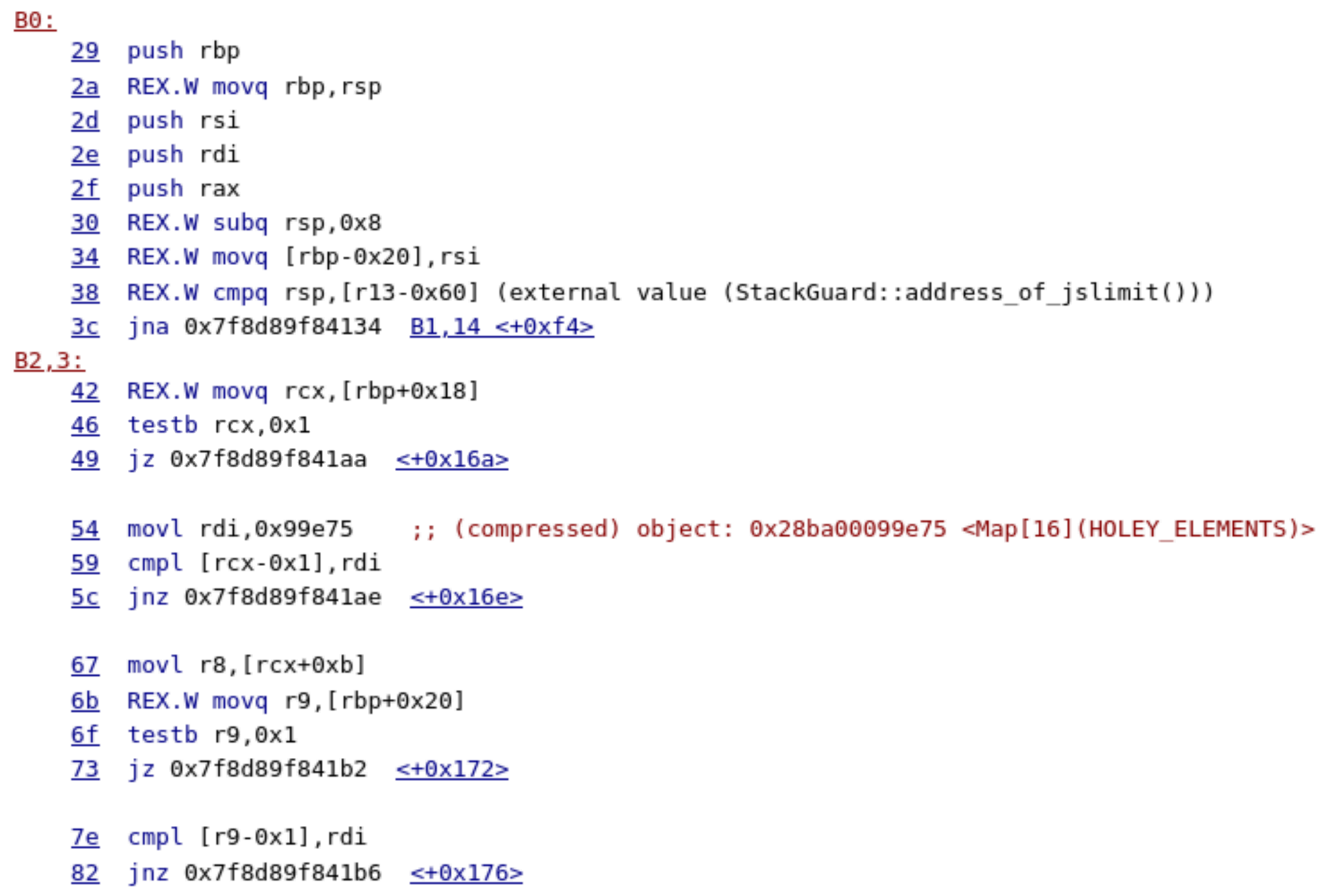
- State is spread accross machine registers and stack



```
B0:
    29  push rbp
    2a  REX.W movq rbp,rsp
    2d  push rsi
    2e  push rdi
    2f  push rax
    30  REX.W subq rsp,0x8
    34  REX.W movq [rbp-0x20],rsi
    38  REX.W cmpq rsp,[r13-0x60] (external value (StackGuard::address_of_jslimit()))
    3c  jna 0x7f8d89f84134  B1,14 <+0xf4>
B2,3:
    42  REX.W movq rcx,[rbp+0x18]
    46  testb rcx,0x1
    49  jz 0x7f8d89f841aa  <+0x16a>

    54  movl rdi,0x99e75    ;; (compressed) object: 0x28ba00099e75 <Map[16](HOLEY_ELEMENTS)>
    59  cmpl [rcx-0x1],rdi
    5c  jnz 0x7f8d89f841ae  <+0x16e>

    67  movl r8,[rcx+0xb]
    6b  REX.W movq r9,[rbp+0x20]
    6f  testb r9,0x1
    73  jz 0x7f8d89f841b2  <+0x172>

    7e  cmpl [r9-0x1],rdi
    82  jnz 0x7f8d89f841b6  <+0x176>
```

STATE EXTRACTION: JIT

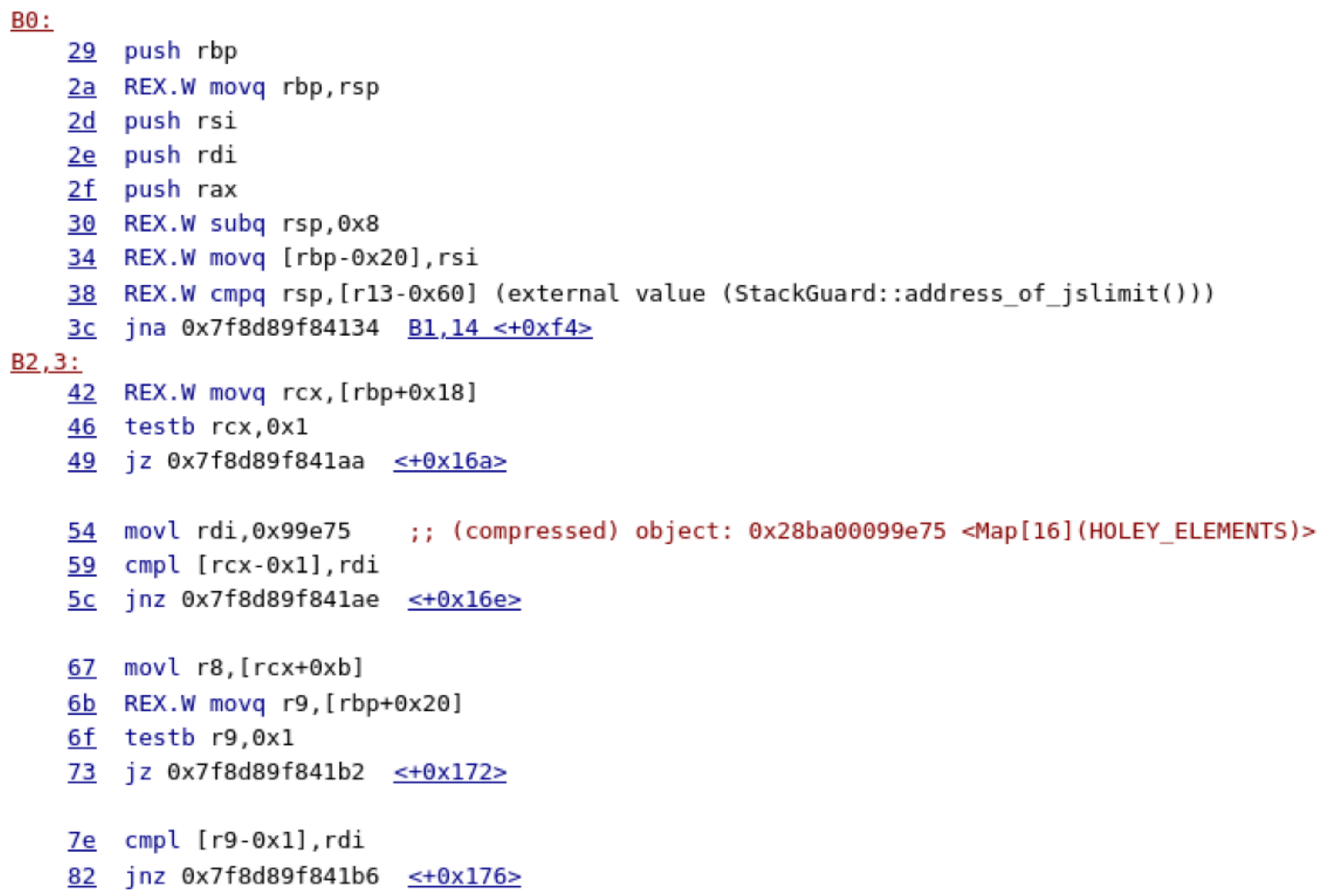
- State is spread accross machine registers and stack
- **How** do we get back to state comparable to interpreter execution?
- **Where** is state extraction possible?



```
B0:
    29  push rbp
    2a  REX.W movq rbp,rsp
    2d  push rsi
    2e  push rdi
    2f  push rax
    30  REX.W subq rsp,0x8
    34  REX.W movq [rbp-0x20],rsi
    38  REX.W cmpq rsp,[r13-0x60] (external value (StackGuard::address_of_jslimit()))
    3c  jna 0x7f8d89f84134  B1,14 <+0xf4>
B2,3:
    42  REX.W movq rcx,[rbp+0x18]
    46  testb rcx,0x1
    49  jz 0x7f8d89f841aa  <+0x16a>

    54  movl rdi,0x99e75    ;; (compressed) object: 0x28ba00099e75 <Map[16](HOLEY_ELEMENTS)>
    59  cmpl [rcx-0x1],rdi
    5c  jnz 0x7f8d89f841ae  <+0x16e>

    67  movl r8,[rcx+0xb]
    6b  REX.W movq r9,[rbp+0x20]
    6f  testb r9,0x1
    73  jz 0x7f8d89f841b2  <+0x172>

    7e  cmpl [r9-0x1],rdi
    82  jnz 0x7f8d89f841b6  <+0x176>
```

STATE EXTRACTION: JIT

- State is spread accross machine registers and stack
- **How** do we get back to state comparable to interpreter execution?
- **Where** is state extraction possible?
- No influence on JS execution semantics and JIT compiler optimizations

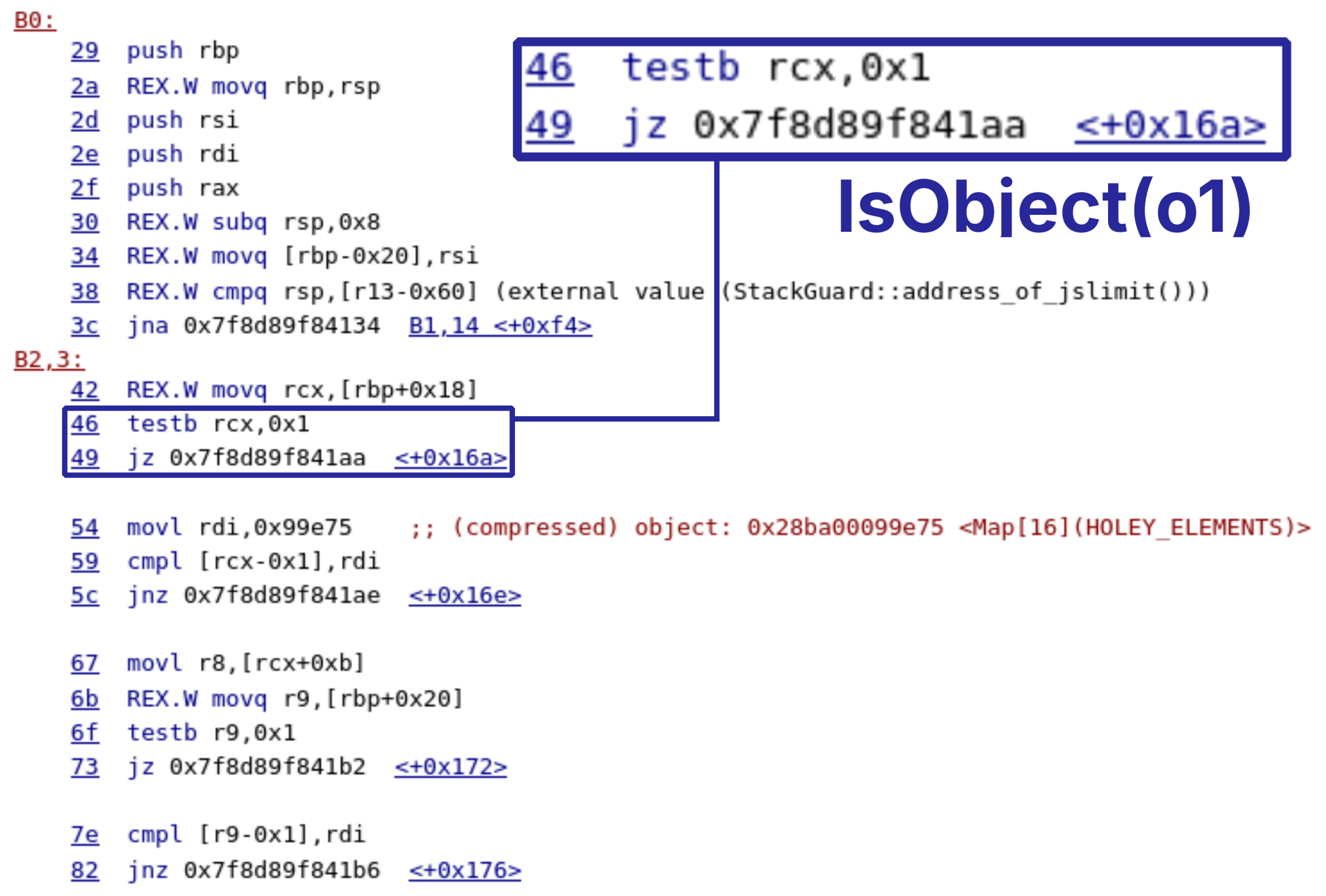


```
B0:
    29  push rbp
    2a  REX.W movq rbp,rsp
    2d  push rsi
    2e  push rdi
    2f  push rax
    30  REX.W subq rsp,0x8
    34  REX.W movq [rbp-0x20],rsi
    38  REX.W cmpq rsp,[r13-0x60] (external value (StackGuard::address_of_jslimit()))
    3c  jna 0x7f8d89f84134  B1,14 <+0xf4>
B2,3:
    42  REX.W movq rcx,[rbp+0x18]
    46  testb rcx,0x1
    49  jz 0x7f8d89f841aa  <+0x16a>

    54  movl rdi,0x99e75    ;; (compressed) object: 0x28ba00099e75 <Map[16](HOLEY_ELEMENTS)>
    59  cmpl [rcx-0x1],rdi
    5c  jnz 0x7f8d89f841ae  <+0x16e>

    67  movl r8,[rcx+0xb]
    6b  REX.W movq r9,[rbp+0x20]
    6f  testb r9,0x1
    73  jz 0x7f8d89f841b2  <+0x172>

    7e  cmpl [r9-0x1],rdi
    82  jnz 0x7f8d89f841b6  <+0x176>
```

DEOPTIMIZATION POINTS

```
function f(o1, o2) {
      return o1.a * o2.a;
  }
```

- Deopt points guard usage of specualtive assumption
- JIT tracks context to restore VM state at deopt points



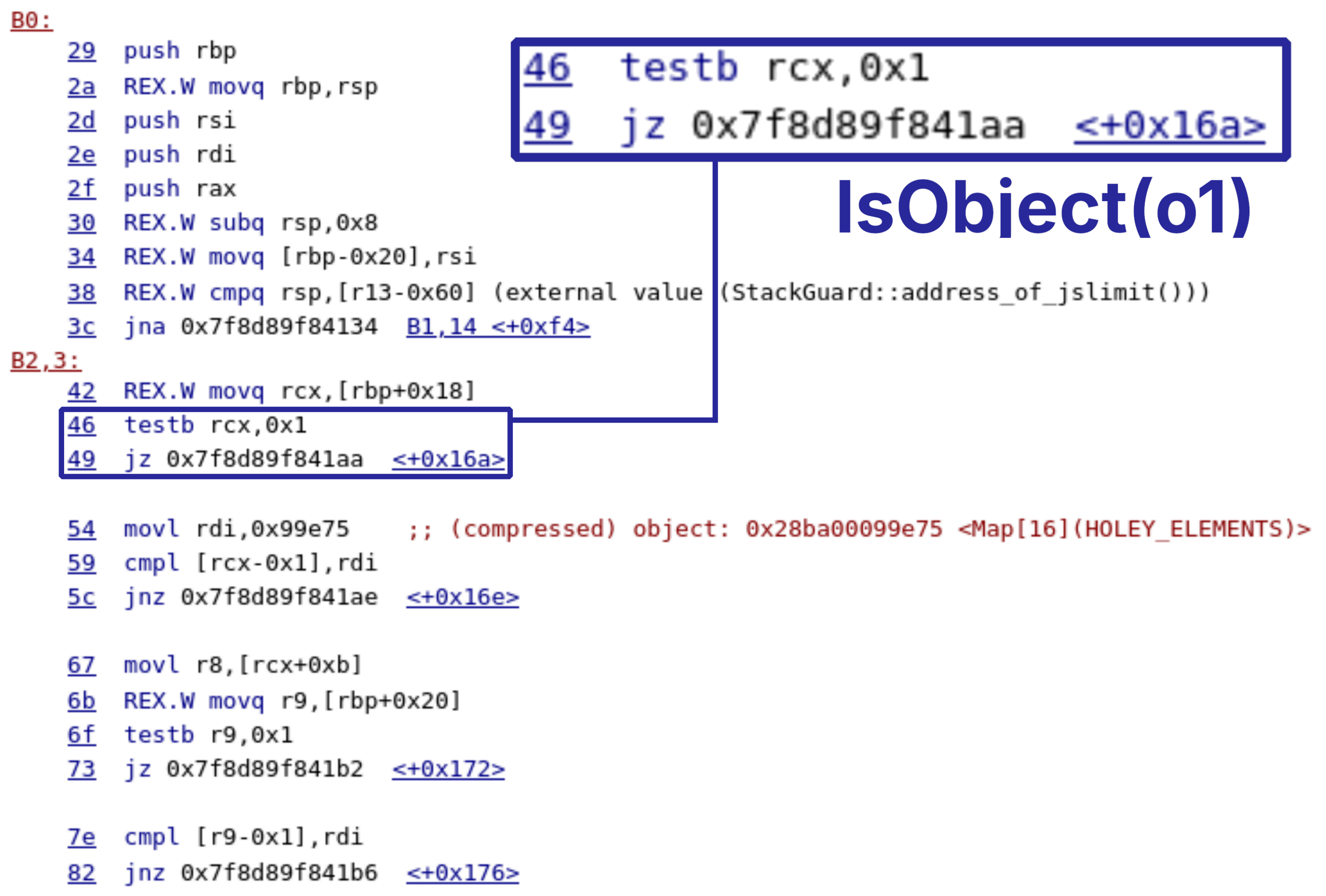
```
46   testb rcx,0x1
49   jz 0x7f8d89f841aa   <+0x16a>
```

```
B0:
    29  push rbp
    2a  REX.W movq rbp,rsp
    2d  push rsi
    2e  push rdi
    2f  push rax
    30  REX.W subq rsp,0x8
    34  REX.W movq [rbp-0x20],rsi
    38  REX.W cmpq rsp,[r13-0x60] (external value (StackGuard::address_of_jslimit()))
    3c  jna 0x7f8d89f84134  B1,14 <+0xf4>
B2,3:
    42  REX.W movq rcx,[rbp+0x18]
    46  testb rcx,0x1
    49  jz 0x7f8d89f841aa  <+0x16a>

    54  movl rdi,0x99e75    ;; (compressed) object: 0x28ba00099e75 <Map[16](HOLEY_ELEMENTS)>
    59  cmpl [rcx-0x1],rdi
    5c  jnz 0x7f8d89f841ae  <+0x16e>

    67  movl r8,[rcx+0xb]
    6b  REX.W movq r9,[rbp+0x20]
    6f  testb r9,0x1
    73  jz 0x7f8d89f841b2  <+0x172>

    7e  cmpl [r9-0x1],rdi
    82  jnz 0x7f8d89f841b6  <+0x176>
```

DEOPTIMIZATION POINTS

```
function f(o1, o2) {
      return o1.a * o2.a;
  }
```

- Deopt points guard usage of specualtive assumption
- JIT tracks context to restore VM state at deopt points
- → Deopt points as natural probing positions for interesting state



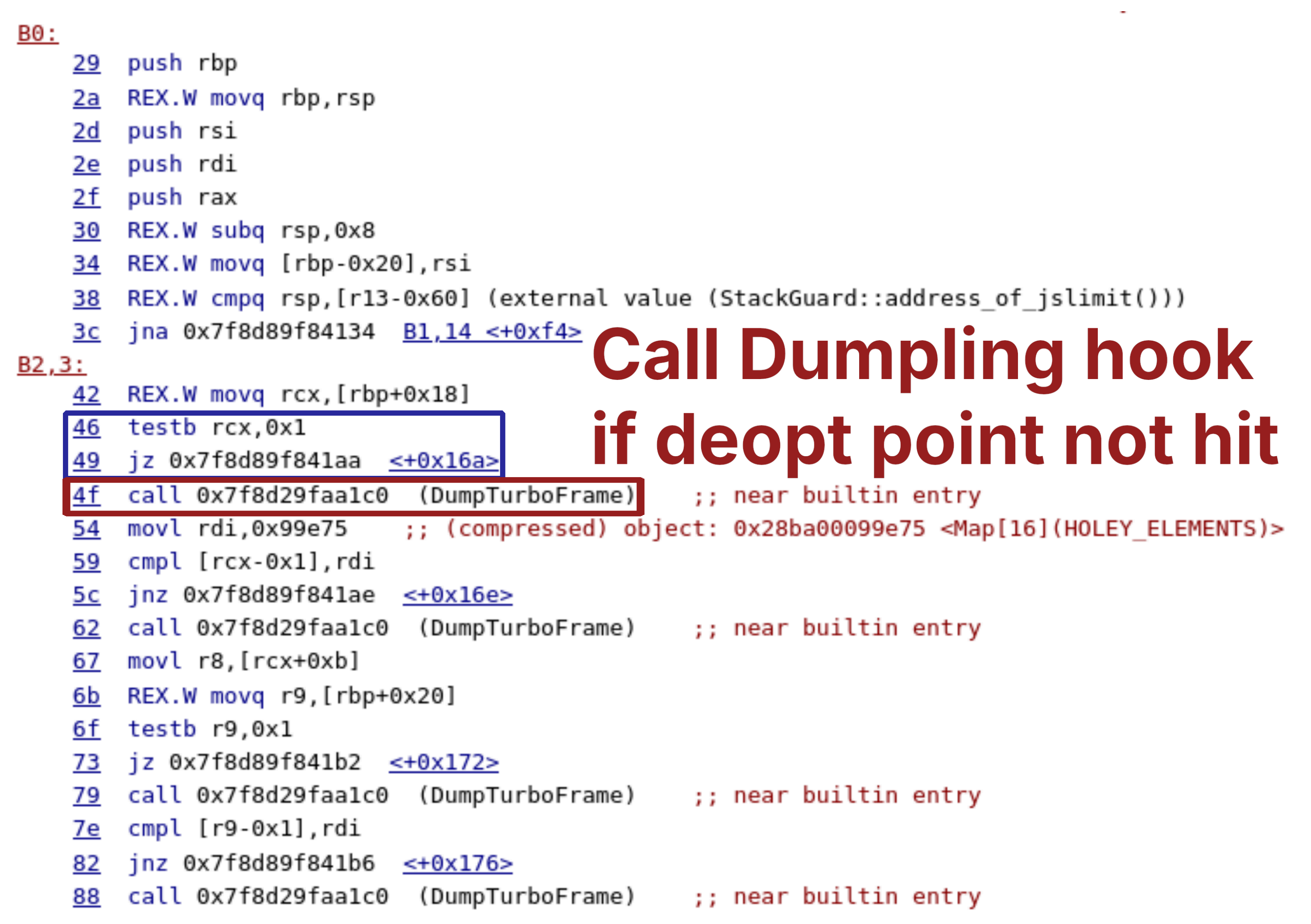
```
B0:
    29  push rbp
    2a  REX.W movq rbp,rsp
    2d  push rsi
    2e  push rdi
    2f  push rax
    30  REX.W subq rsp,0x8
    34  REX.W movq [rbp-0x20],rsi
    38  REX.W cmpq rsp,[r13-0x60] (external value (StackGuard::address_of_jslimit()))
    3c  jna 0x7f8d89f84134  B1,14 <+0xf4>
B2,3:
    42  REX.W movq rcx,[rbp+0x18]
    46  testb rcx,0x1
    49  jz 0x7f8d89f841aa  <+0x16a>

    54  movl rdi,0x99e75    ;; (compressed) object: 0x28ba00099e75 <Map[16](HOLEY_ELEMENTS)>
    59  cmpl [rcx-0x1],rdi
    5c  jnz 0x7f8d89f841ae  <+0x16e>

    67  movl r8,[rcx+0xb]
    6b  REX.W movq r9,[rbp+0x20]
    6f  testb r9,0x1
    73  jz 0x7f8d89f841b2  <+0x172>

    7e  cmpl [r9-0x1],rdi
    82  jnz 0x7f8d89f841b6  <+0x176>
```

```
46  testb rcx,0x1
49  jz 0x7f8d89f841aa  <+0x16a>
```

DUMPING DURING SPECULATIVE JIT EXECUTION

1. Save state
2. Build VM state
3. Rematerialize escaped values
4. "Dump" VM state
5. Restore state and continue JIT execution

→ partial use of existing deopt mechanism



```
B0:
    29  push rbp
    2a  REX.W movq rbp,rsp
    2d  push rsi
    2e  push rdi
    2f  push rax
    30  REX.W subq rsp,0x8
    34  REX.W movq [rbp-0x20],rsi
    38  REX.W cmpq rsp,[r13-0x60] (external value (StackGuard::address_of_jslimit()))
    3c  jna 0x7f8d89f84134  B1,14 <+0xf4>
B2,3:
    42  REX.W movq rcx,[rbp+0x18]
    46  testb rcx,0x1
    49  jz 0x7f8d89f841aa  <+0x16a>
    4f  call 0x7f8d29faa1c0  (DumpTurboFrame)    ;; near builtin entry
    54  movl rdi,0x99e75    ;; (compressed) object: 0x28ba00099e75 <Map[16](HOLEY_ELEMENTS)>
    59  cmpl [rcx-0x1],rdi
    5c  jnz 0x7f8d89f841ae  <+0x16e>
    62  call 0x7f8d29faa1c0  (DumpTurboFrame)    ;; near builtin entry
    67  movl r8,[rcx+0xb]
    6b  REX.W movq r9,[rbp+0x20]
    6f  testb r9,0x1
    73  jz 0x7f8d89f841b2  <+0x172>
    79  call 0x7f8d29faa1c0  (DumpTurboFrame)    ;; near builtin entry
    7e  cmpl [r9-0x1],rdi
    82  jnz 0x7f8d89f841b6  <+0x176>
    88  call 0x7f8d29faa1c0  (DumpTurboFrame)    ;; near builtin entry
```

STATE EXTRACTION: DUMPLING MODE - INTERPRETER

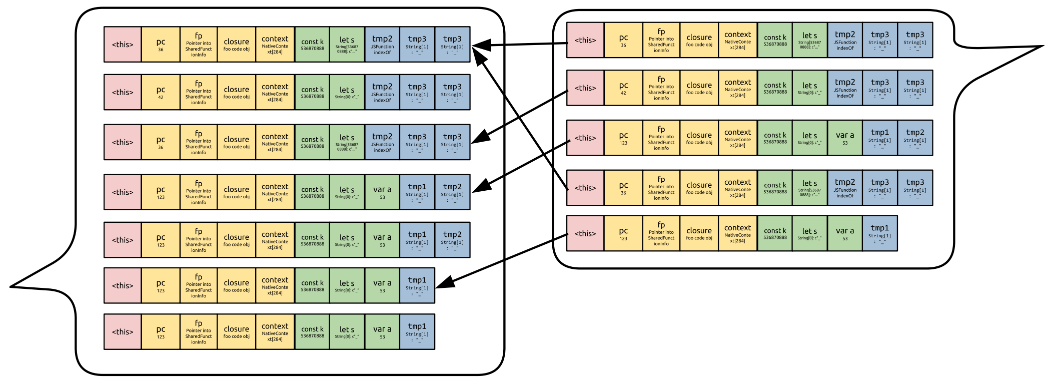
- Optimized run reports dump locations to the fuzzer
- Hook bytecode execution and extract state at those dump locations

STATE SERIALIZATION

```
-------TurboFan frame dump-------
pc: 7
acc: 13.37
a0: <Object>{
__proto__: <Class C7>{<String[1]: f>[enumerable]<JSArray>[]},
<String[1]: a>[configurable][enumerable]42(enum cache: 2),
<String[1]: f>[configurable][enumerable]13.37(enum cache: 0)
}
r0: -INFINITY
context: <ScriptContext[4]>
receiver: <JSGlobalProxy>
closure: <JSFunction f0>
Function ID: 27
```

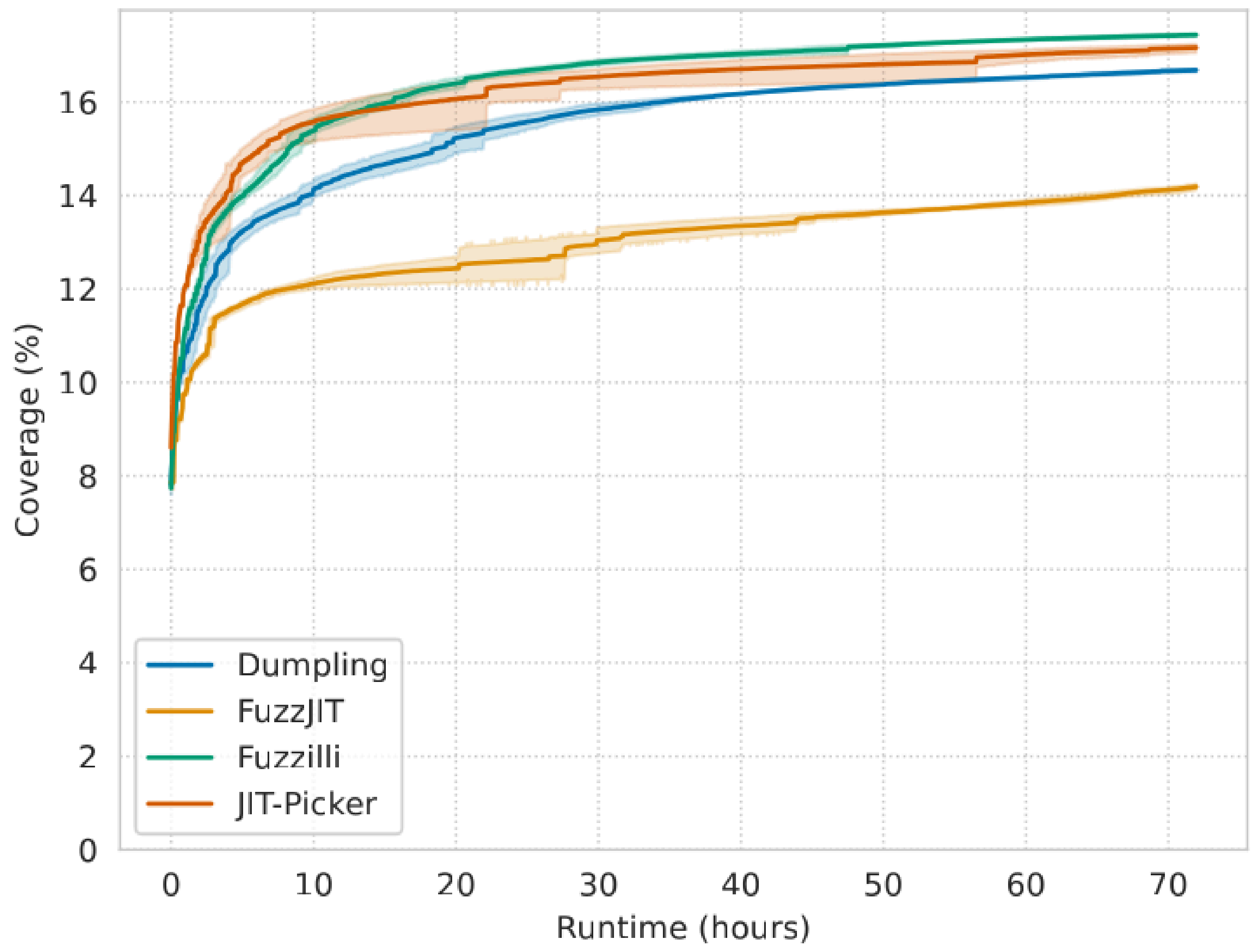
- Invariant across execution tiers
- Fine-grained and in-depth
- Concise to minimize transmission overhead

DIFFERENTIAL ORACLE



- No 1:1 mapping of dumps
- Any JIT dump must have an interpreter equivalent in the **same** function invocation

EVALUATION: OVERHEAD



Fuzzer	Fuzzilli	JIT-Picker	FuzzJIT	Dumpling
Executions	63,775,062	99,240,042	61,434,736	51,535,553

BUGS

Found 8 new V8 bugs 🎉

Bug Id	Kind	Status	Changes	By	Description
CR41488094	Diff	fixed	+28/-23	D, J	Enumerating properties eagerly, has incorrect side effect
CR335310000	Diff	fixed	+15/0	D	Property not marked as enumerable by Maglev and TurboFan
CR332745405	Diff	fixed	+5/0	D	DefineOwnProperty called the setter of an existing accessor property
CR329330868	assert	dup	N/A	D, J	array.shift does not update pointers in spill slots
CR41484971	Diff	fixed	+52/-40	D	Store inline cache handlers are incorrectly used for defining properties
V8:14605	Diff	fixed	+1/-1	D	The JumpLoop bytecode does not clobber the accumulator in all cases
CR345960102	Diff	fixed	+6/-4	D	TurboFan incorrectly optimizes 64 bit BigInt shifts
CR346086168	Diff	fixed	+109/-107	D	Overflow in BigInt64 shift optimization
V8:14556	Diff	available	N/A	D	The arguments array is handled differently in optimizing compilers
CR40945996	assert	dup	N/A	D	The profiler in Maglev interferes with deoptimization

CASE STUDY

```
function A() {
    Object.defineProperty(this, "x", { writable: true, configurable: true, value: undefined });
}

class B extends A {
    x = {};
}

for (let i = 0; i < 100; i++) {
    new B();
}
```

Here not "visible", but already detectable by Dumpling

CASE STUDY

```
function A() {
    Object.defineProperty(this, "x", { writable: true, configurable: true, value: undefined });
}

class B extends A {
    x = {};
}

for (let i = 0; i < 100; i++) {
    new B();
}
```

Here not "visible", but already detectable by Dumpling

Other fuzzers need generate something like

```
let b = new B();
console.log(b.propertyIsEnumerable("x"));
```

optimizations enabled: "`true`", optimizations disabled: "`false`"

CONCLUSION

KEY PROBLEM

Find differentials between JS engine execution tiers automatically

DUMPLING

Extract VM states during runtime and compare between JIT and interpreter

Leveraging deoptimization points, a mechanism already implemented in JS engines

RESULT

Find bugs before they become "visible"

QUESTIONS?

Find our artifact here: github.com/two-heart/dumpling-artifact-evaluation

Artifact Evaluated NDSS
Available
Functional
Reproduced

@95p@mastodon.cloud

@NearBeteigeuze

liam@seine.email

BIBLIOGRAPHY

[GSV22] Jakob Gruber, Leszek Swirski, and Toon Verwaest. Maglev. 2022. url: https://docs.google.com/document/d/13CwgSL4yawxuYg3iNlM-4ZPCB8RgJya6b8H_ E2F-Aek/ (visited on 11/28/2023).
[Röt18] Stephen Röttger. Chrome: V8: incorrect type information on Math.expm1. 2018. url: https://crbug.com/project-zero/1710 (visited on 03/18/2024).
[Flü16] Olvier Flückiger. Ignition: V8 Interpreter. 2016. url: https://docs.google.com/document/d/11T2CRex9hXxoJwbYqVQ32yIPMh0uouUZLdyrtmMoL44 (visited on 11/20/2023).