Inline Caching in JavaScriptCore

Filip Pizlo
Apple Inc.

webkit.org

https://svn.webkit.org/repository/webkit/trunk

JavaScriptCore.framework

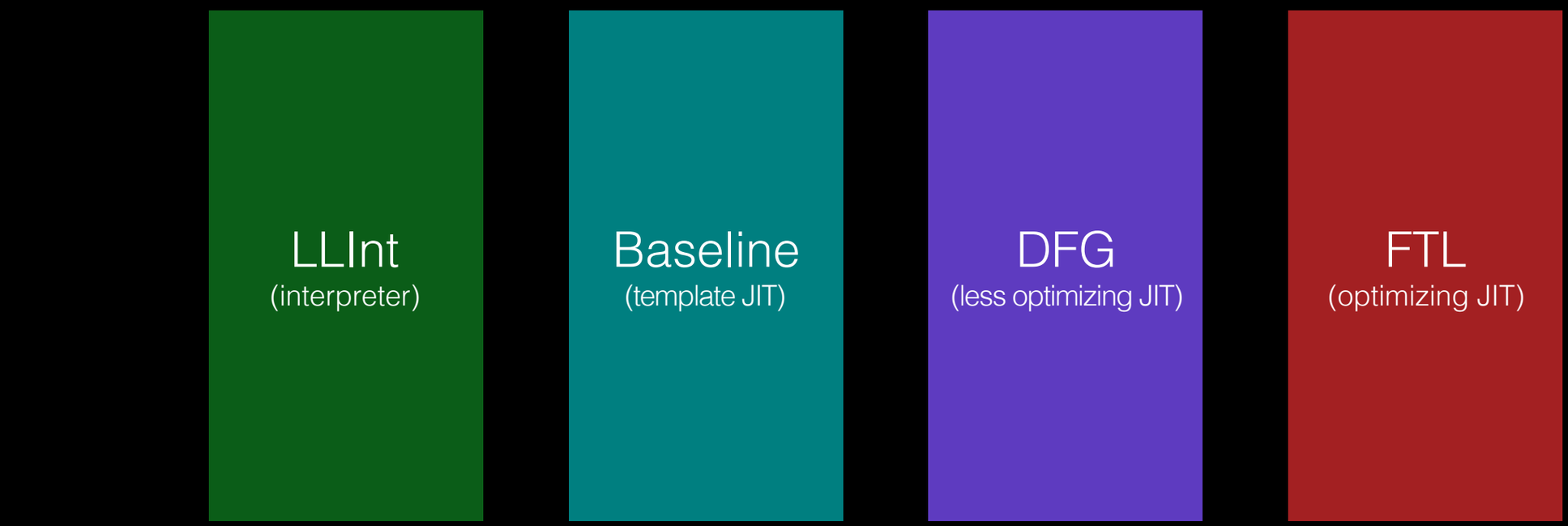
Safari

Agenda

- JavaScript execution strategy
- Focus on *property access* inline caches
  - Simple inline caching
  - Inline Caching Optimizations
  - Lots of perf data

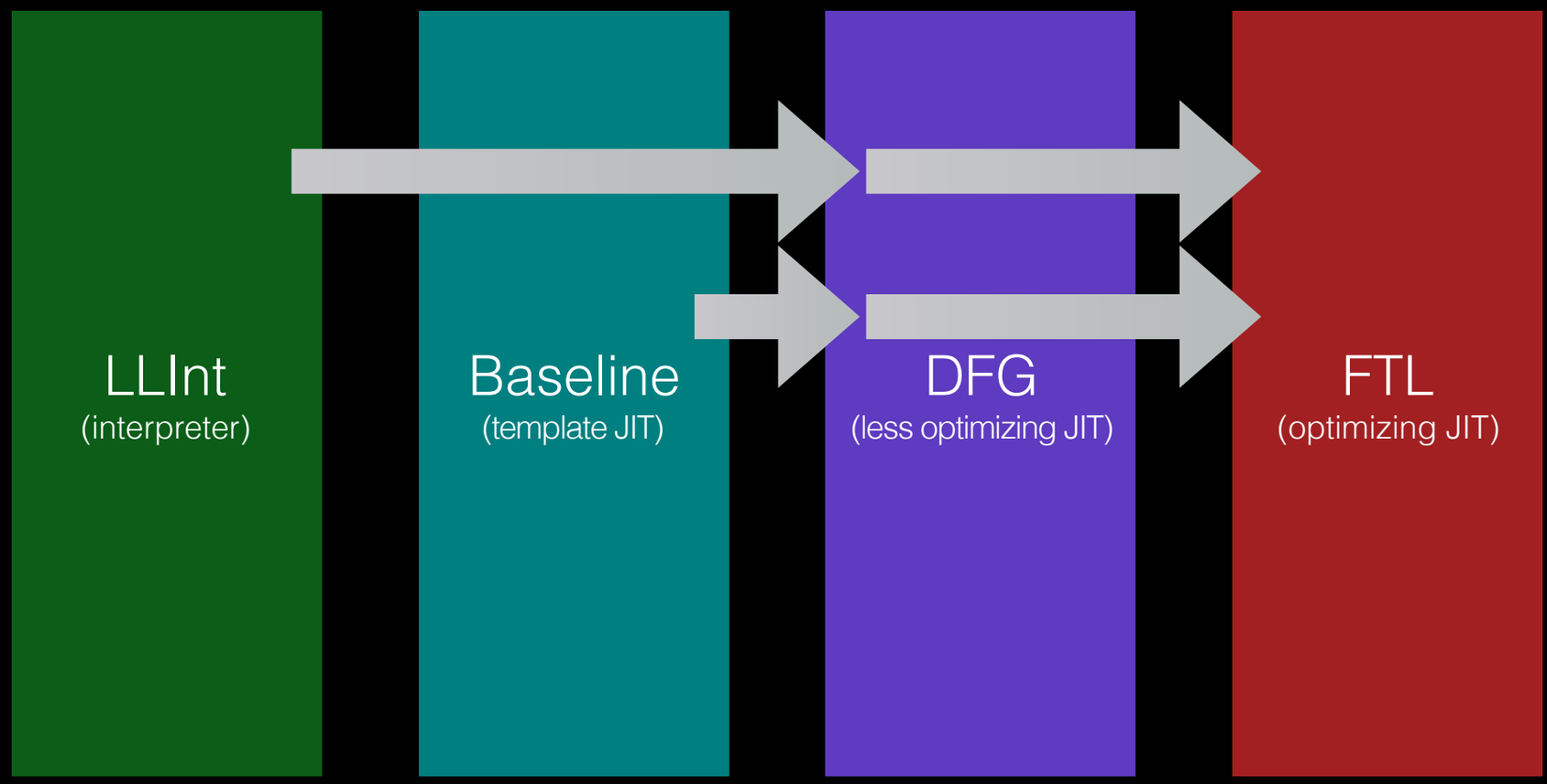
JavaScript execution strategy

Four Tiers

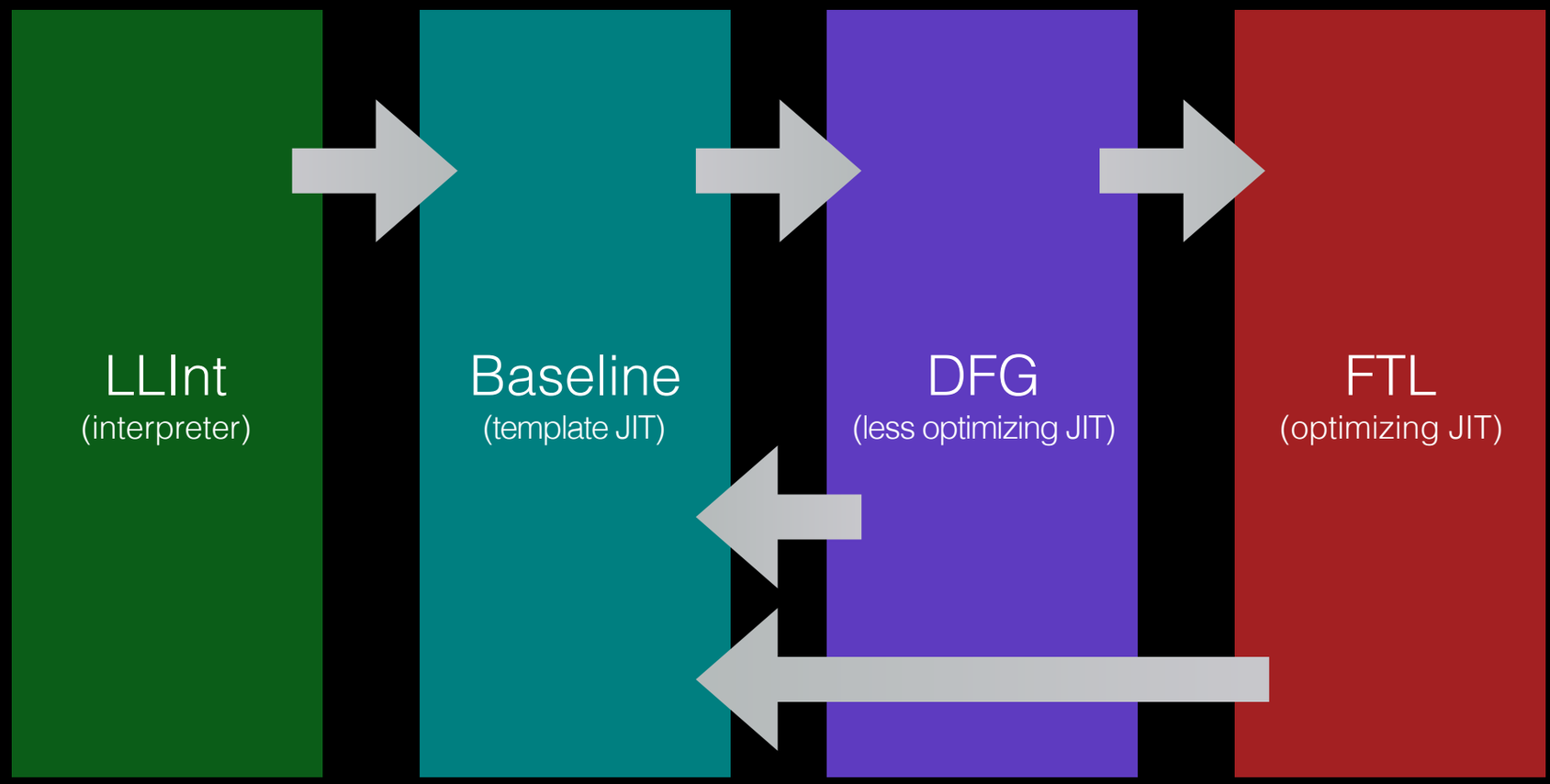

LLInt
(interpreter)
Baseline
(template JIT)
DFG
(less optimizing JIT)
FTL
(optimizing JIT)


latency
throughput

Profiling


LLInt
(interpreter)
Baseline
(template JIT)
DFG
(less optimizing JIT)
FTL
(optimizing JIT)

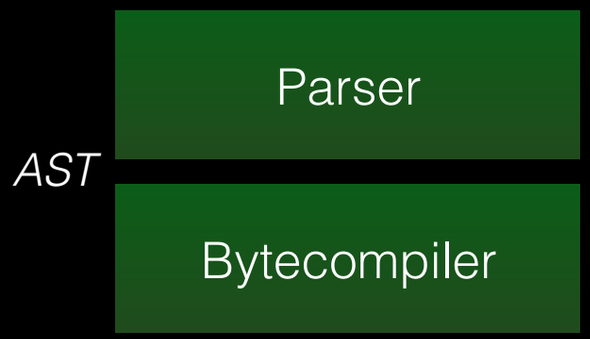
Speculation and OSR

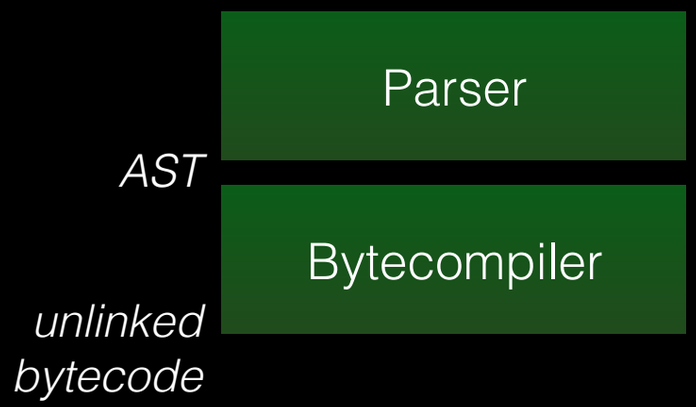


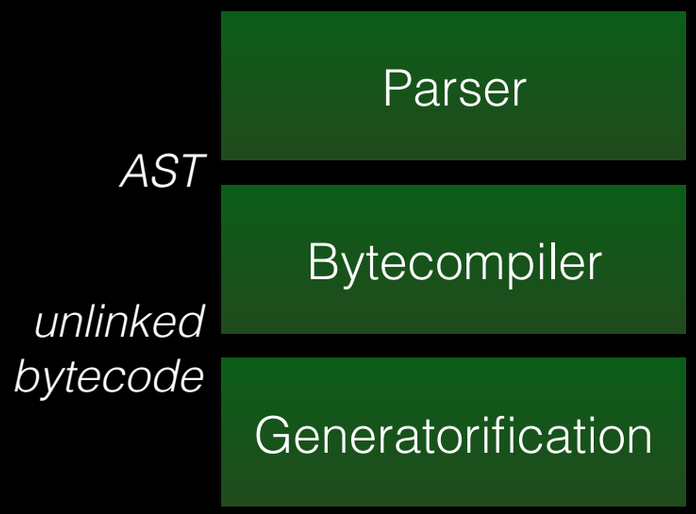
Parser

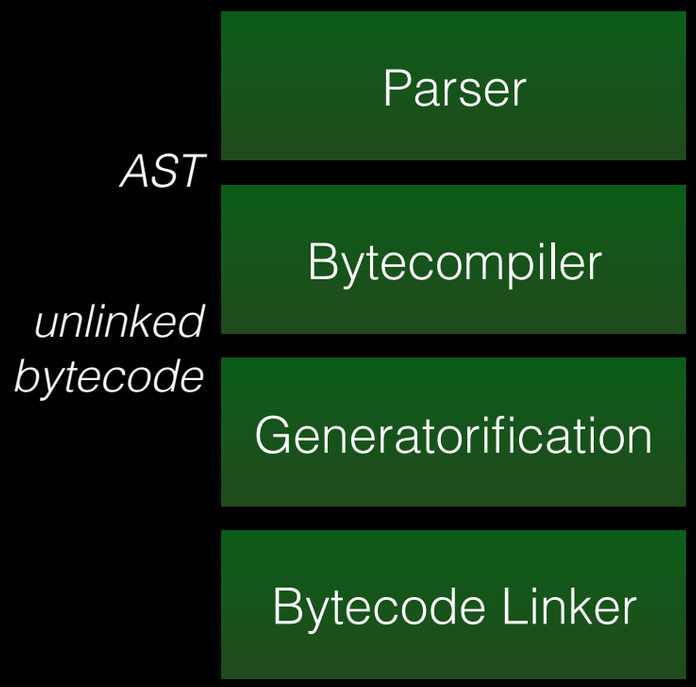
Parser

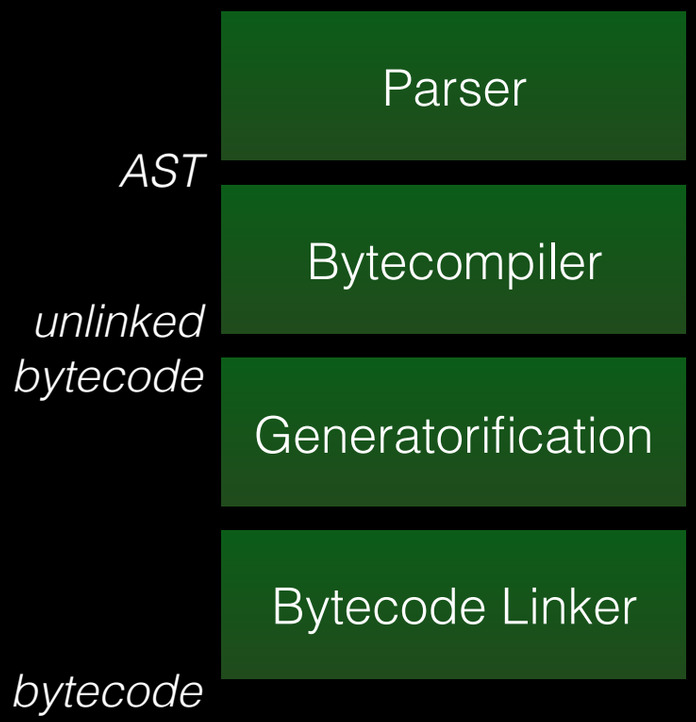
AST


Parser
AST
Bytecompiler


Parser
AST
Bytecompiler
unlinked
bytecode


Parser
AST
Bytecompiler
unlinked bytecode
Generatorification


Parser
AST
Bytecompiler
unlinked bytecode
Generatorification
Bytecode Linker


Parser
AST
Bytecompiler
unlinked bytecode
Generatorification
Bytecode Linker
bytecode

Parser

AST

Bytecompiler

unlinked bytecode

Generatorification

Bytecode Linker

bytecode

LLInt

Parser

AST

Bytecompiler

unlinked bytecode

Generatorification

Bytecode Linker

bytecode

LLInt

Bytecode Template JIT

Parser

AST

Bytecompiler

unlinked bytecode

Generatorification

Bytecode Linker

bytecode

LLInt

Bytecode Template JIT

DFG

Parser

AST

Bytecompiler

unlinked bytecode

Generatorification

Bytecode Linker

bytecode

LLInt

Bytecode Template JIT

DFG

DFG Bytecode Parser

Parser

AST

Bytecompiler

unlinked bytecode

Generatorification

Bytecode Linker

bytecode

LLInt

Bytecode Template JIT

DFG

DFG Bytecode Parser

DFG Optimizer

Parser

AST

Bytecompiler

unlinked bytecode

Generatorification

Bytecode Linker

bytecode

LLInt

Bytecode Template JIT

DFG

DFG Bytecode Parser

DFG Optimizer

DFG Backend

Parser

AST

Bytecompiler

unlinked bytecode

Generatorification

Bytecode Linker

bytecode

LLInt

Bytecode Template JIT

DFG

DFG Bytecode Parser

DFG Optimizer

DFG Backend

FTL

Parser

AST

Bytecompiler

unlinked bytecode

Generatorification

Bytecode Linker

bytecode

LLInt

Bytecode Template JIT

DFG

DFG Bytecode Parser

DFG Optimizer

DFG Backend

FTL

DFG Bytecode Parser

Parser

AST

Bytecompiler

unlinked bytecode

Generatorification

Bytecode Linker

bytecode

LLInt

Bytecode Template JIT

DFG

DFG Bytecode Parser

DFG Optimizer

DFG Backend

FTL

DFG Bytecode Parser

Extended DFG Optimizer

Parser

AST

Bytecompiler

unlinked bytecode

Generatorification

Bytecode Linker

bytecode

LLInt

Bytecode Template JIT

DFG

DFG Bytecode Parser

DFG Optimizer

DFG Backend

FTL

DFG Bytecode Parser

Extended DFG Optimizer

FTL-to-B3 lowering

Parser

AST

Bytecompiler

unlinked bytecode

Generatorification

Bytecode Linker

bytecode

LLInt

Bytecode Template JIT

DFG

DFG Bytecode Parser

DFG Optimizer

DFG Backend

FTL

DFG Bytecode Parser

Extended DFG Optimizer

FTL-to-B3 lowering

B3 Optimizer

Parser

AST

Bytecompiler

unlinked bytecode

Generatorification

Bytecode Linker

bytecode

LLInt

Bytecode Template JIT

DFG

DFG Bytecode Parser

DFG Optimizer

DFG Backend

FTL

DFG Bytecode Parser

Extended DFG Optimizer

FTL-to-B3 lowering

B3 Optimizer

Instruction Selection

Parser

AST

Bytecompiler

unlinked bytecode

Generatorification

Bytecode Linker

bytecode

LLInt

Bytecode Template JIT

DFG

DFG Bytecode Parser

DFG Optimizer

DFG Backend

FTL

DFG Bytecode Parser

Extended DFG Optimizer

FTL-to-B3 lowering

B3 Optimizer

Instruction Selection

Air Optimizer

Parser

AST

Bytecompiler

unlinked bytecode

Generatorification

Bytecode Linker

bytecode

LLInt

Bytecode Template JIT

DFG

DFG Bytecode Parser

DFG Optimizer

DFG Backend

FTL

DFG Bytecode Parser

Extended DFG Optimizer

FTL-to-B3 lowering

B3 Optimizer

Instruction Selection

Air Optimizer

Air Backend

Inline Caching

Inline Caching

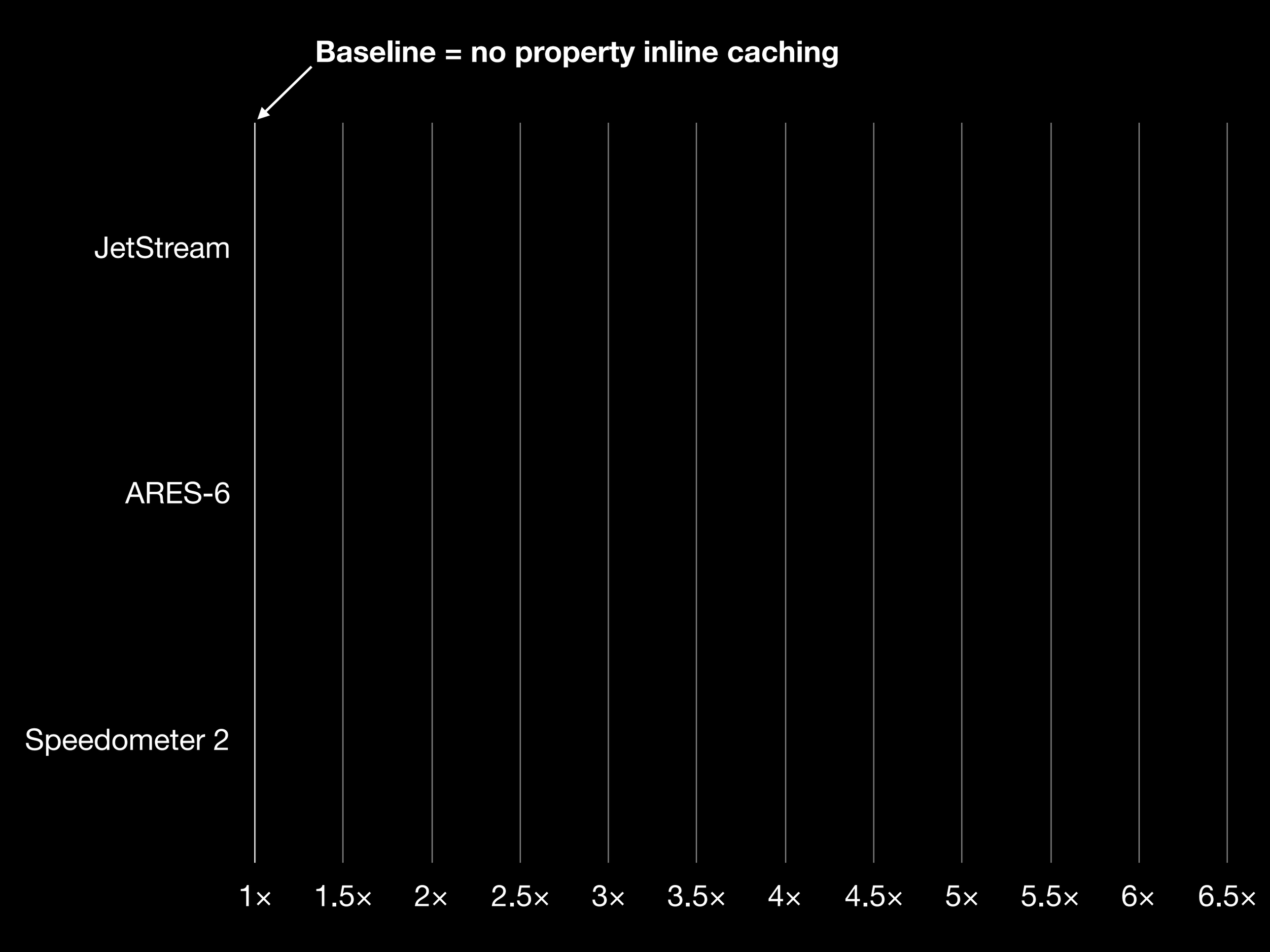
- Performance
- Why?
- Simple inline caching
  - in interpreter
  - in JIT
  - prototype
- Advanced topics
  - Polymorphic Access JIT
  - Inlining inline caches
  - Type inference

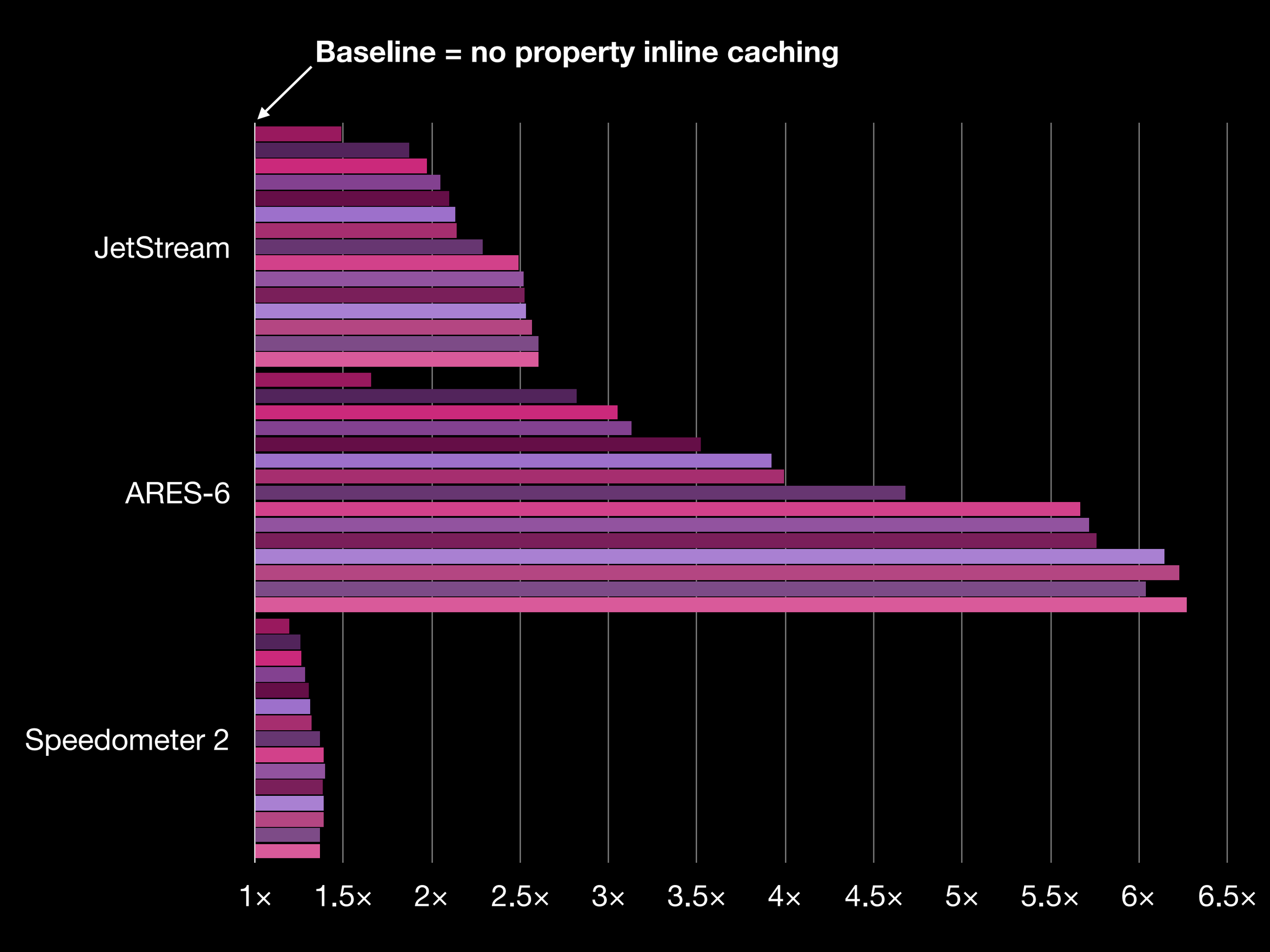
Performance

Benchmark	Summary	Number of Samples in my experiments
JetStream 1.1	ES5 benchmarks (real and synthetic)	9
ARES-6 1.0.1	ES6 benchmarks (real and synthetic)	24
Speedometer 2.0	DOM framework benchmarks (real)	30

Source: `browserbench.org`


JetStream
ARES-6
Speedometer 2
1×
1.5×
2×
2.5×
3×
3.5×
4×
4.5×
5×
5.5×
6×
6.5×


Baseline = no property inline caching
JetStream
ARES-6
Speedometer 2
1×
1.5×
2×
2.5×
3×
3.5×
4×
4.5×
5×
5.5×
6×
6.5×

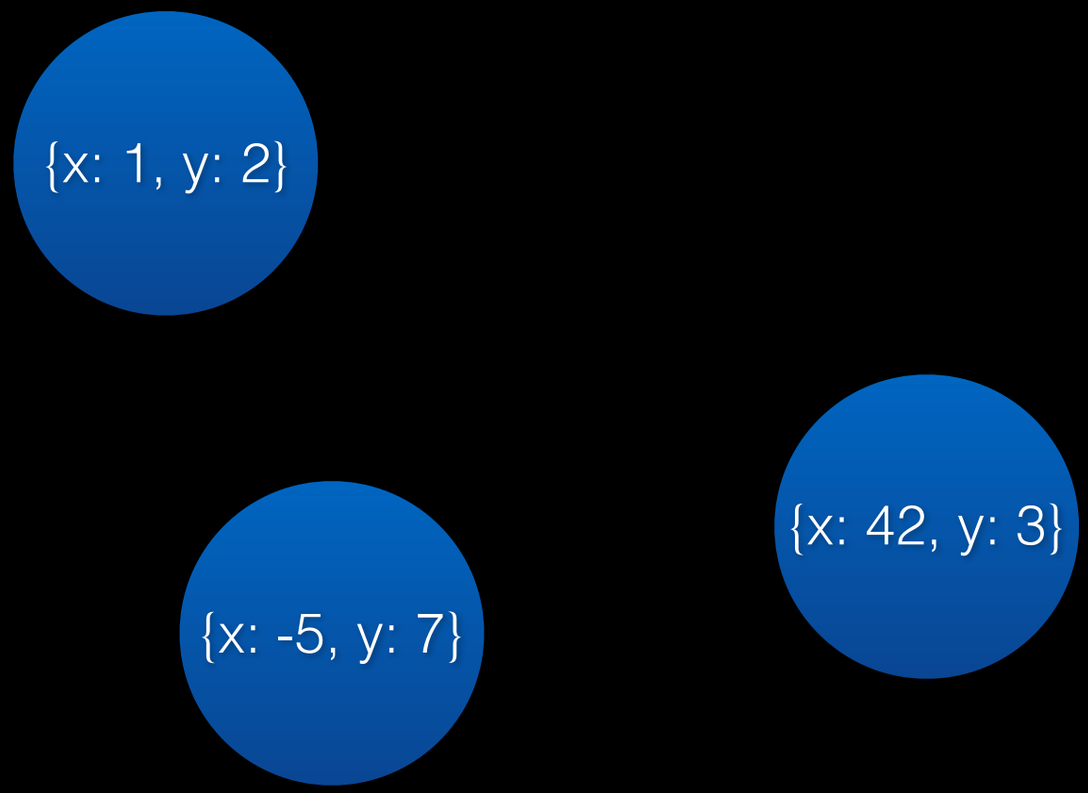

Baseline = no property inline caching
JetStream
ARES-6
Speedometer 2
1×
1.5×
2×
2.5×
3×
3.5×
4×
4.5×
5×
5.5×
6×
6.5×

{x: 1, y: 2}

{x: -5, y: 7}

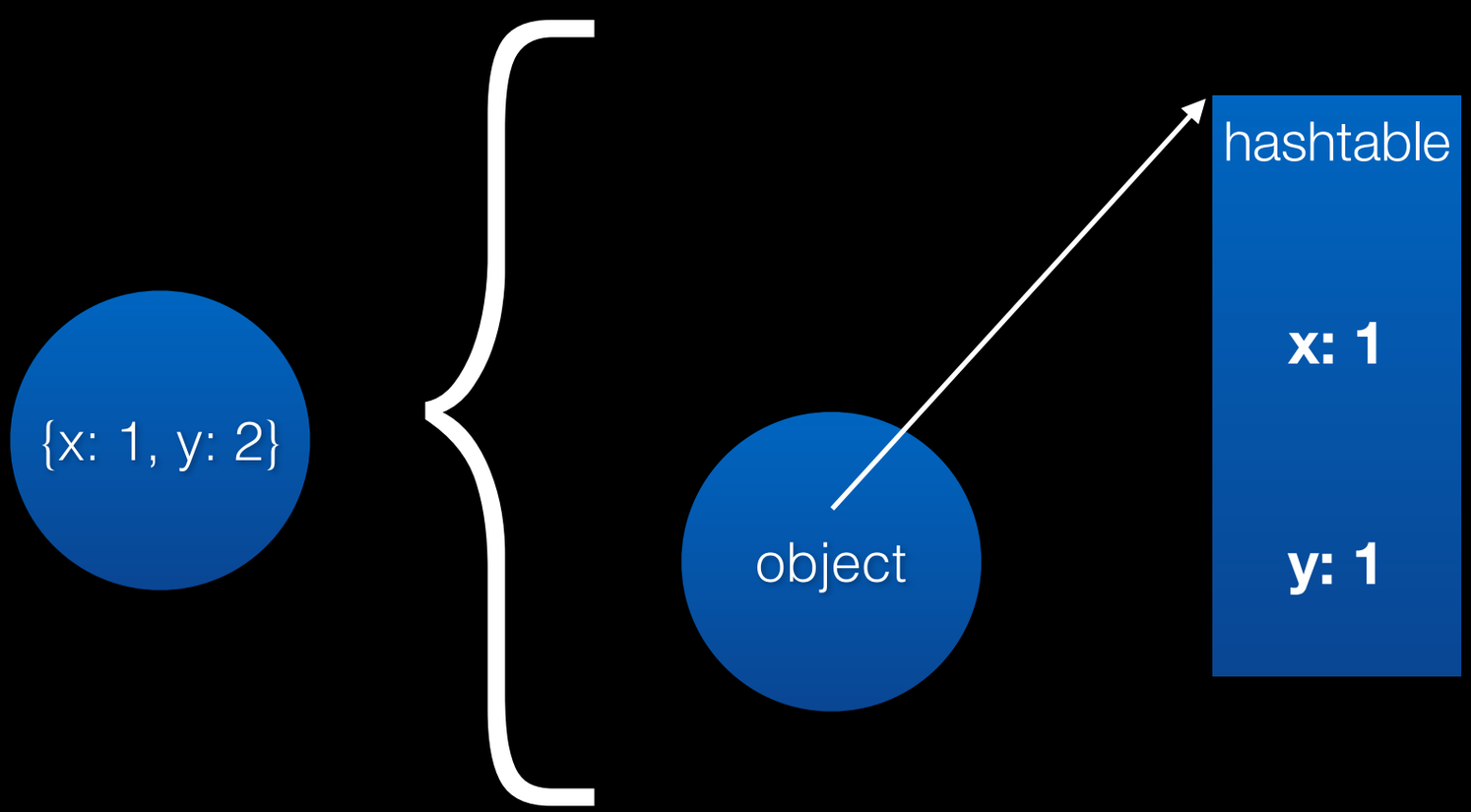
{x: 42, y: 3}

```
var x = o.x;
```

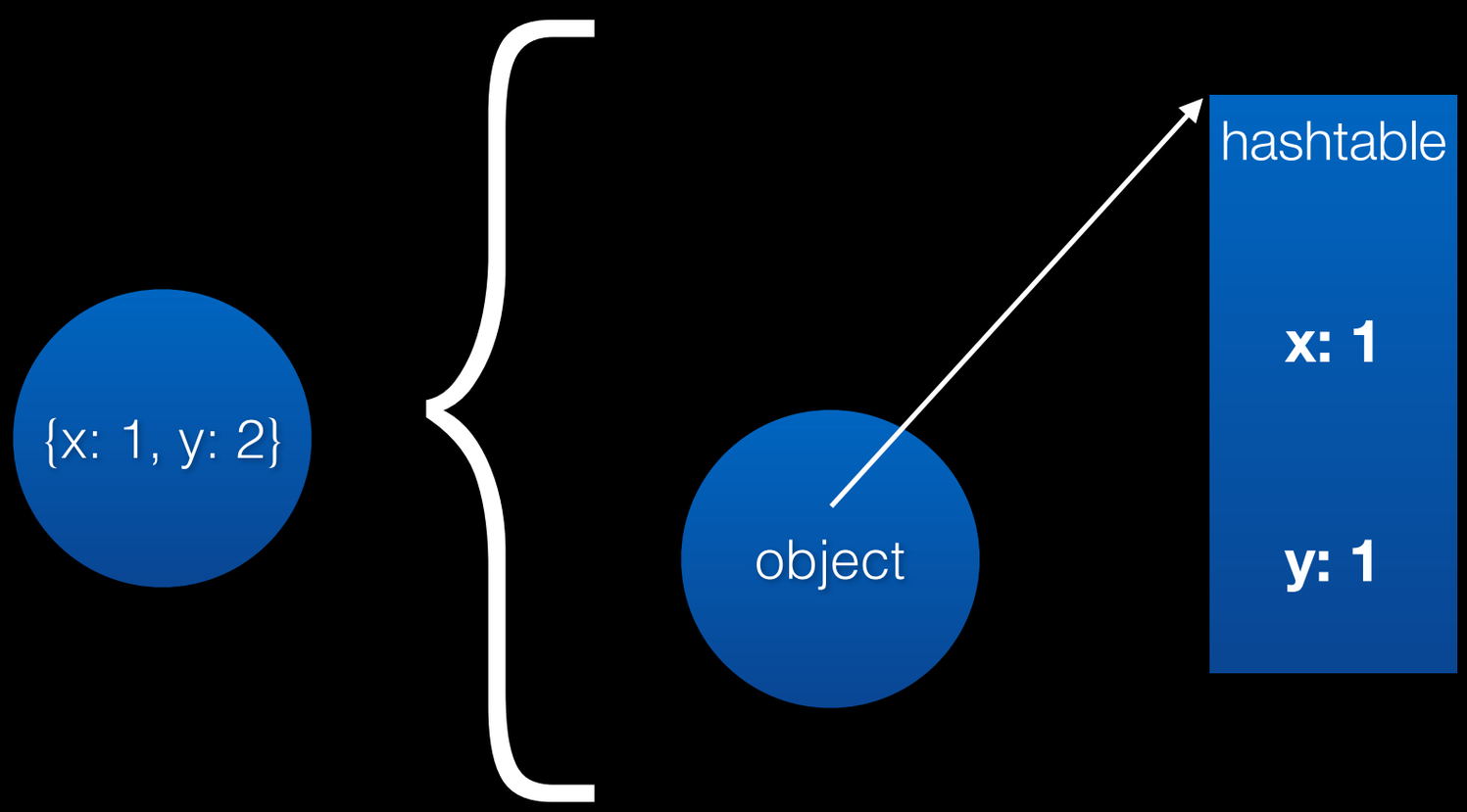


```
o.x = x;
```

Hashtable

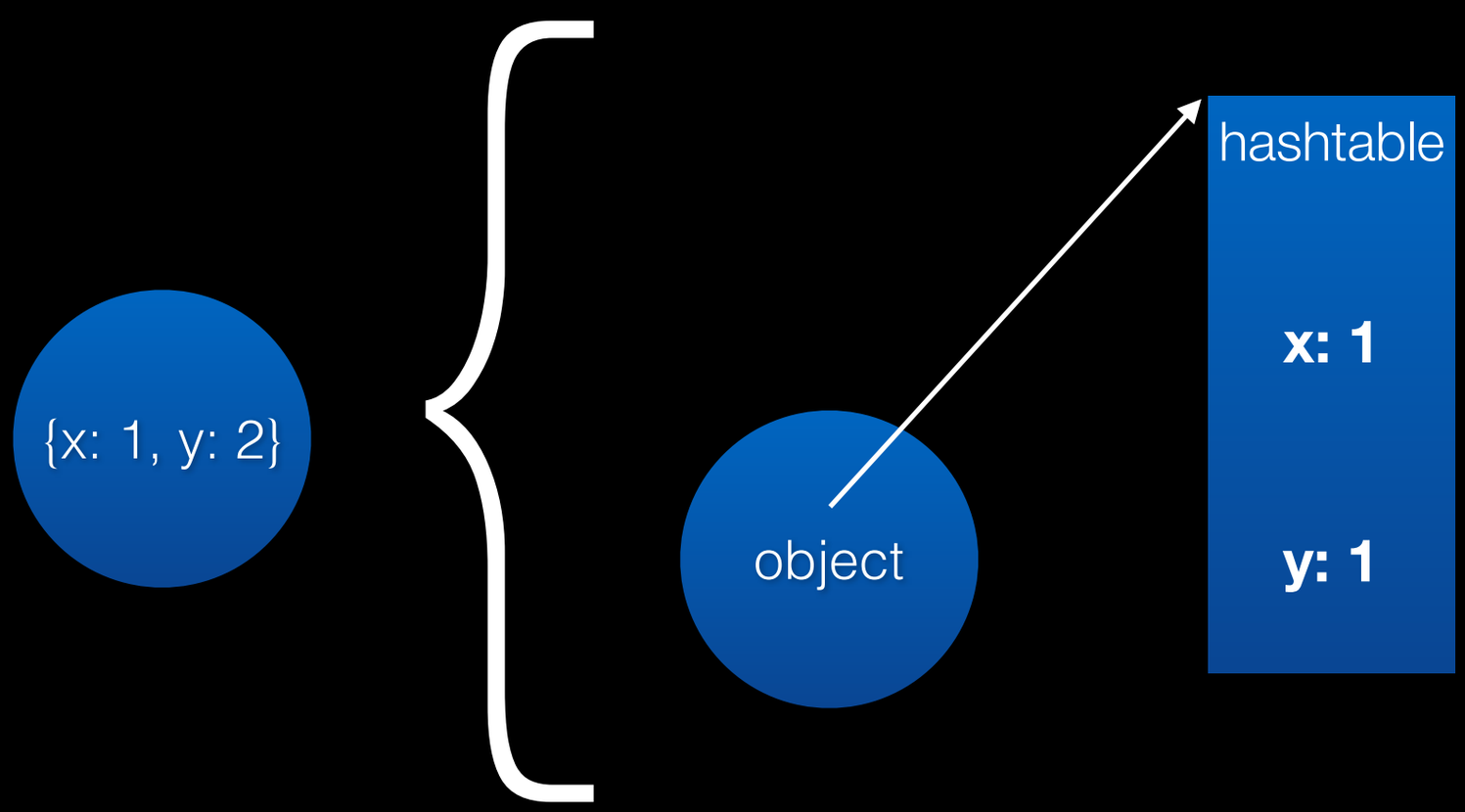


Hashtable



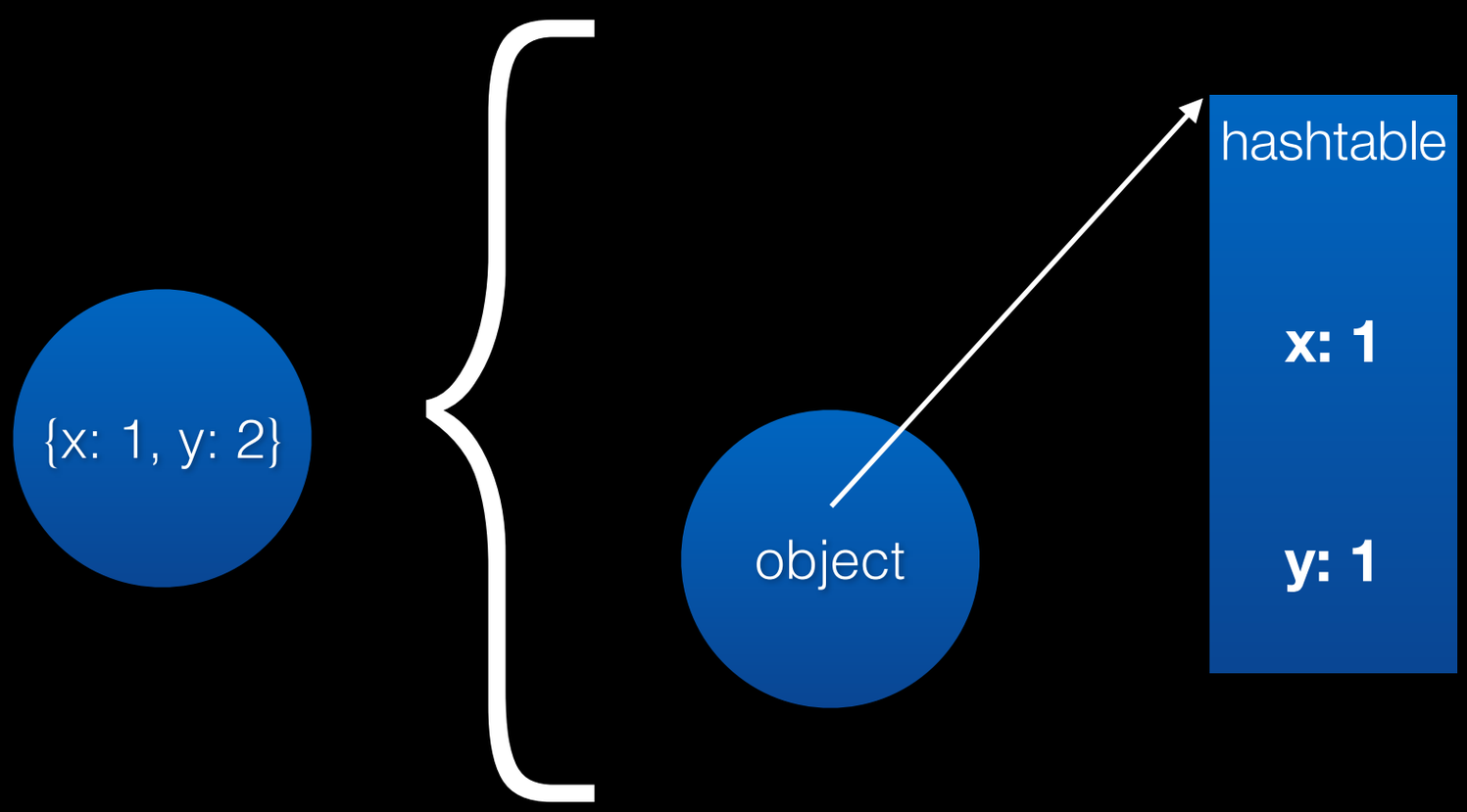
- Pointer chasing is slow

Hashtable



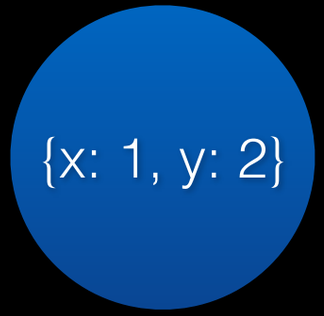
- Pointer chasing is slow
- Hash codes take time to compute

Hashtable



- Pointer chasing is slow
- Hash codes take time to compute
- Lots of instructions, hard to inline

Hashtable

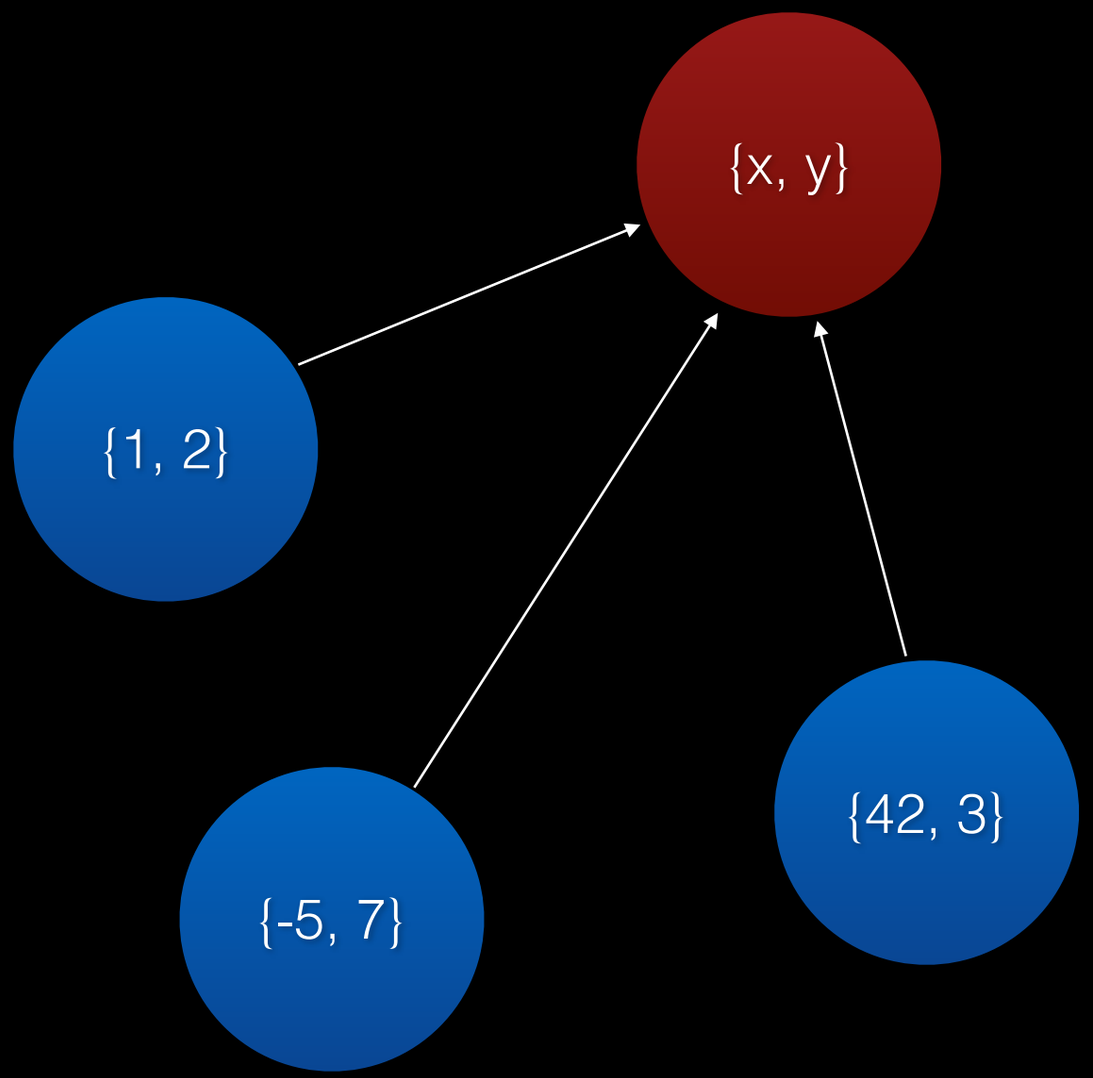


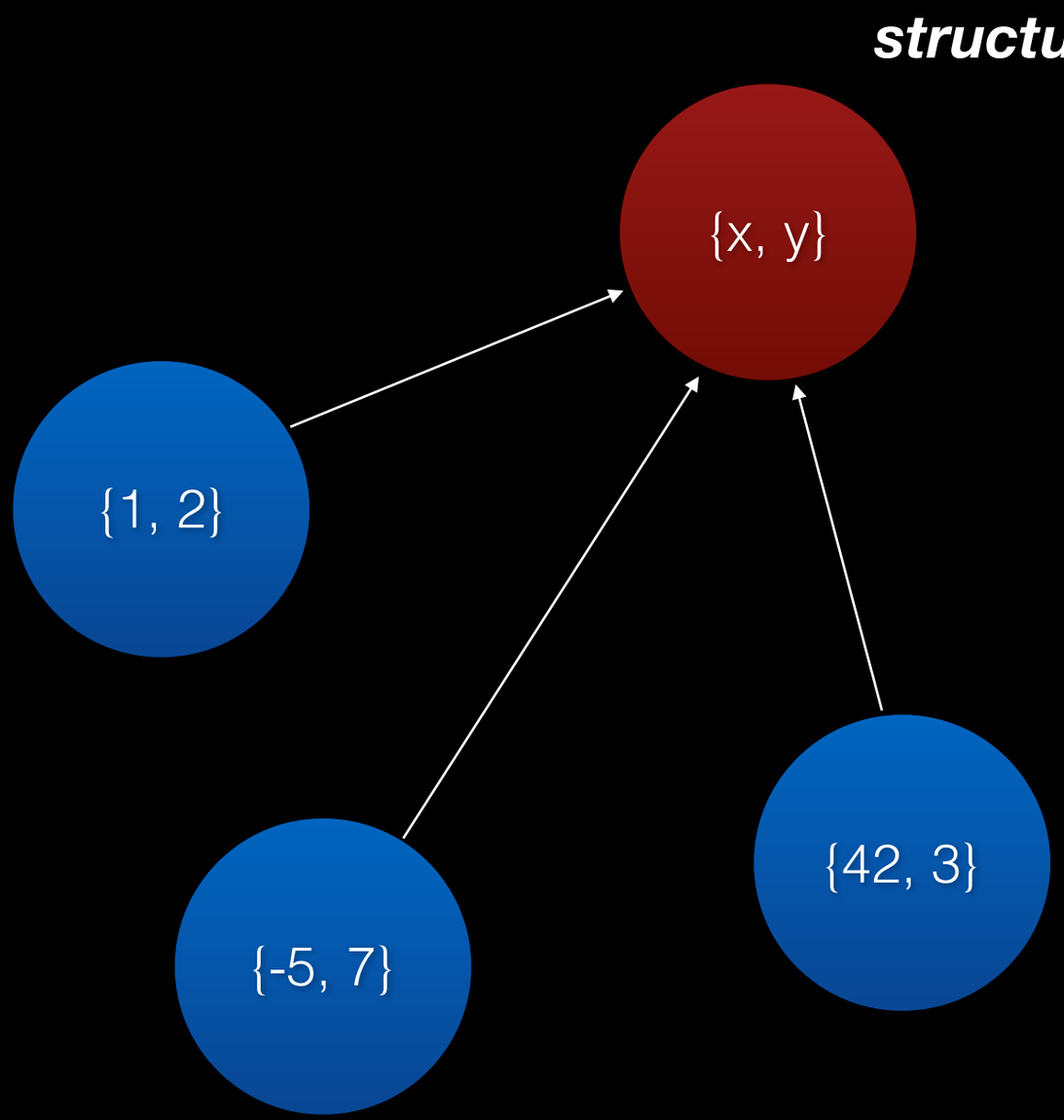
- Pointer chasing is slow
- Hash codes take time to compute
- Lots of instructions, hard to inline

{x: 1, y: 2}

{x: -5, y: 7}

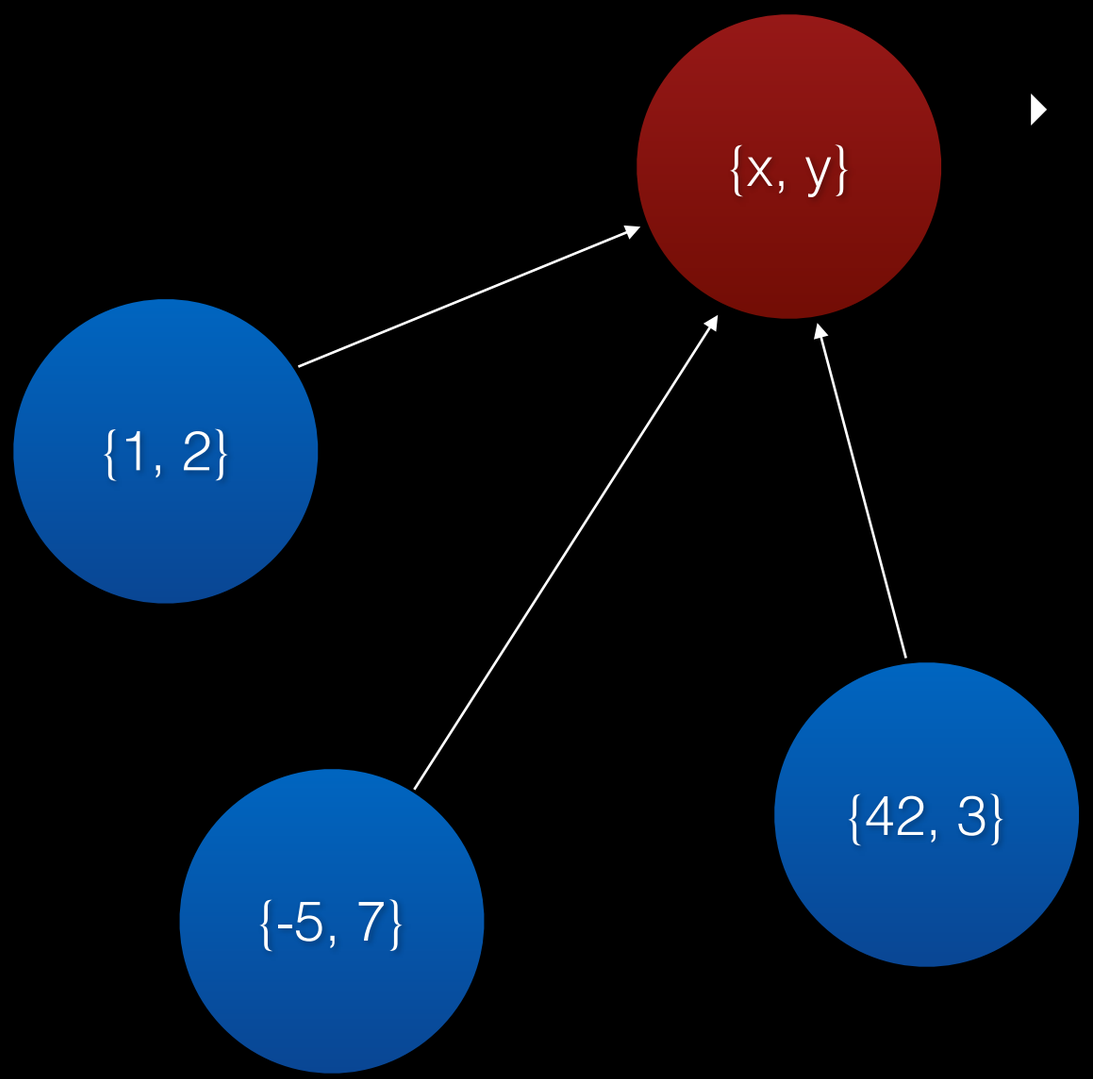
{x: 42, y: 3}


{x, y}
{1, 2}
{-5, 7}
{42, 3}

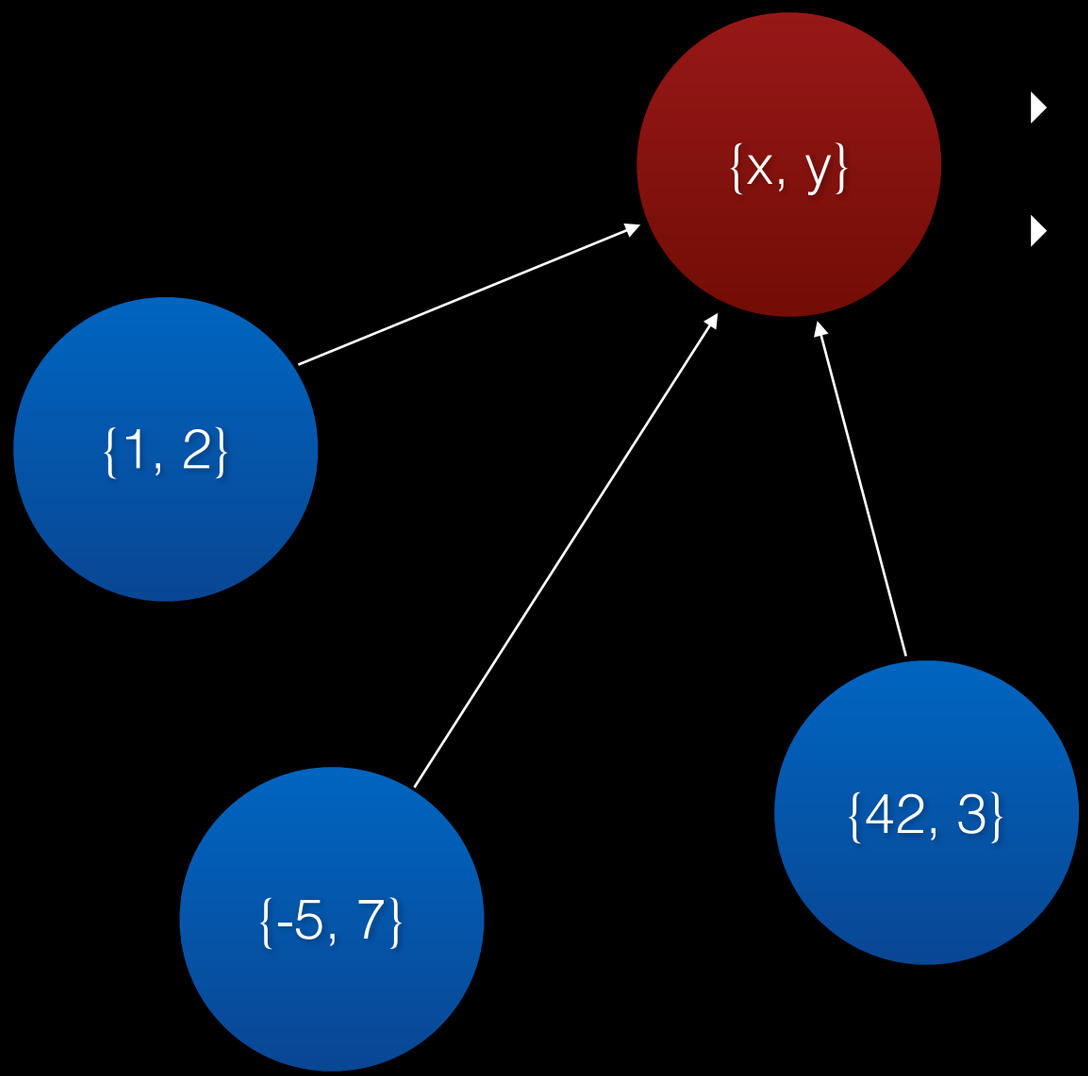

structure
{x, y}
{1, 2}
{-5, 7}
{42, 3}

structure

- hashtable

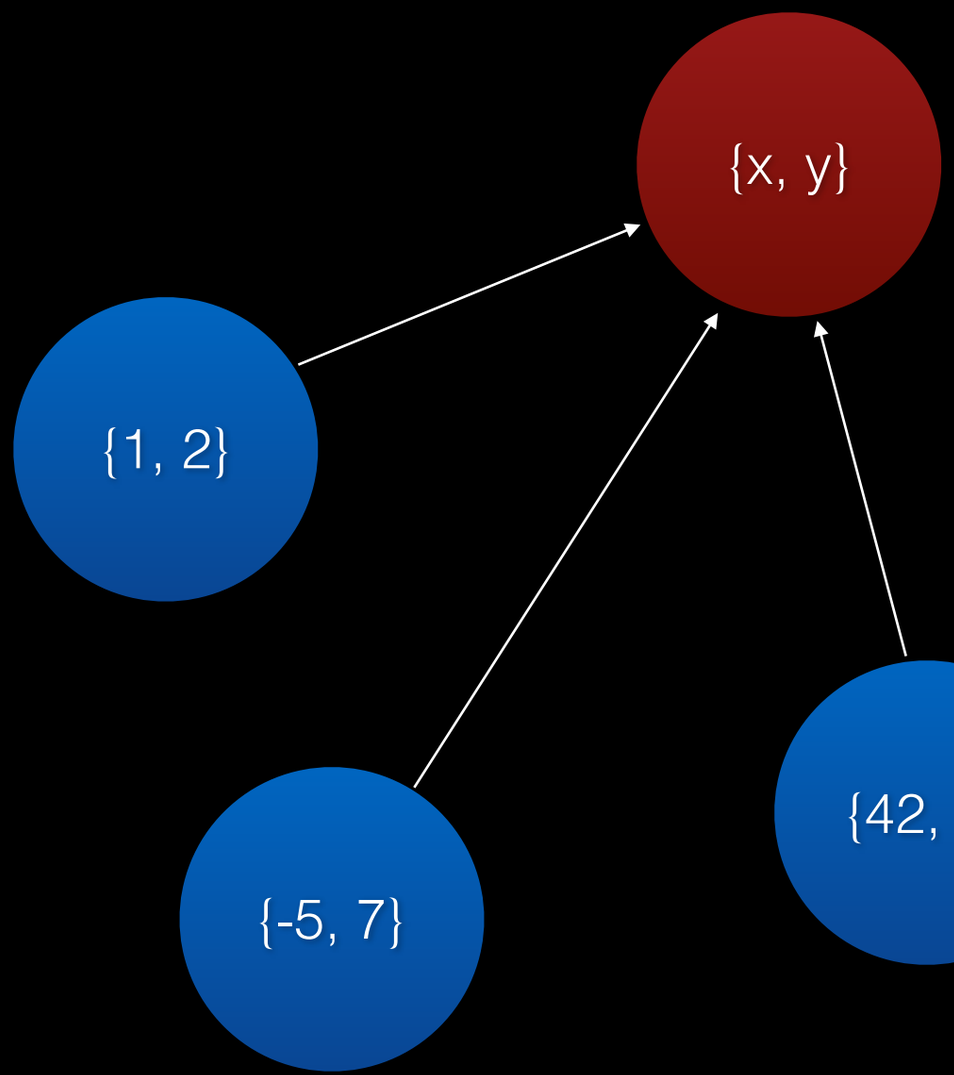


structure



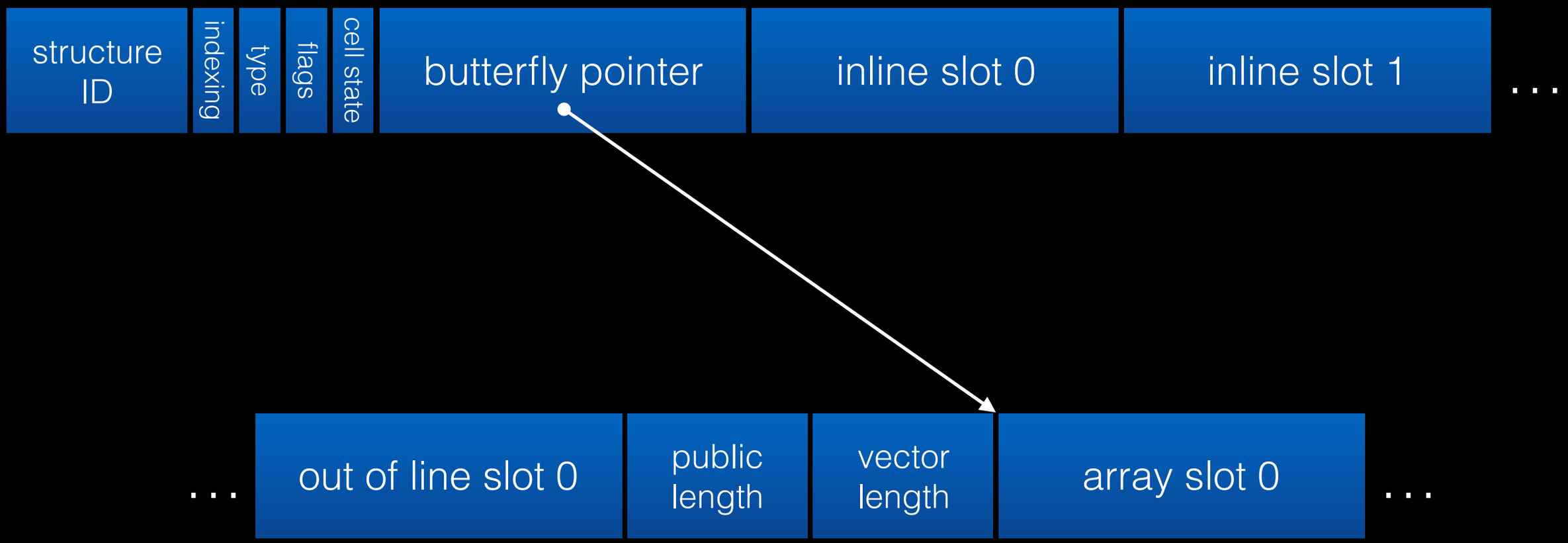
- hashtable
- maps name to offset

structure

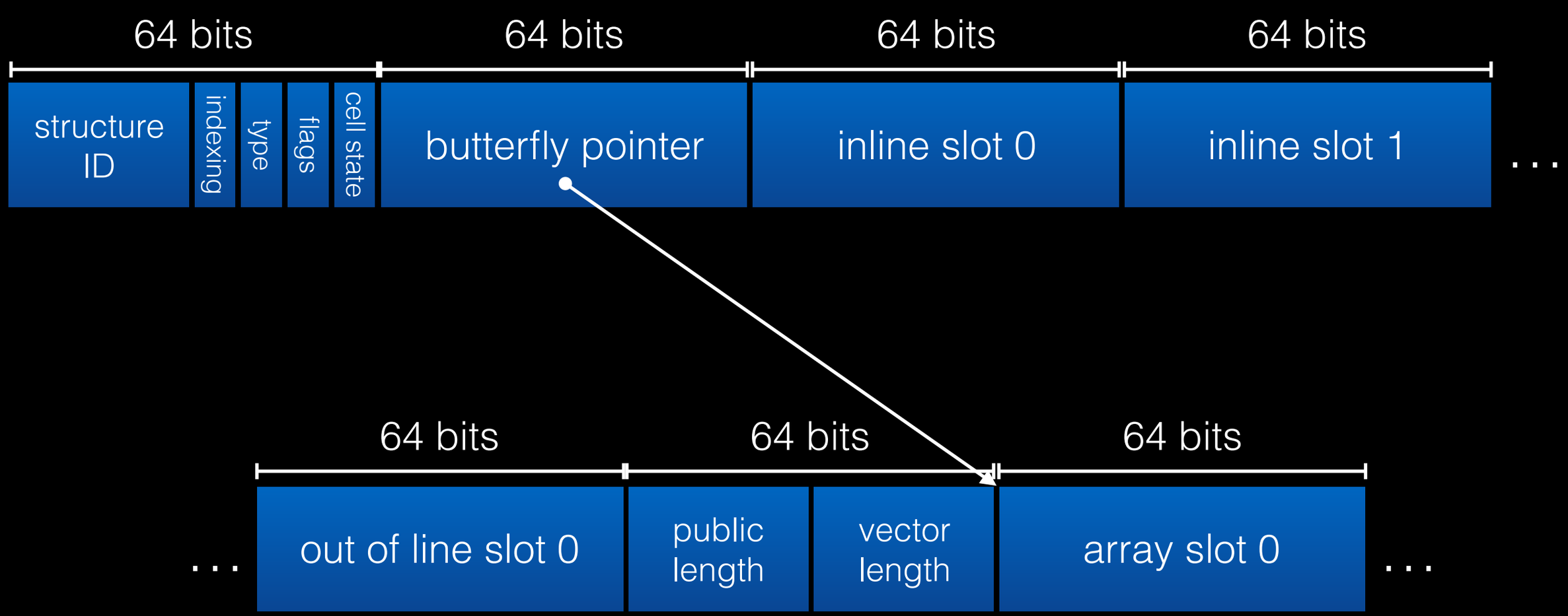


- hashtable
- maps name to offset
- hash-consed

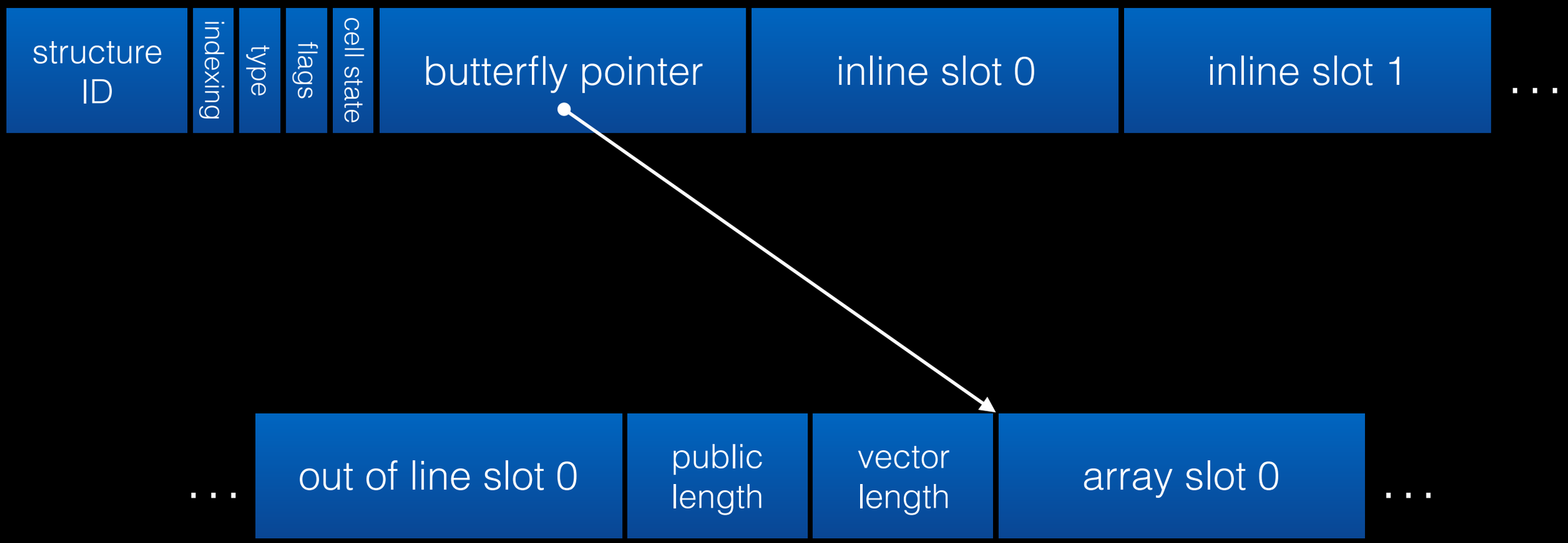
JSC Object Model



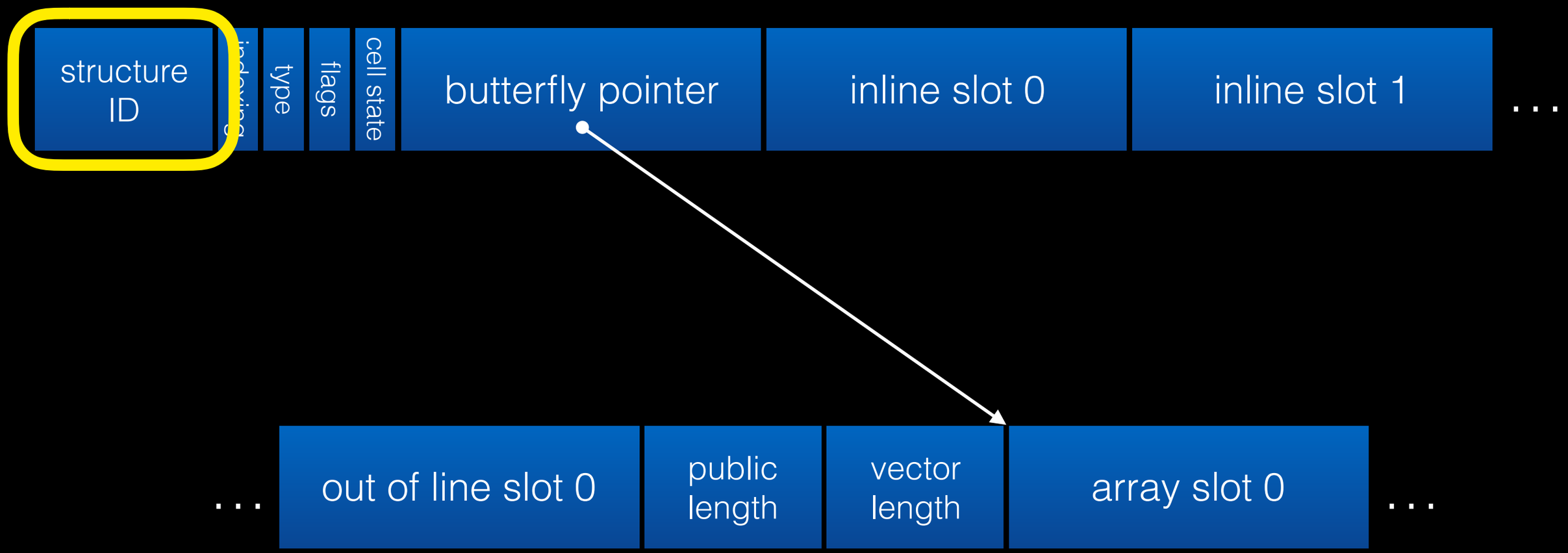
JSC Object Model



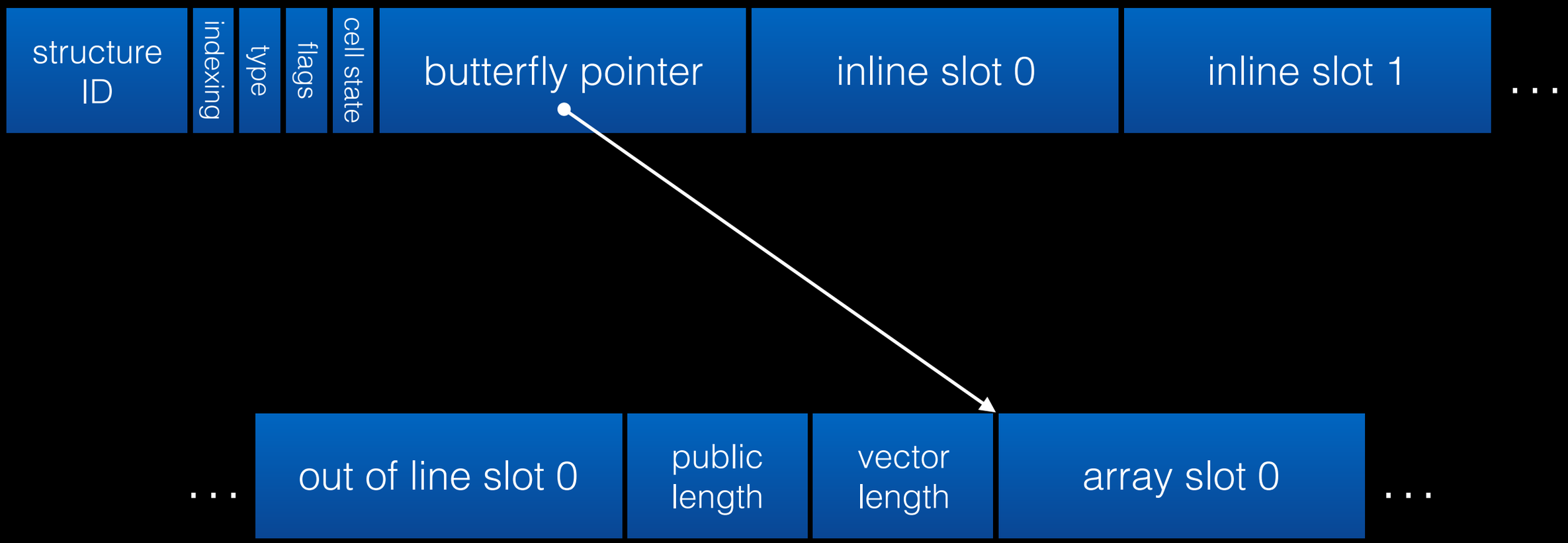
JSC Object Model



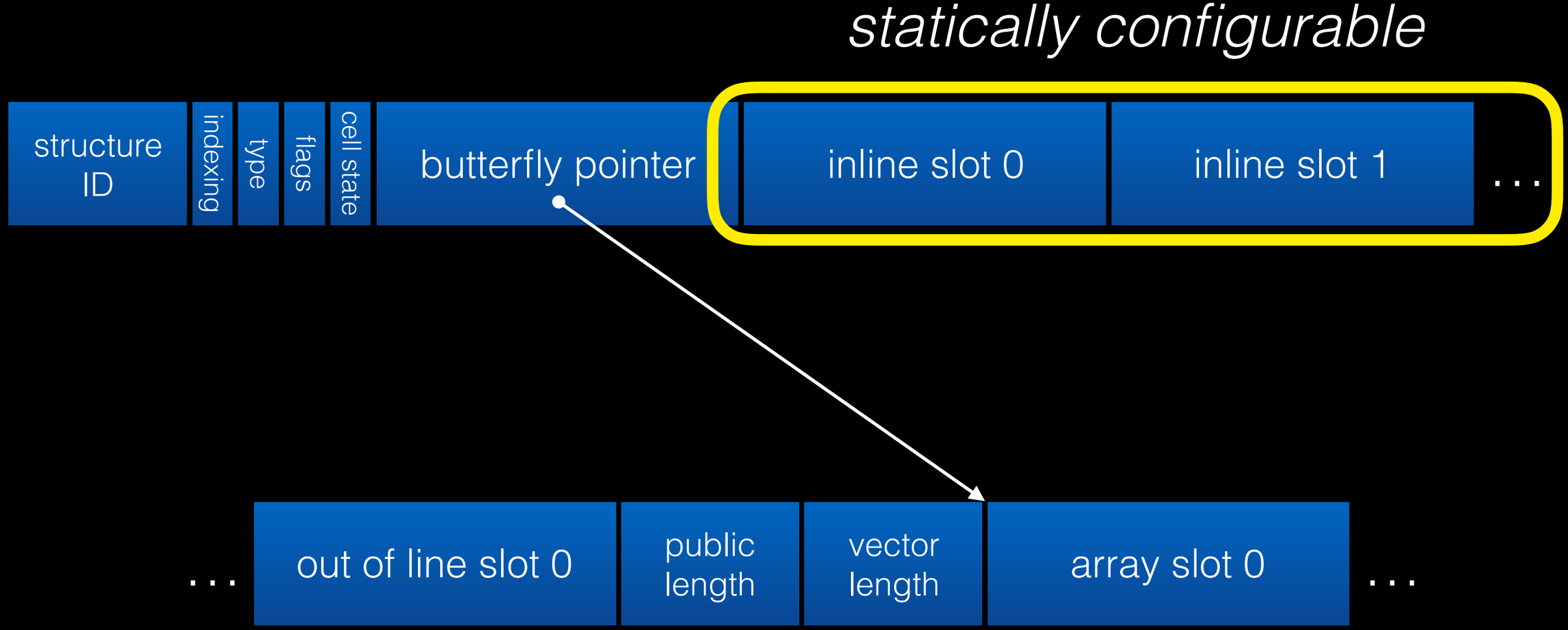
JSC Object Model



JSC Object Model

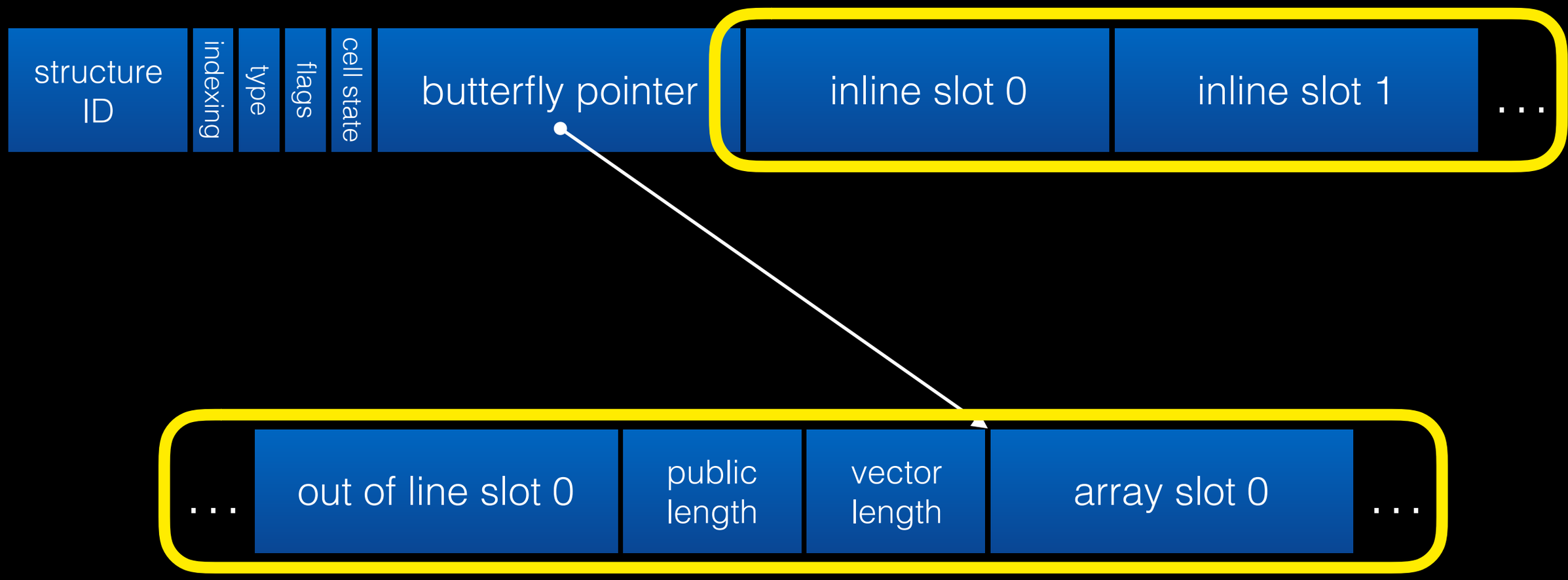


JSC Object Model



JSC Object Model

statically configurable



dynamically configurable

Fast JSObject

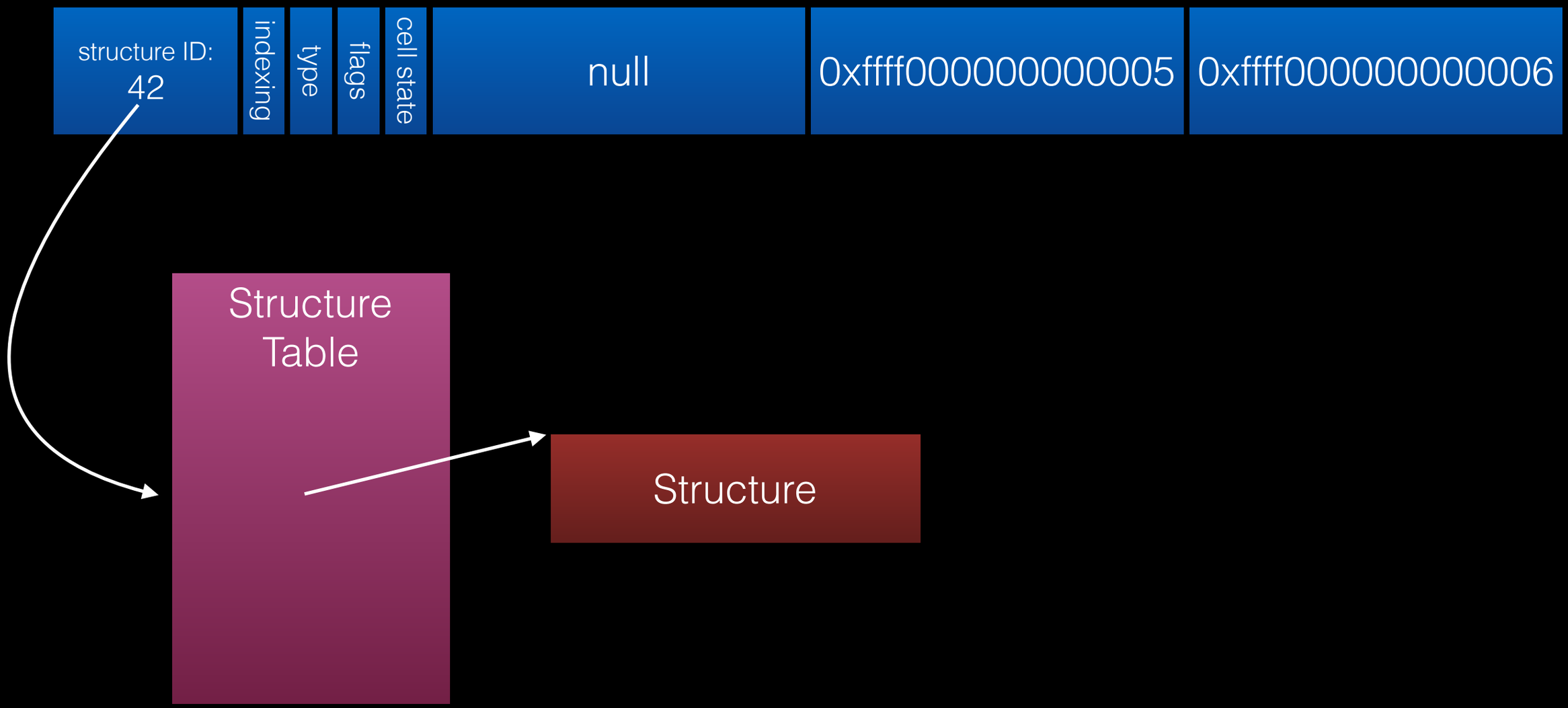
var o = {f: 5, g: 6};

Fast JSObject

var o = {f: 5, g: 6};

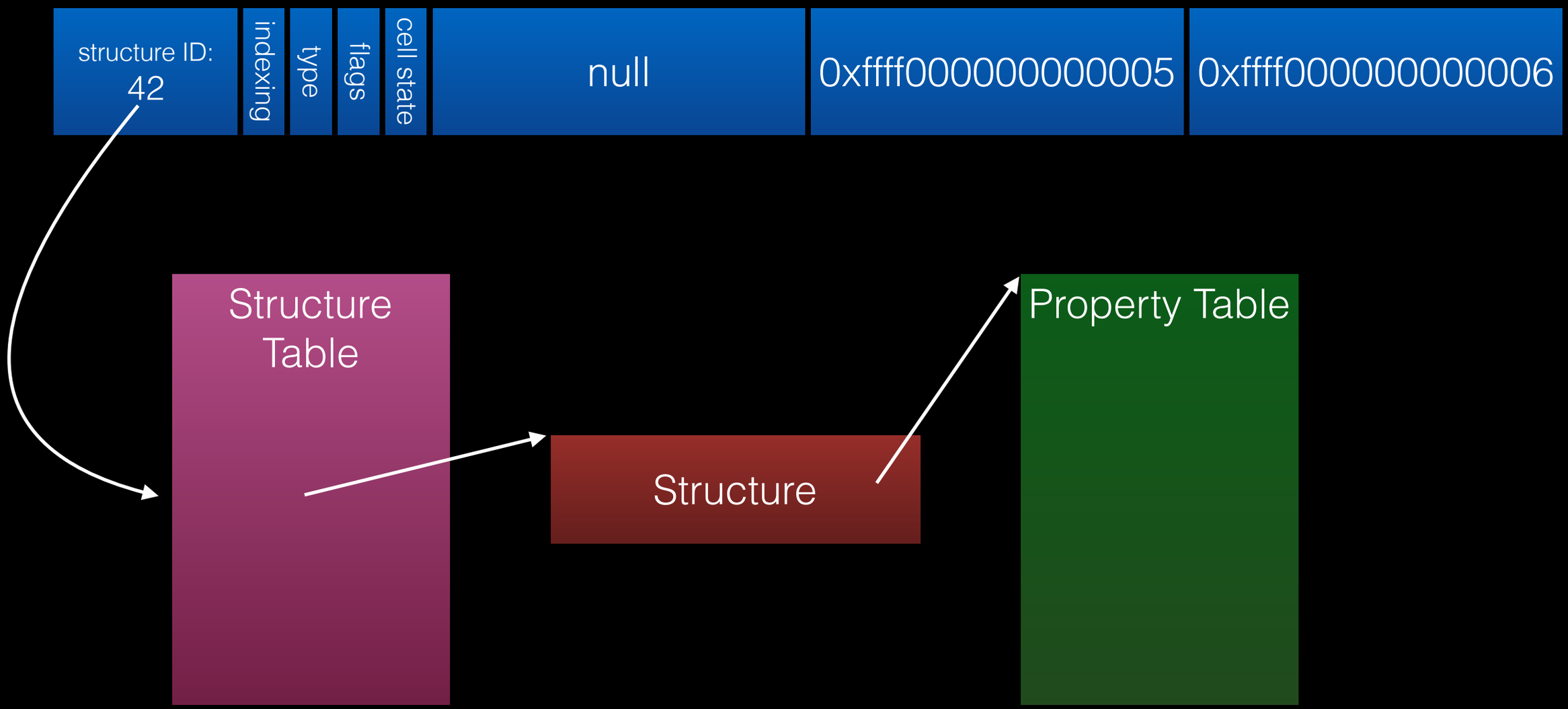
Fast JSObject

var o = {f: 5, g: 6};



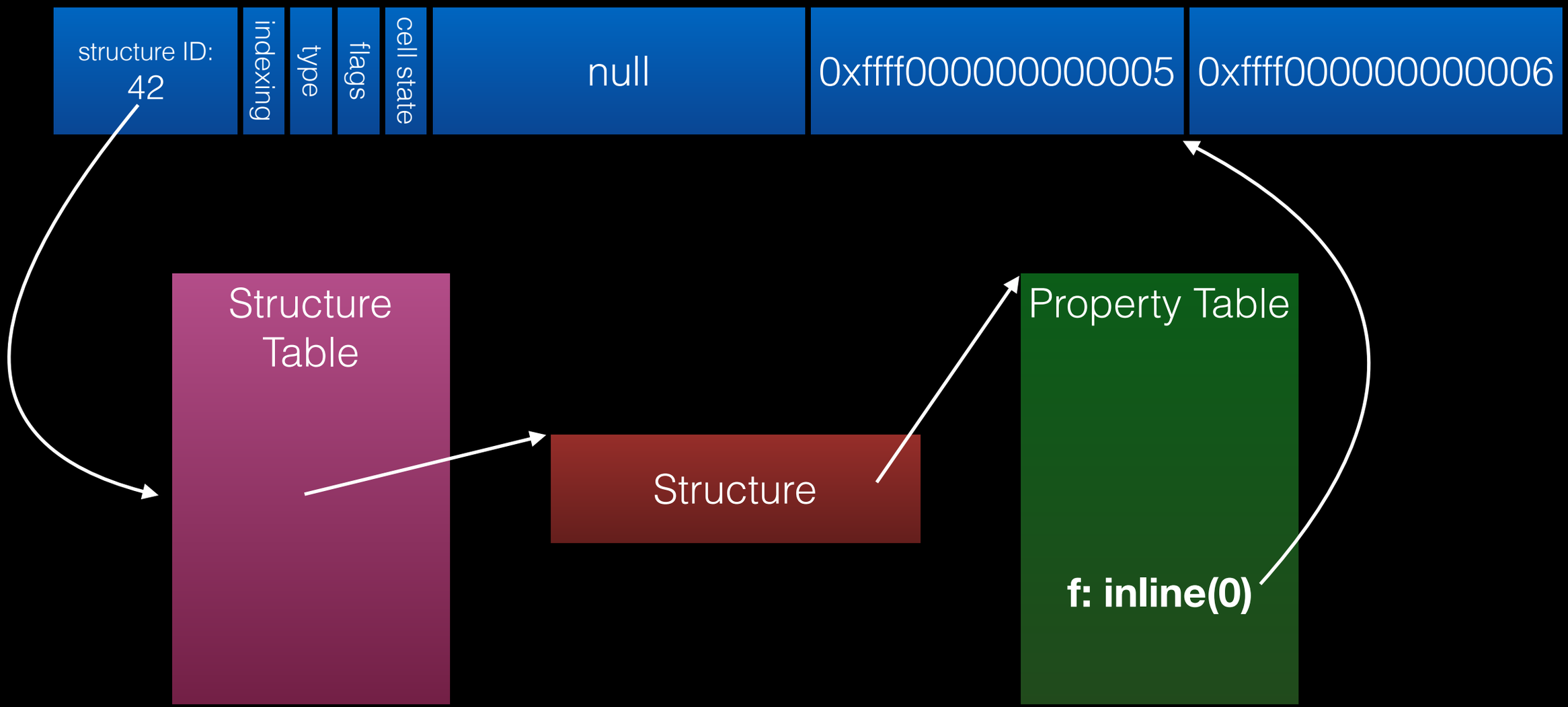
Fast JSObject

var o = {f: 5, g: 6};



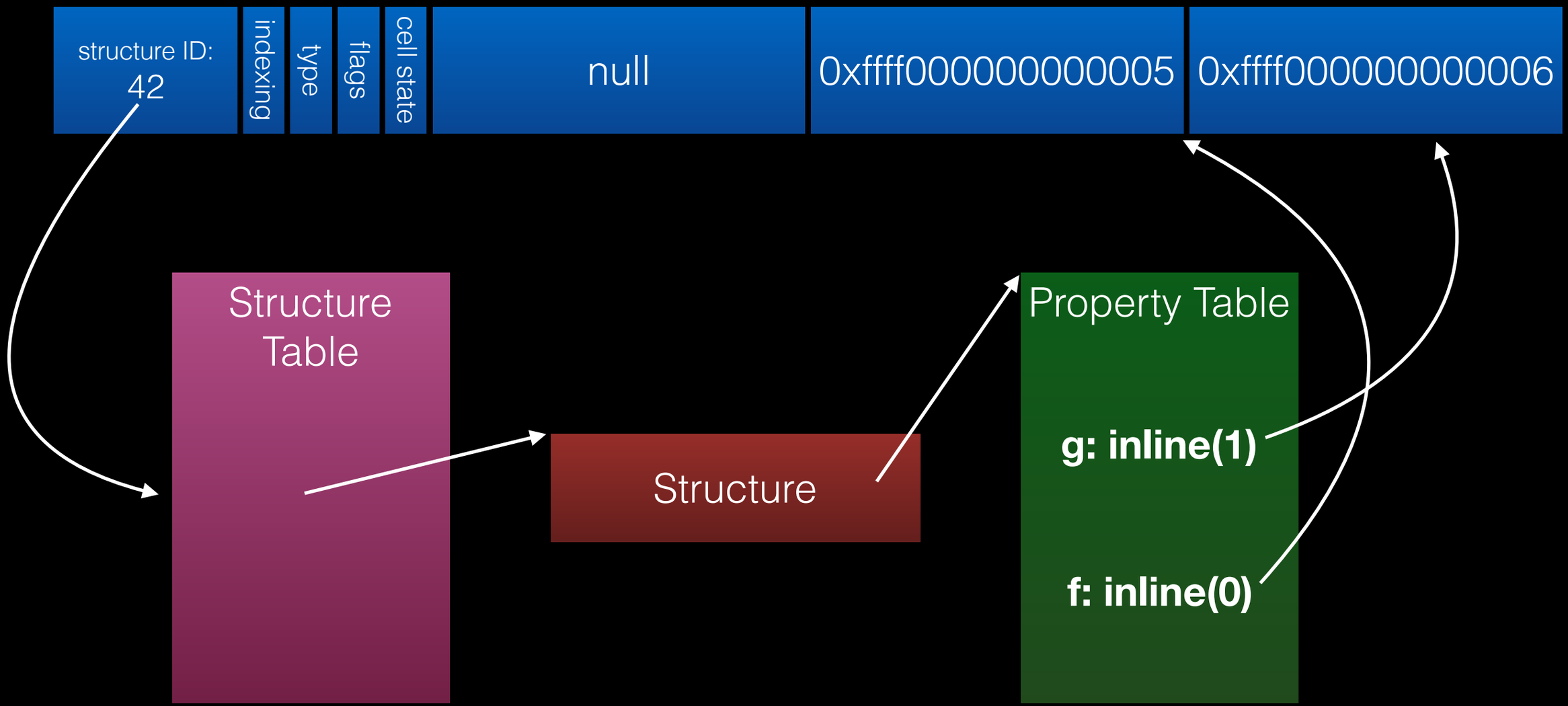
Fast JSObject

var o = {f: 5, g: 6};

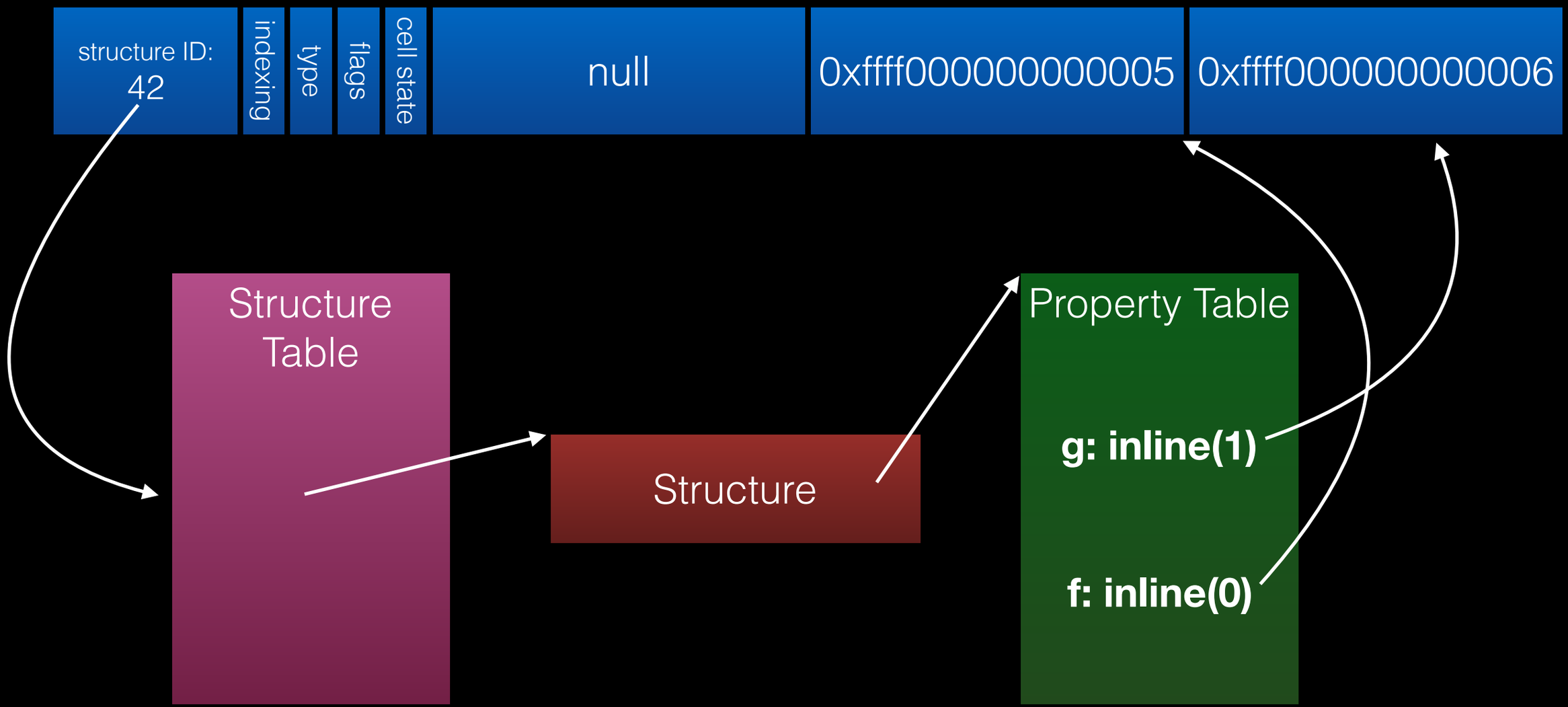


Fast JSObject

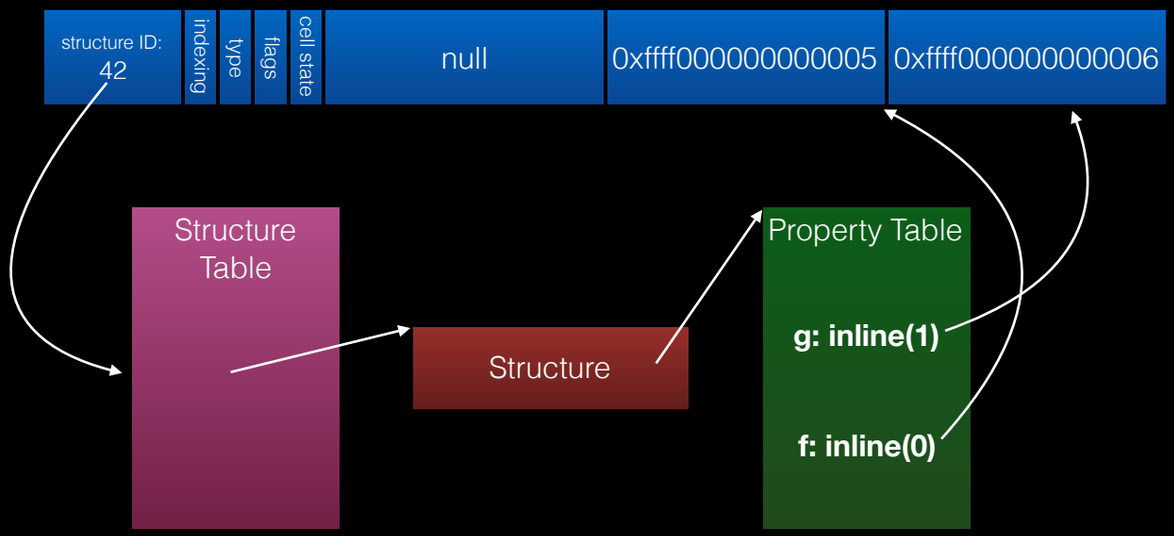
var o = {f: 5, g: 6};



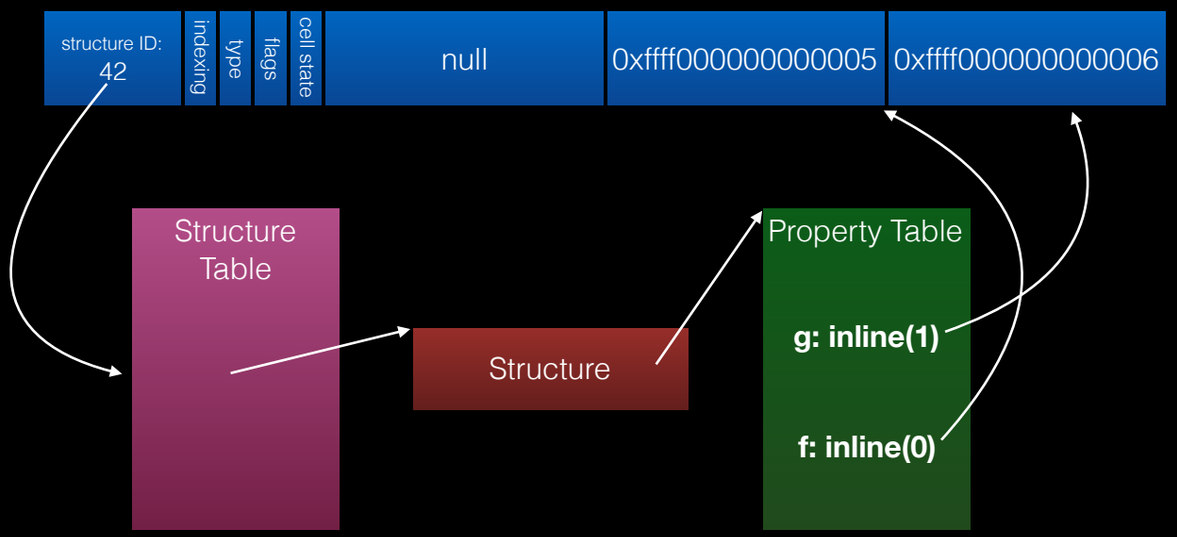
var o = {f: 5, g: 6};


structure ID:
42
indexing
type
flags
cell state
null
0xffff000000000005
0xffff000000000006
Structure Table
Structure
Property Table
g: inline(1)
f: inline(0)

var o = {f: 5, g: 6};

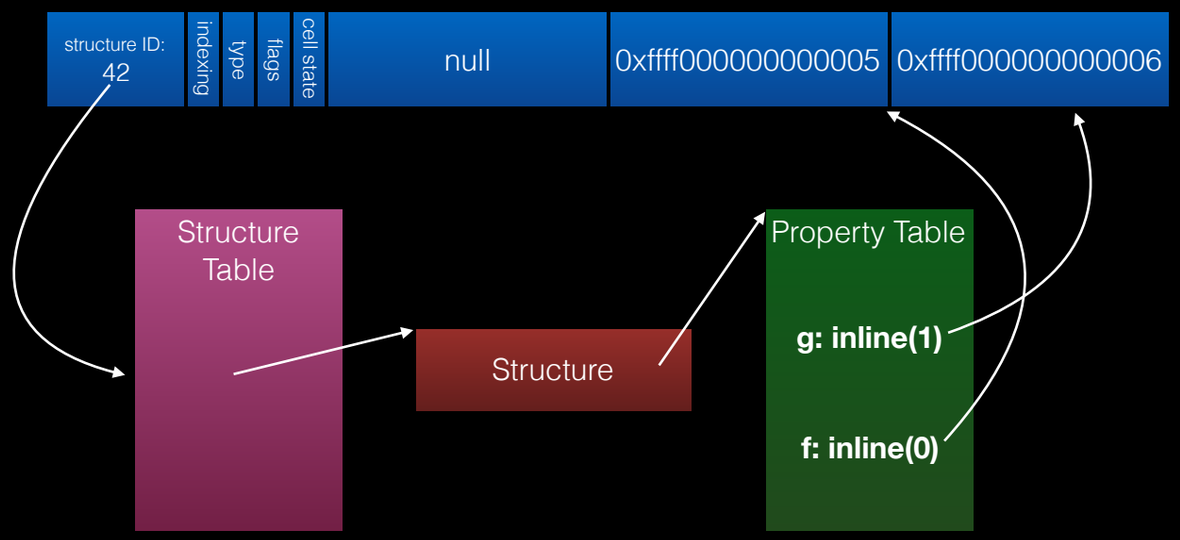

structure ID:
42
indexing
type
flags
cell state
null
0xffff000000000005
0xffff000000000006
Structure Table
Structure
Property Table
g: inline(1)
f: inline(0)

var o = {f: 5, g: 6};




```
var v = o.f;
```

var o = {f: 5, g: 6};

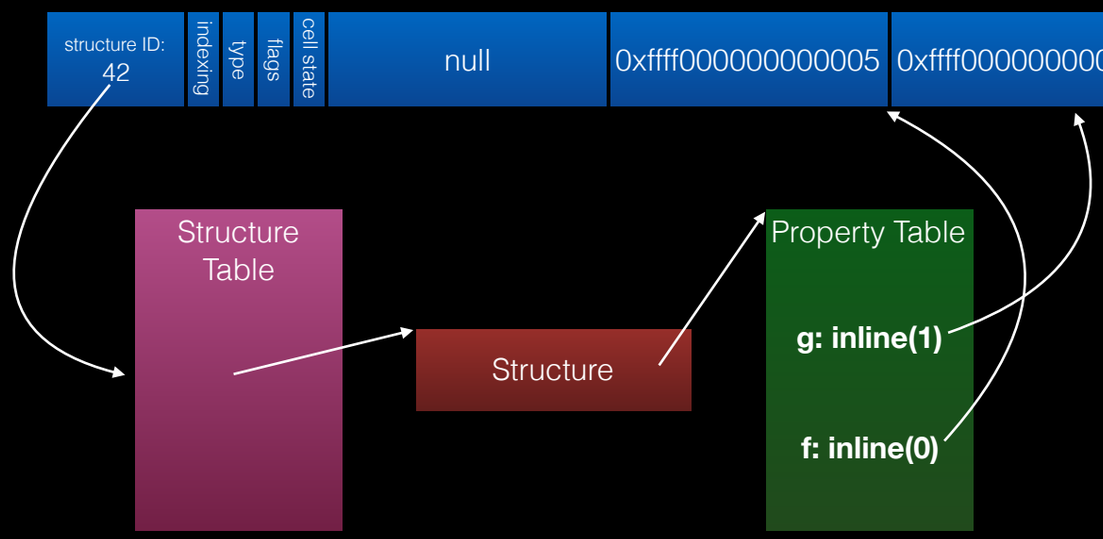



```
var v = o.f;
```

```
if (o->structureID == 42)
    v = o->inlineStorage[0]
else
    v = slowGet(o, “f”)
```

“Inline Cache”

var o = {f: 5, g: 6};




```
var v = o.f;
```

```
if (o->structureID == 42)
    v = o->inlineStorage[0]
else
    v = slowGet(o, “f”)
```

Interpreter Inline Cache

```
get_by_id <result>, <base>, <properyName>
```

Interpreter Inline Cache

```
get_by_id <result>, <base>, <properyName>,
    <cachedStructureID>, <cachedOffset>
```

Interpreter Inline Cache

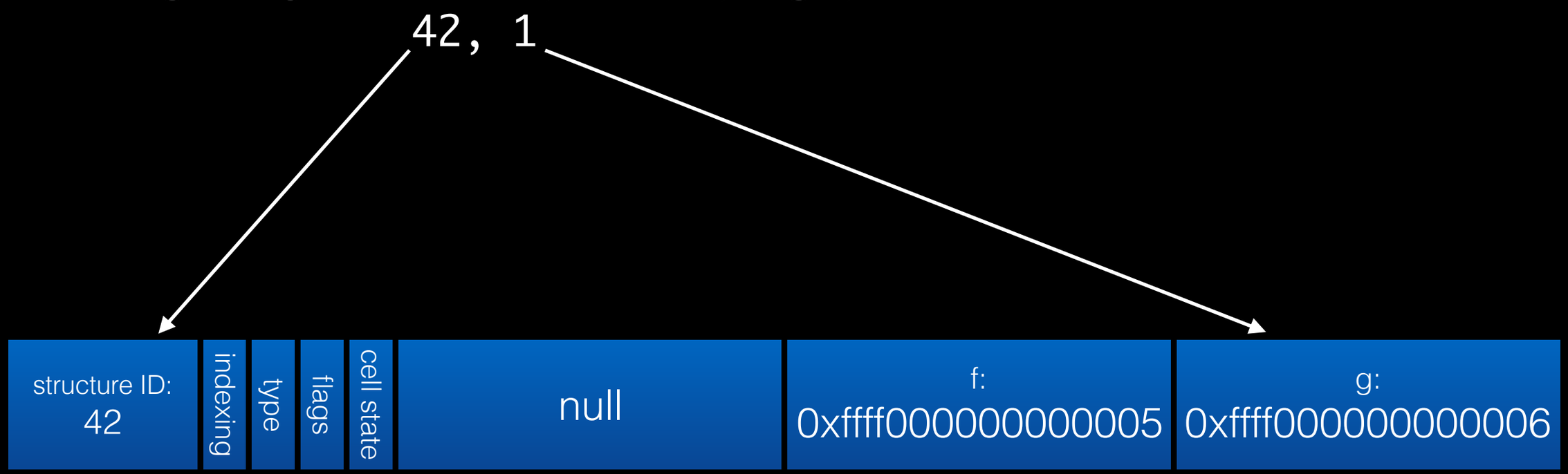
```
get_by_id loc42, loc43, “g”,
    0, 0
```

Interpreter Inline Cache

```
get_by_id loc42, loc43, “g”,
0, 0
```

Interpreter Inline Cache

```
get_by_id loc42, loc43, “g”,
42, 1
```



JIT Inline Cache

```
0x46f8c30b9b0: mov 0x30(%rbp), %rax
0x46f8c30b9b4: test %rax, %r15
0x46f8c30b9b7: jnz 0x46f8c30ba2c
0x46f8c30b9bd: jmp 0x46f8c30ba2c
0x46f8c30b9c2: o16 nop %cs:0x200(%rax,%rax)
0x46f8c30b9d1: nop (%rax)
0x46f8c30b9d4: mov %rax, -0x38(%rbp)
```

JIT Inline Cache

```
0x46f8c30b9b0: mov 0x30(%rbp), %rax
0x46f8c30b9b4: test %rax, %r15
0x46f8c30b9b7: jnz 0x46f8c30ba2c
0x46f8c30b9bd: jmp 0x46f8c30ba2c
0x46f8c30b9c2: o16 nop %cs:0x200(%rax,%rax)
0x46f8c30b9d1: nop (%rax)
0x46f8c30b9d4: mov %rax, -0x38(%rbp)
```

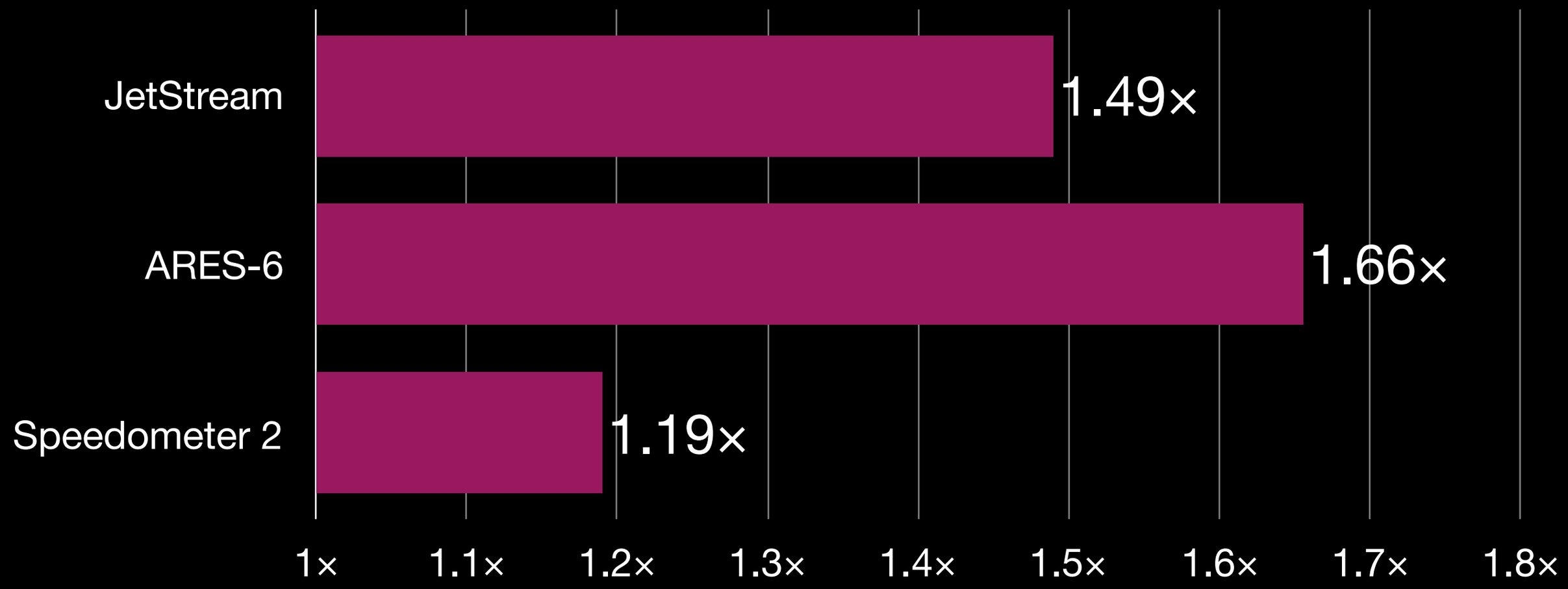
JIT Inline Cache

```
0x46f8c30b9b0: mov 0x30(%rbp), %rax
0x46f8c30b9b4: test %rax, %r15
0x46f8c30b9b7: jnz 0x46f8c30ba2c
0x46f8c30b9bd: jmp 0x46f8c30ba2c
0x46f8c30b9c2: o16 nop %cs:0x200(%rax,%rax)
0x46f8c30b9d1: nop (%rax)
0x46f8c30b9d4: mov %rax, -0x38(%rbp)
```

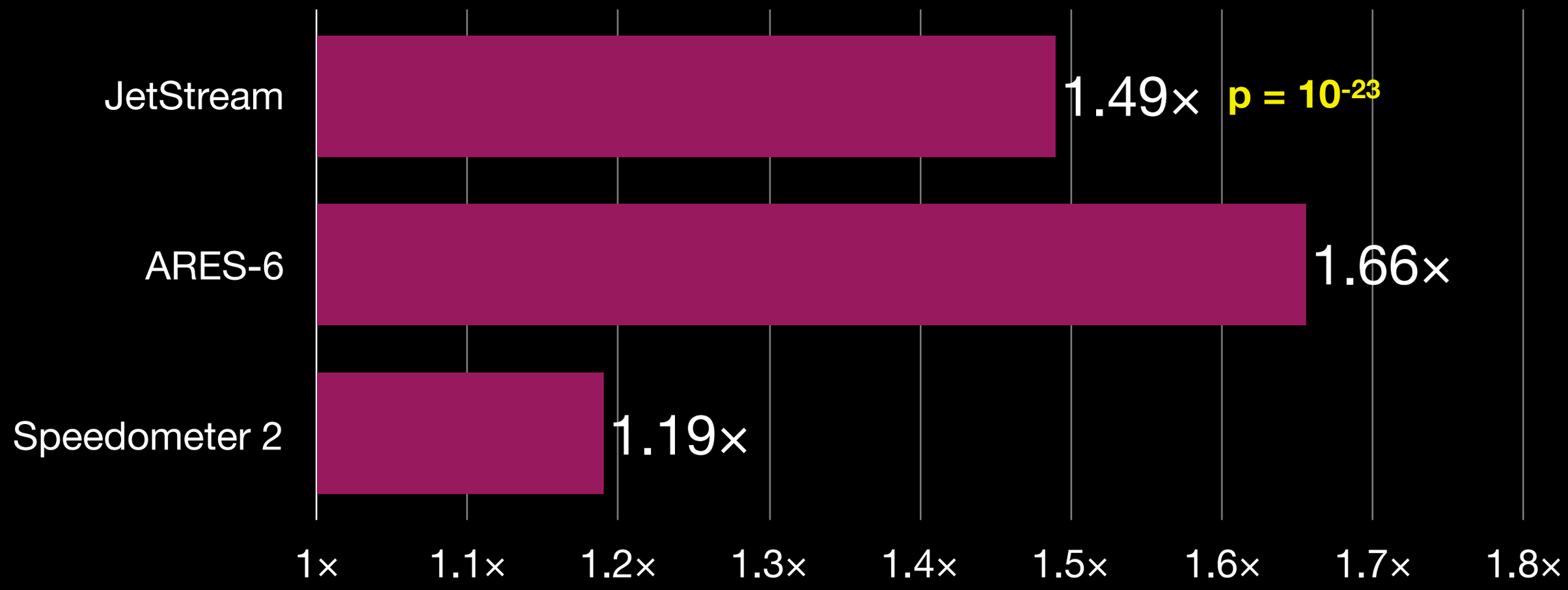
JIT Inline Cache

```
0x46f8c30b9b0: mov 0x30(%rbp), %rax
0x46f8c30b9b4: test %rax, %r15
0x46f8c30b9b7: jnz 0x46f8c30ba2c
0x46f8c30b9bd: cmp $0x125, (%rax)
0x46f8c30b9c3: jnz 0x46f8c30ba2c
0x46f8c30b9c9: mov 0x18(%rax), %rax
0x46f8c30b9cd: nop 0x200(%rax)
0x46f8c30b9d4: mov %rax, -0x38(%rbp)
```

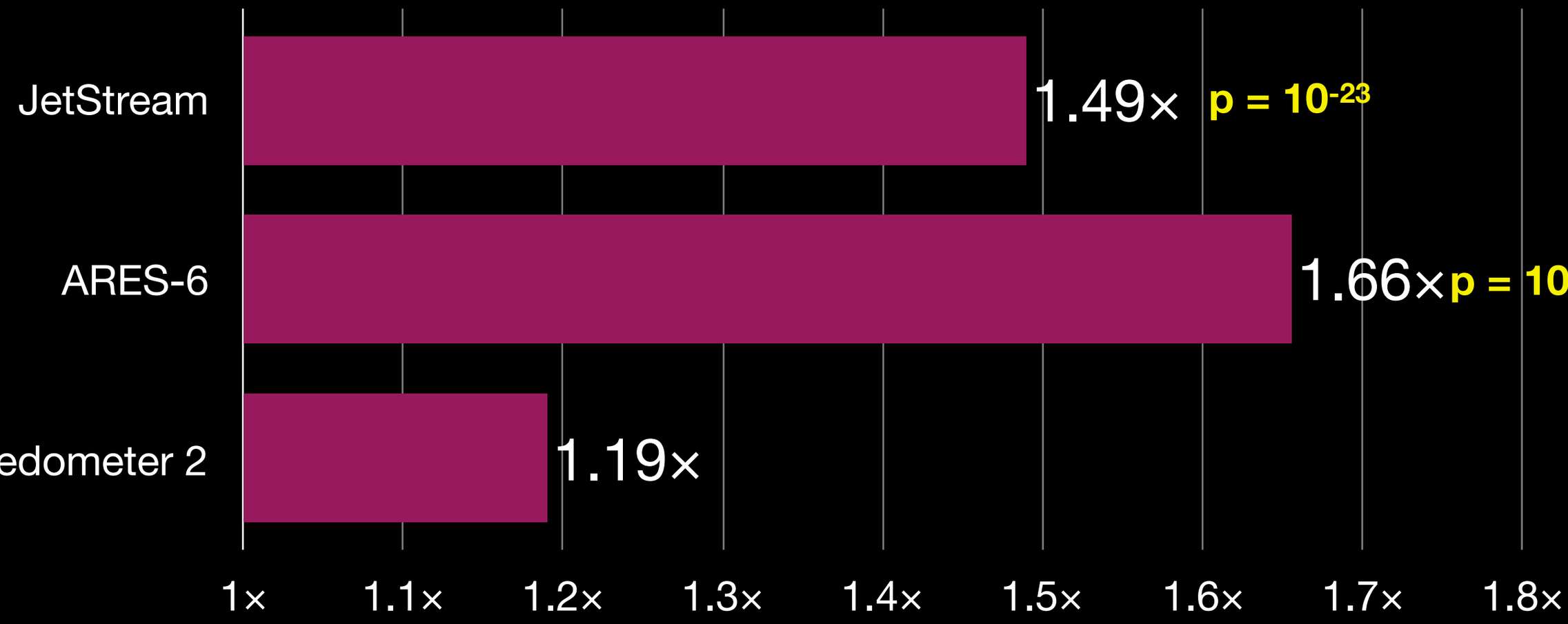
“Self” IC speed-up



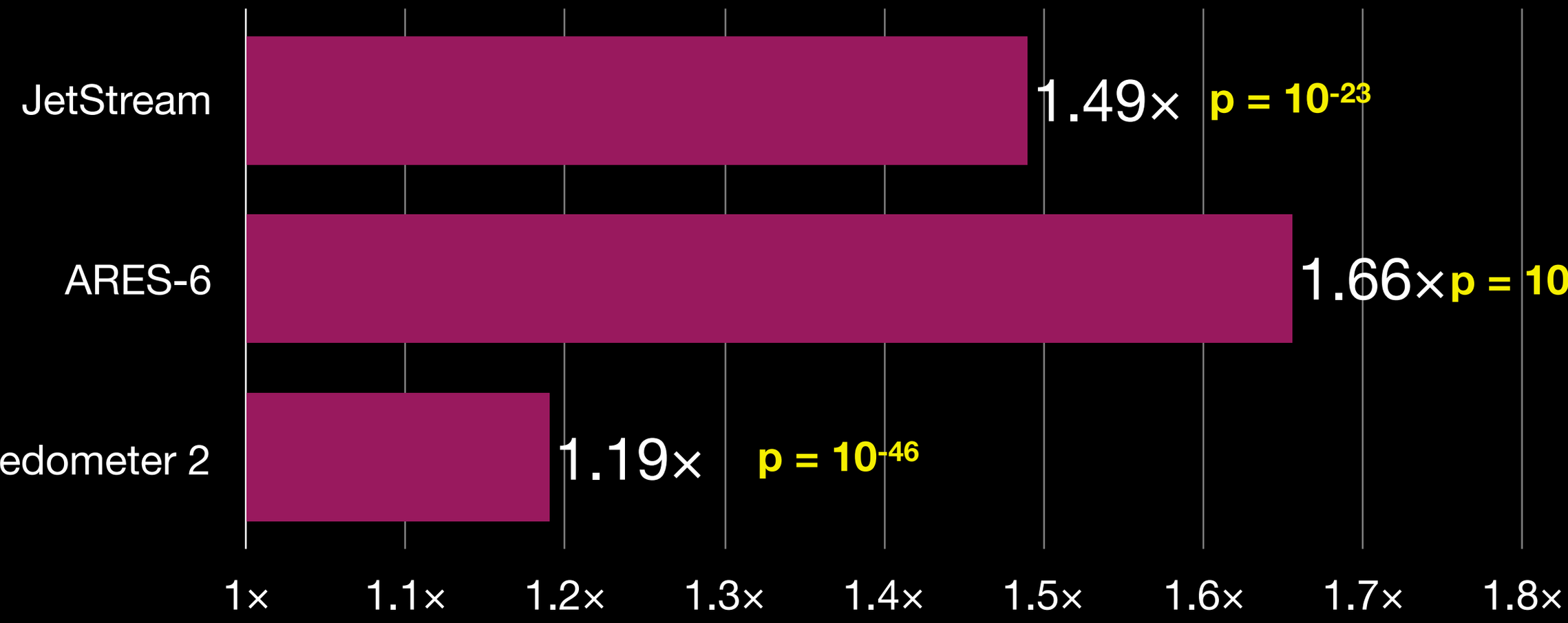
“Self” IC speed-up



“Self” IC speed-up



“Self” IC speed-up



Simple Inline Caching

- Interpreter edits instruction operands.
- JIT reserves a *nop sled* and patches it.

Prototypes

Prototypes

```
function Foo(f)
{
    this.f = f;
}
```

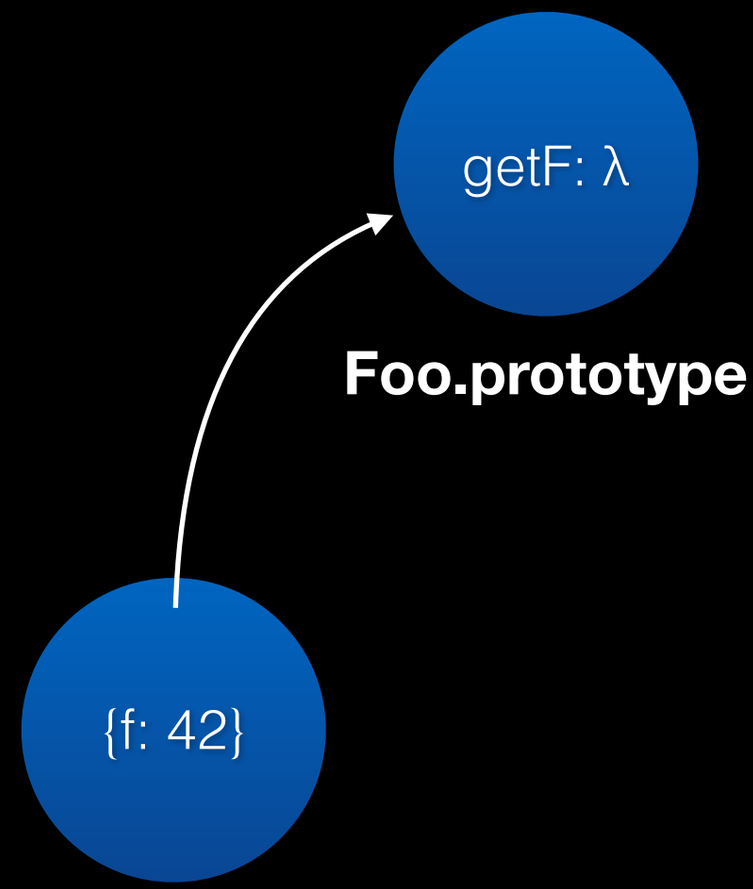
```
Foo.prototype.getF = function()
{
    return this.f;
}
```

Prototypes

```
function Foo(f)
{
    this.f = f;
}

Foo.prototype.getF = function()
{
    return this.f;
}
```

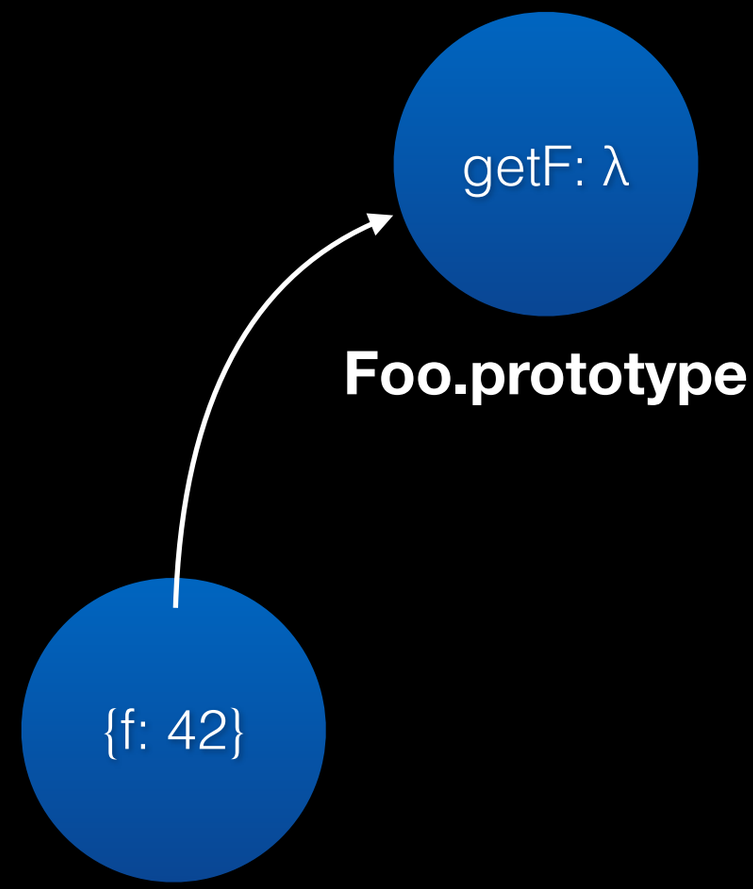
Prototypes

```
function Foo(f)
{
    this.f = f;
}

Foo.prototype.getF = function()
{
    return this.f;
}

var o = new Foo(42);
```



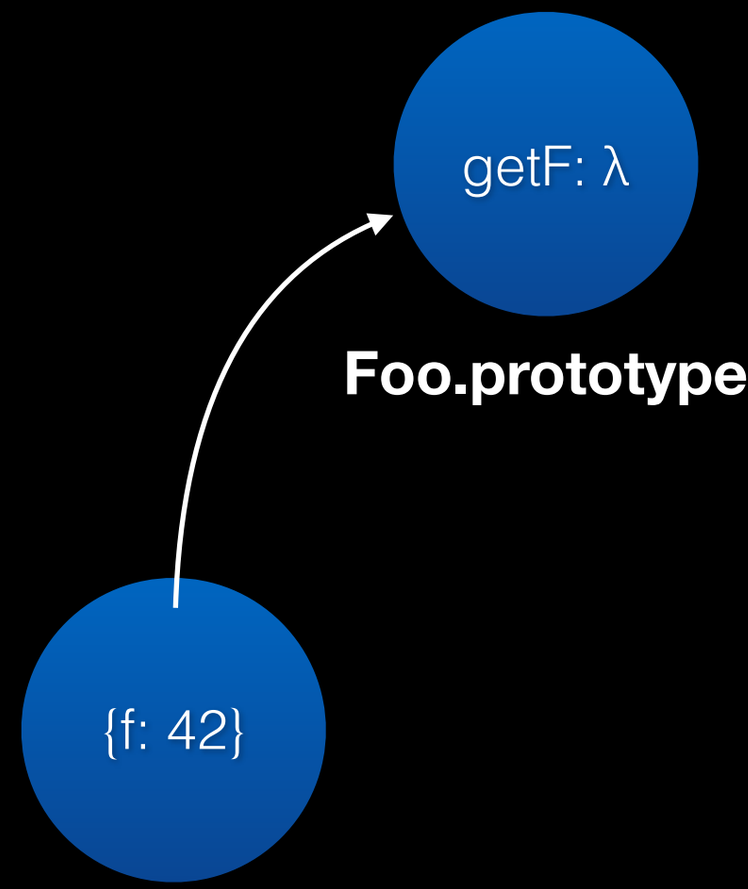
Prototypes

```
function Foo(f)
{
    this.f = f;
}

Foo.prototype.getF = function()
{
    return this.f;
}

var o = new Foo(42);

var tmp = o.getF();
```



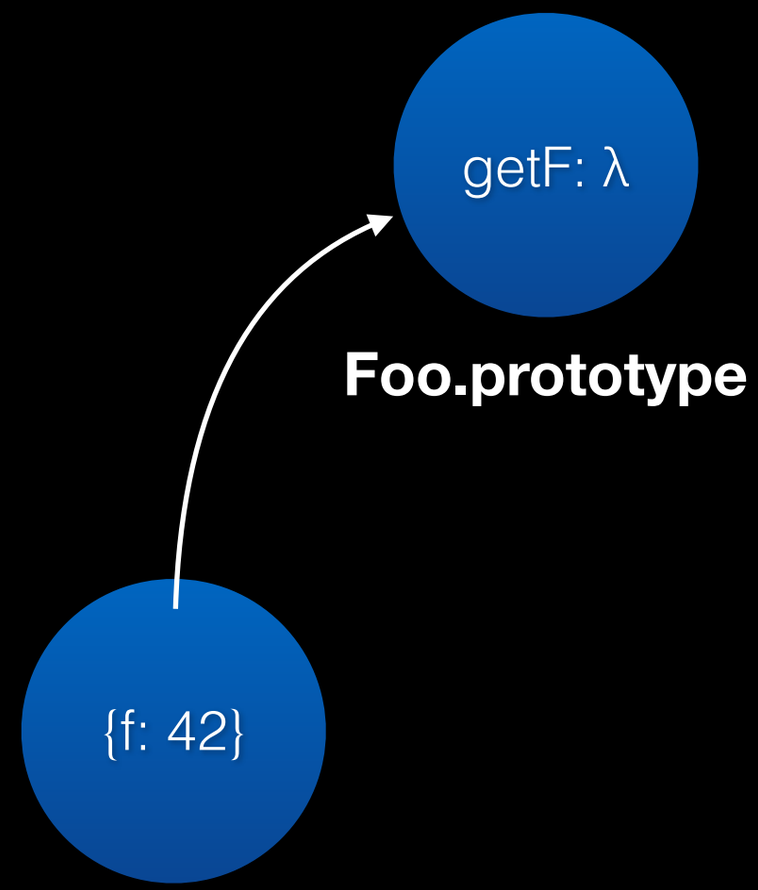
Prototypes

```
function Foo(f)
{
    this.f = f;
}

Foo.prototype.getF = function()
{
    return this.f;
}

var o = new Foo(42);

var tmp = o.getF();
```



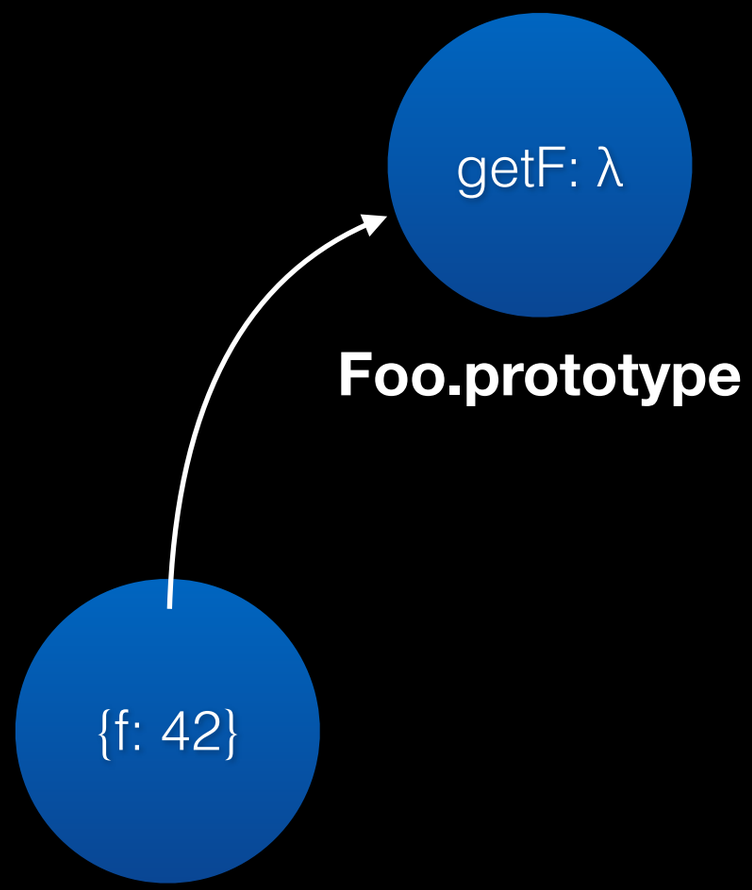
Prototypes

```
class Foo {
    constructor(f)
    {
        this.f = f;
    }

    getF()
    {
        return this.f;
    }
}

var o = new Foo(42);

var tmp = o.getF();
```



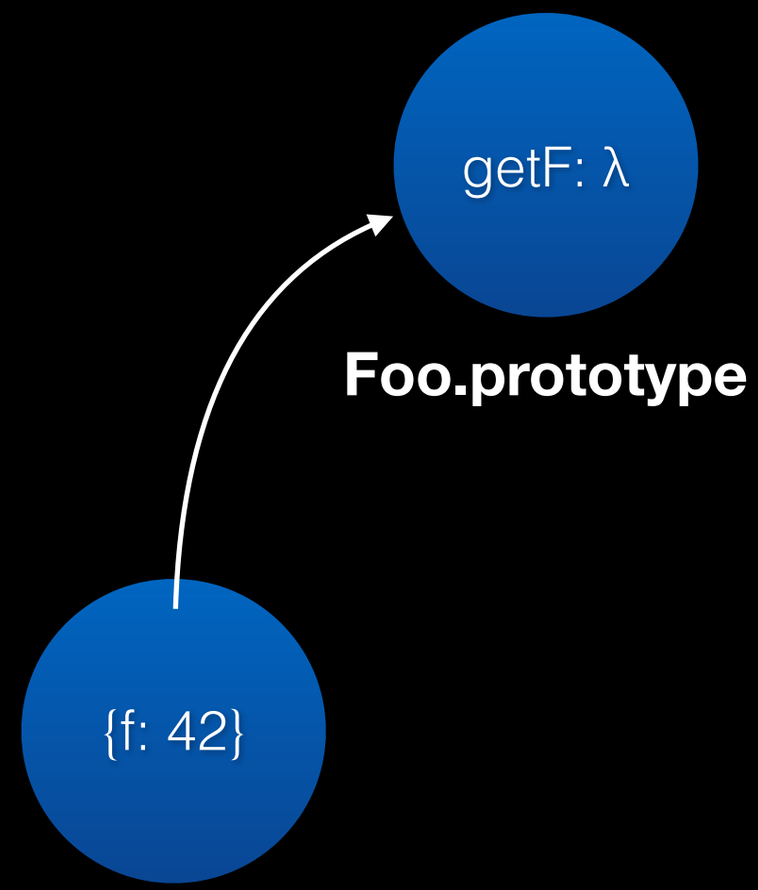
Prototypes

```
class Foo {
    constructor(f)
    {
        this.f = f;
    }

    getF()
    {
        return this.f;
    }
}

var o = new Foo(42);

var tmp = o.getF();
```



Prototypes

```
class Foo {
    constructor(f)
    {
        this.f = f;
    }

    getF()
    {
        return this.f;
    }
}

var o = new Foo(42);

var tmp = o.getF();
```



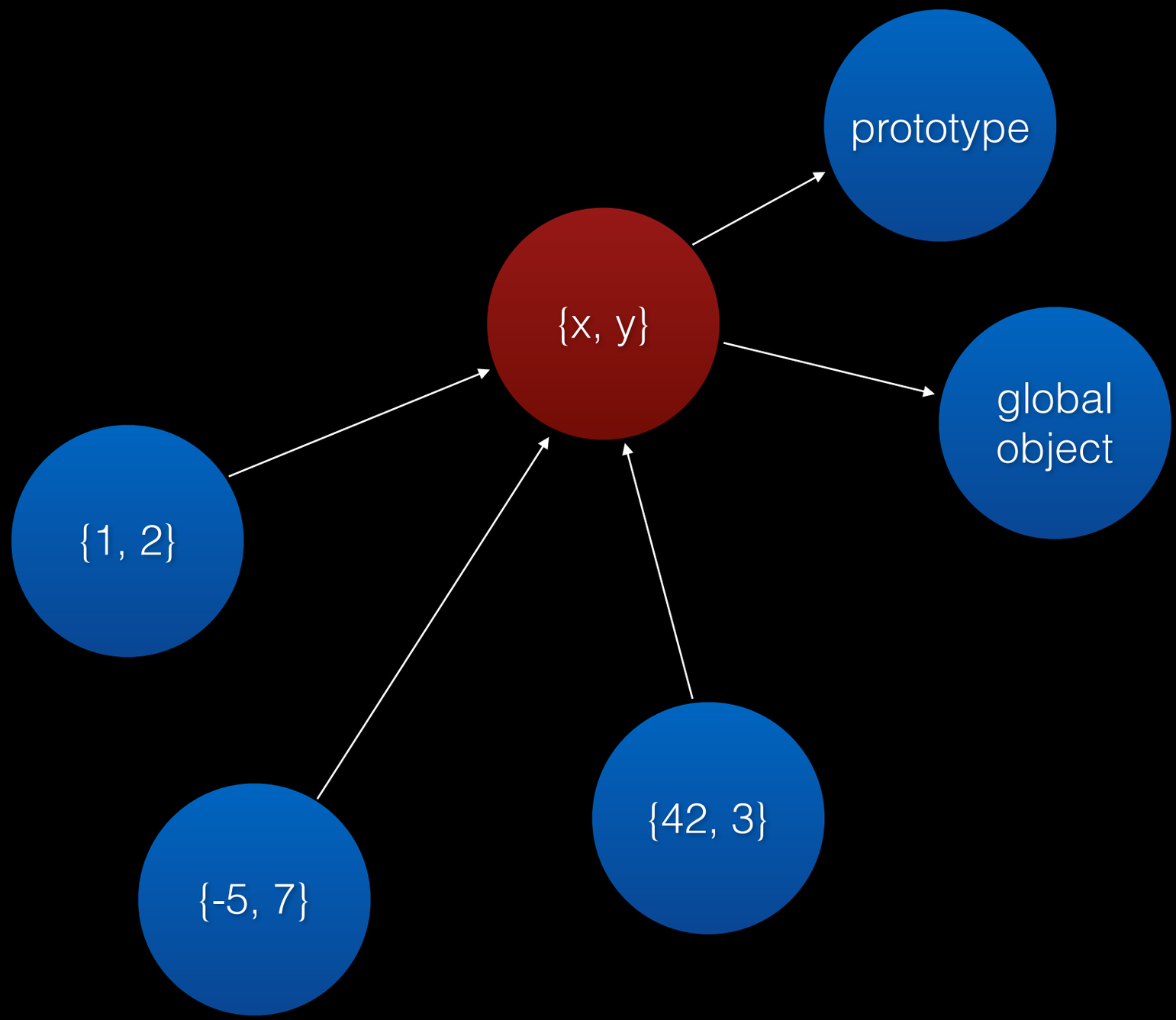
Prototype Inline Caching

- Goals:
  - o.getF should be fast
  - o.getF() should be inlineable
  - o.newField = value should be fast

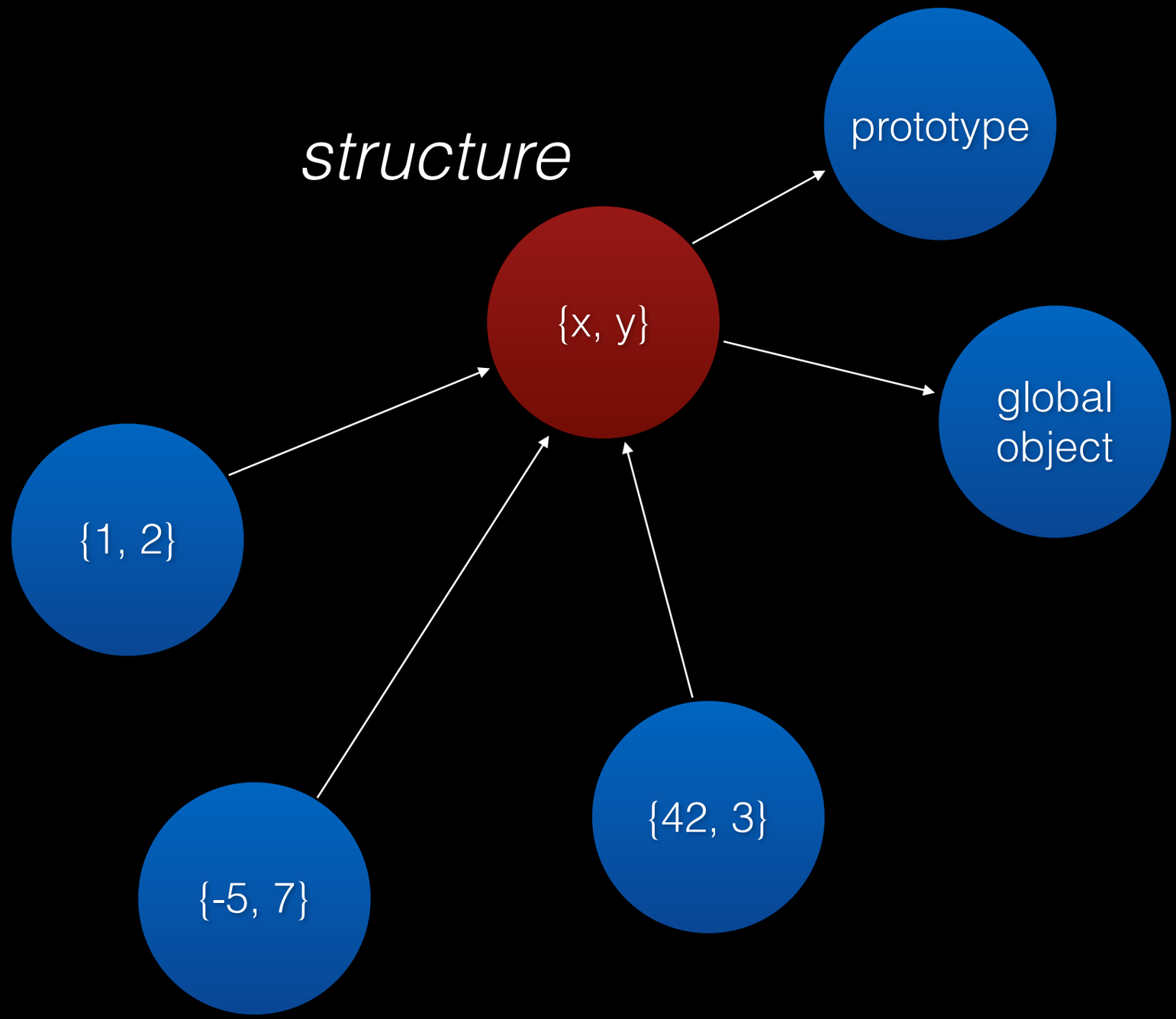
Prototype Inline Caching

- Mono proto
- Transition watchpoint
- Replacement watchpoint

Mono Proto

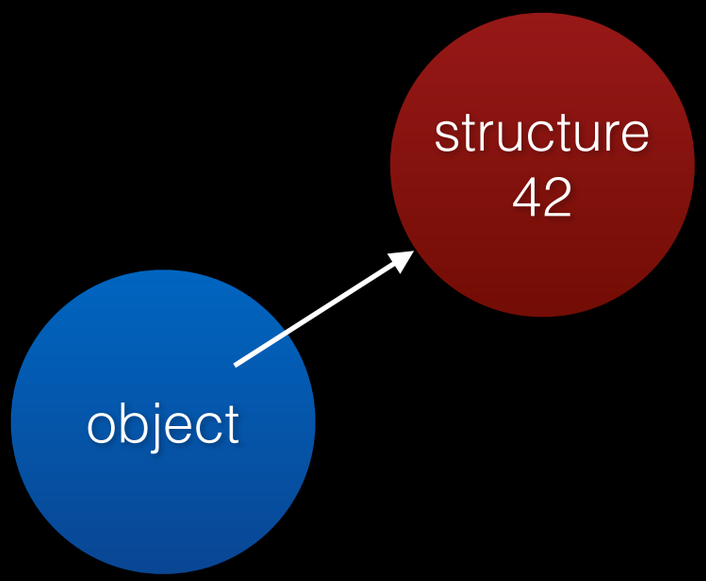


Mono Proto

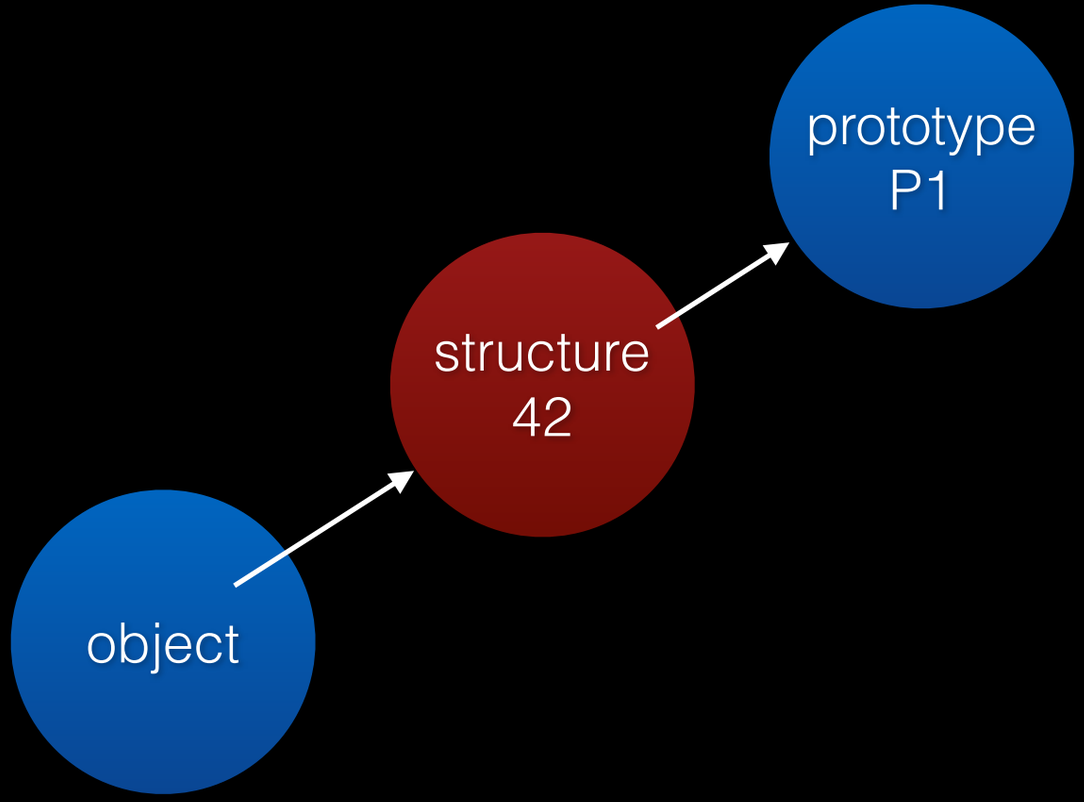


Proto Chain

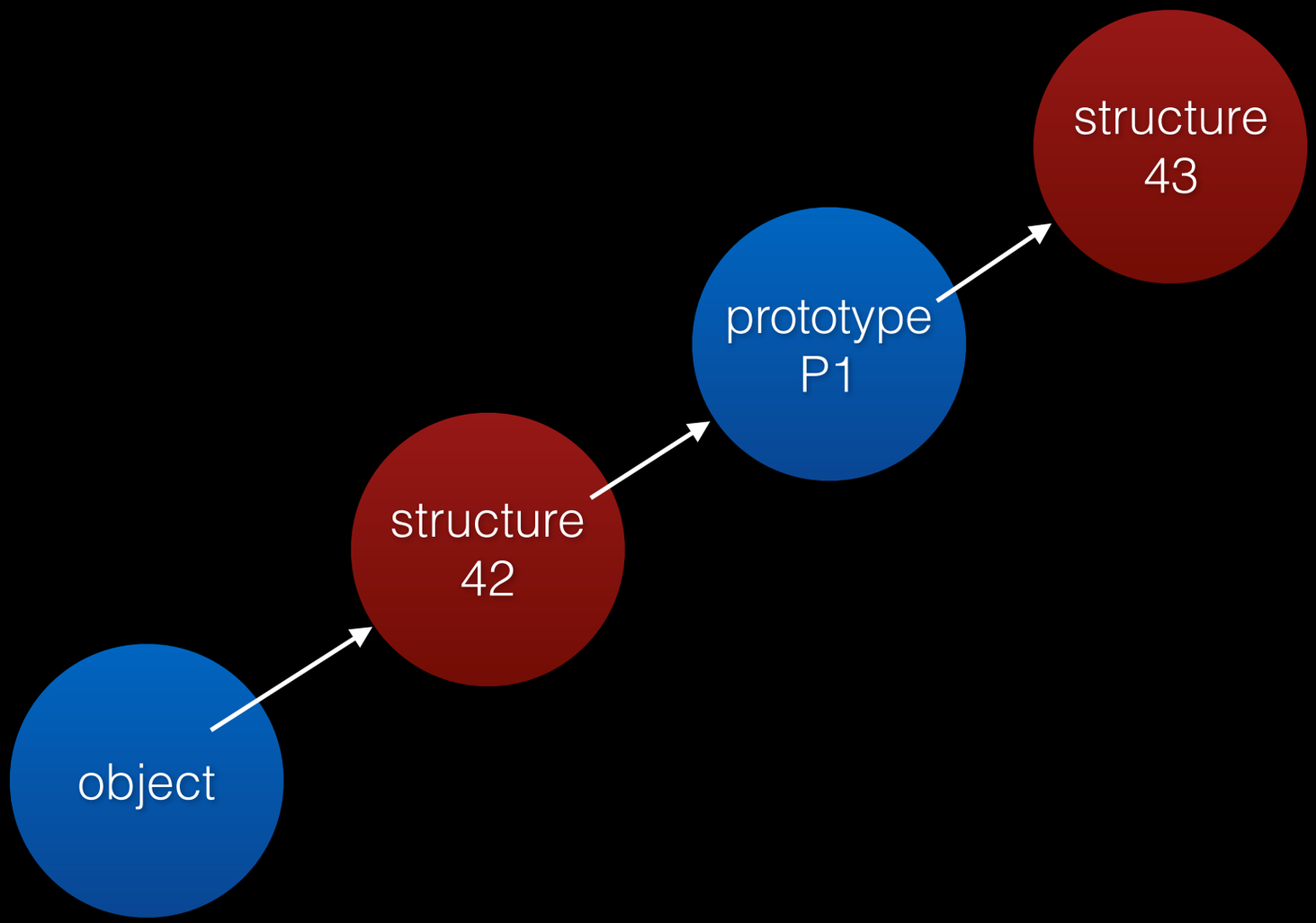
Proto Chain



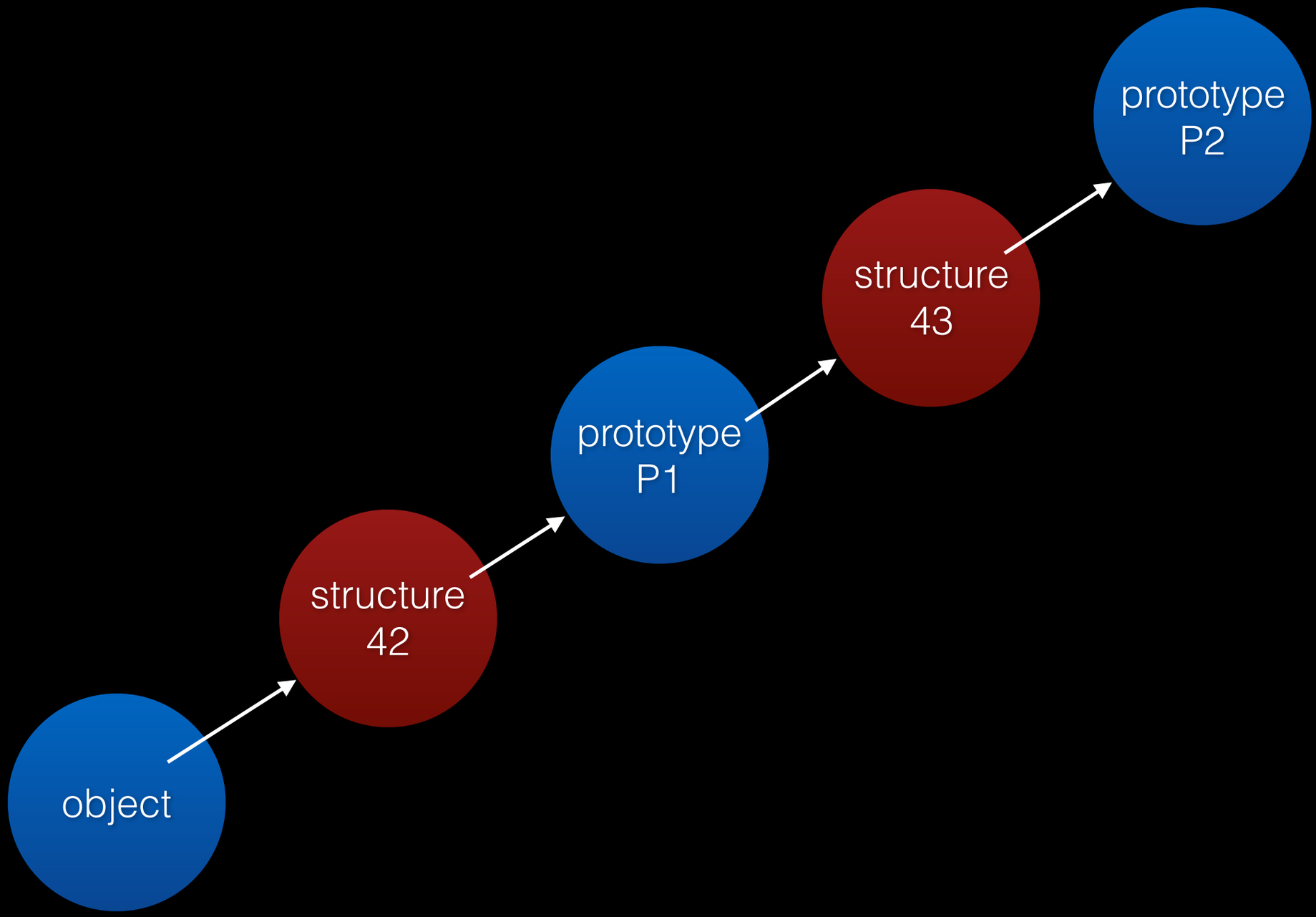
Proto Chain



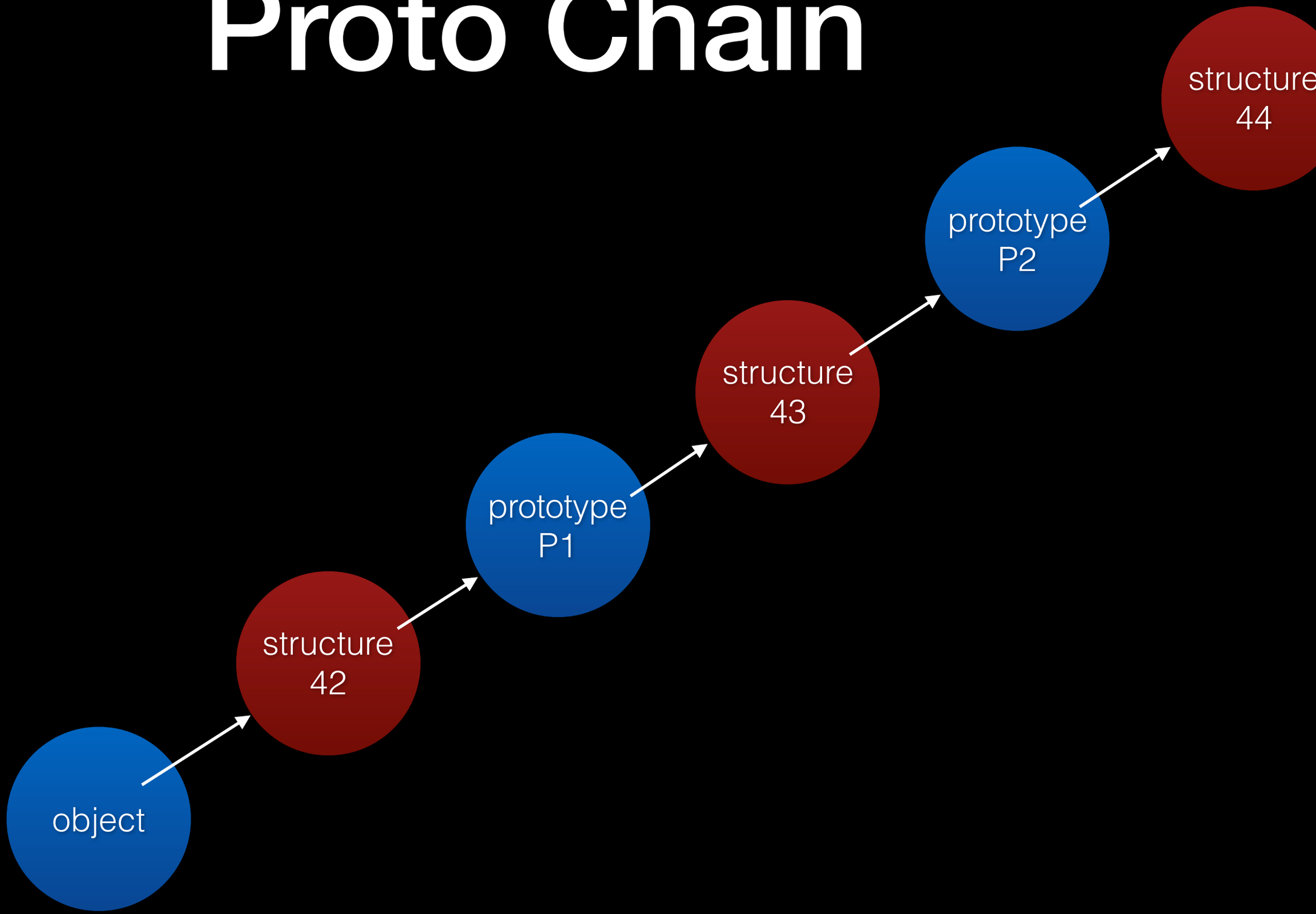
Proto Chain



Proto Chain

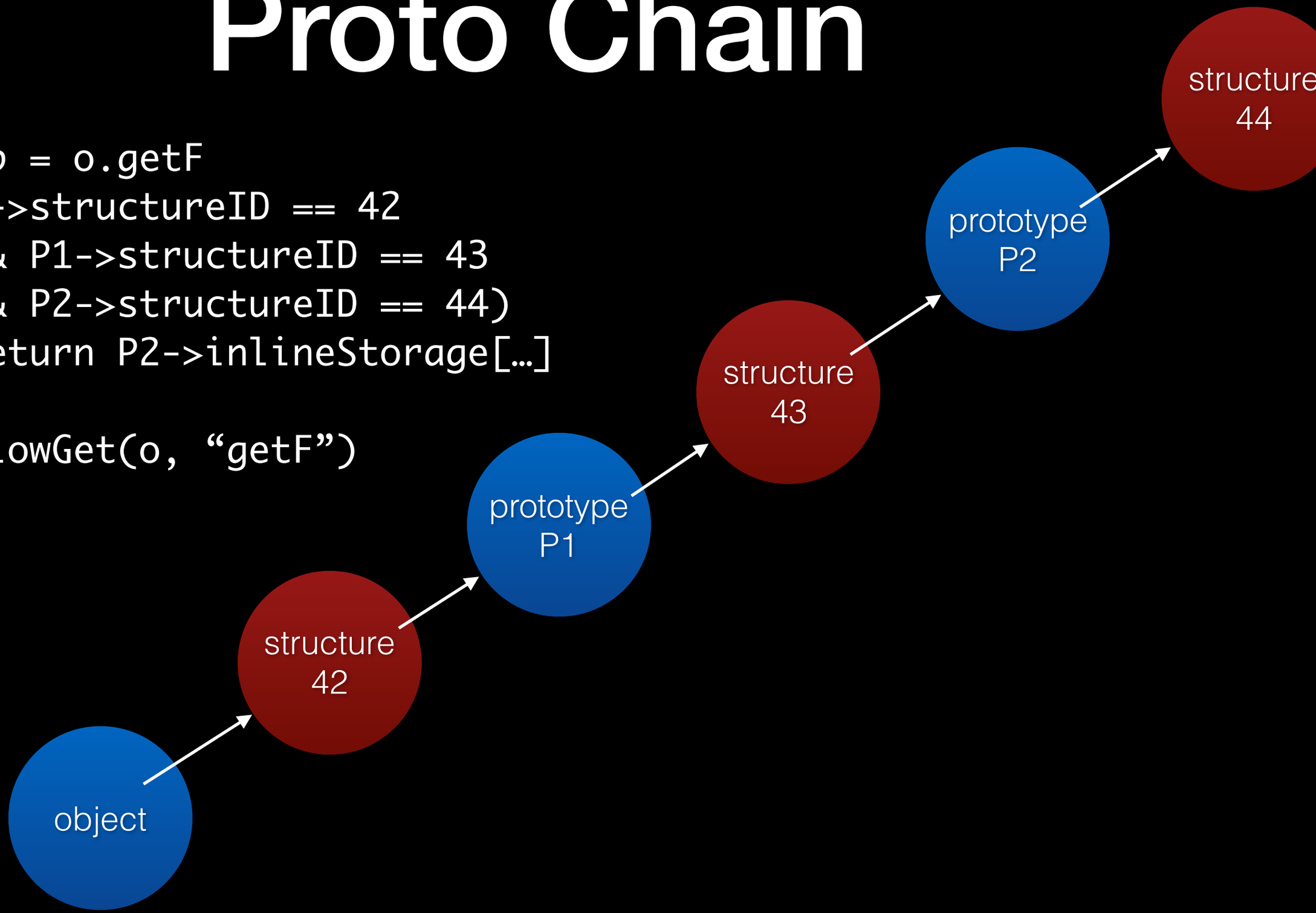

prototype
P2
structure
43
prototype
P1
structure
42
object

Proto Chain


structure
44
prototype
P2
structure
43
prototype
P1
structure
42
object

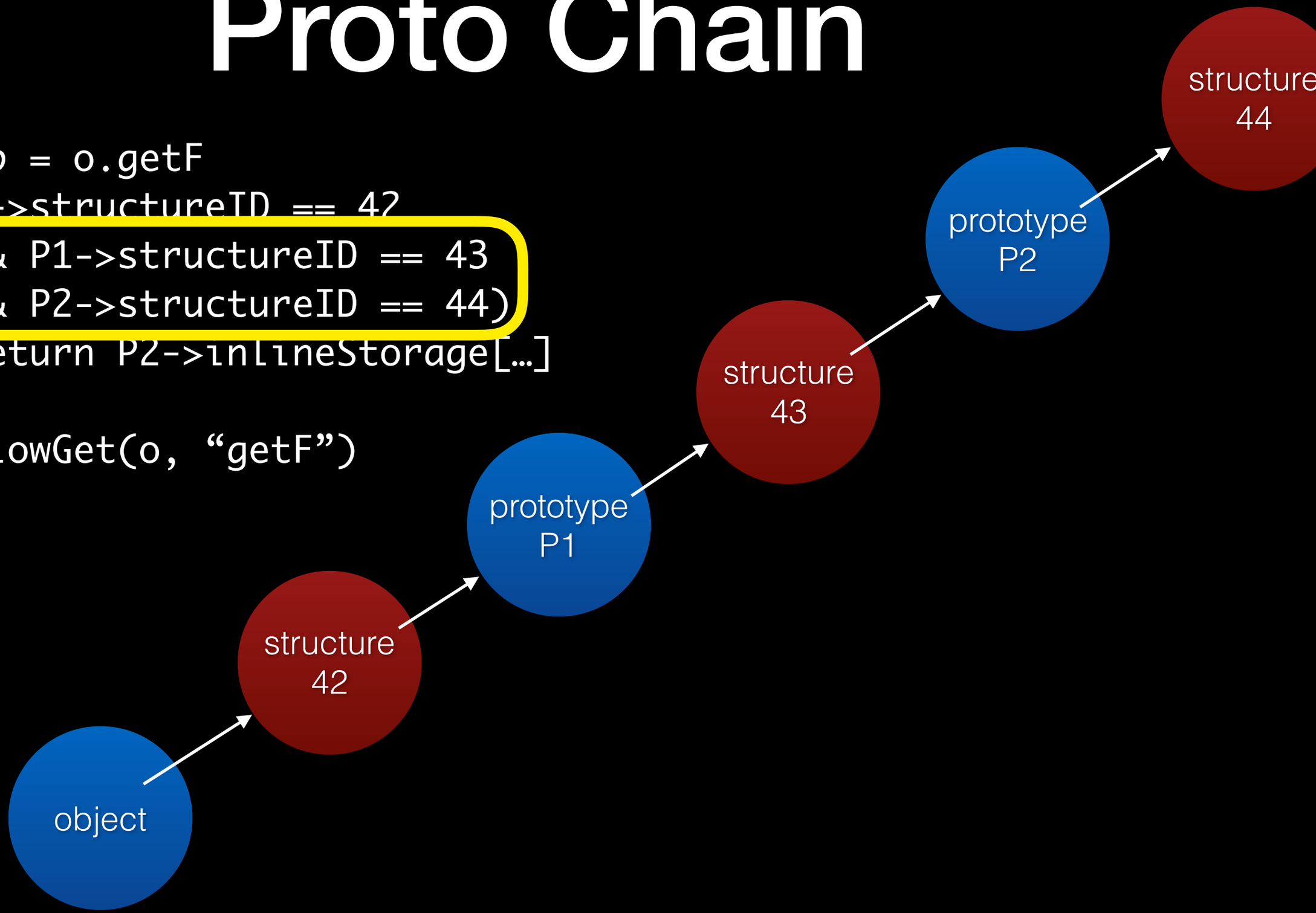
Proto Chain

```
// tmp = o.getF
if (o->structureID == 42
    && P1->structureID == 43
    && P2->structureID == 44)
    return P2->inlineStorage[…]
else
slowGet(o, “getF”)
```



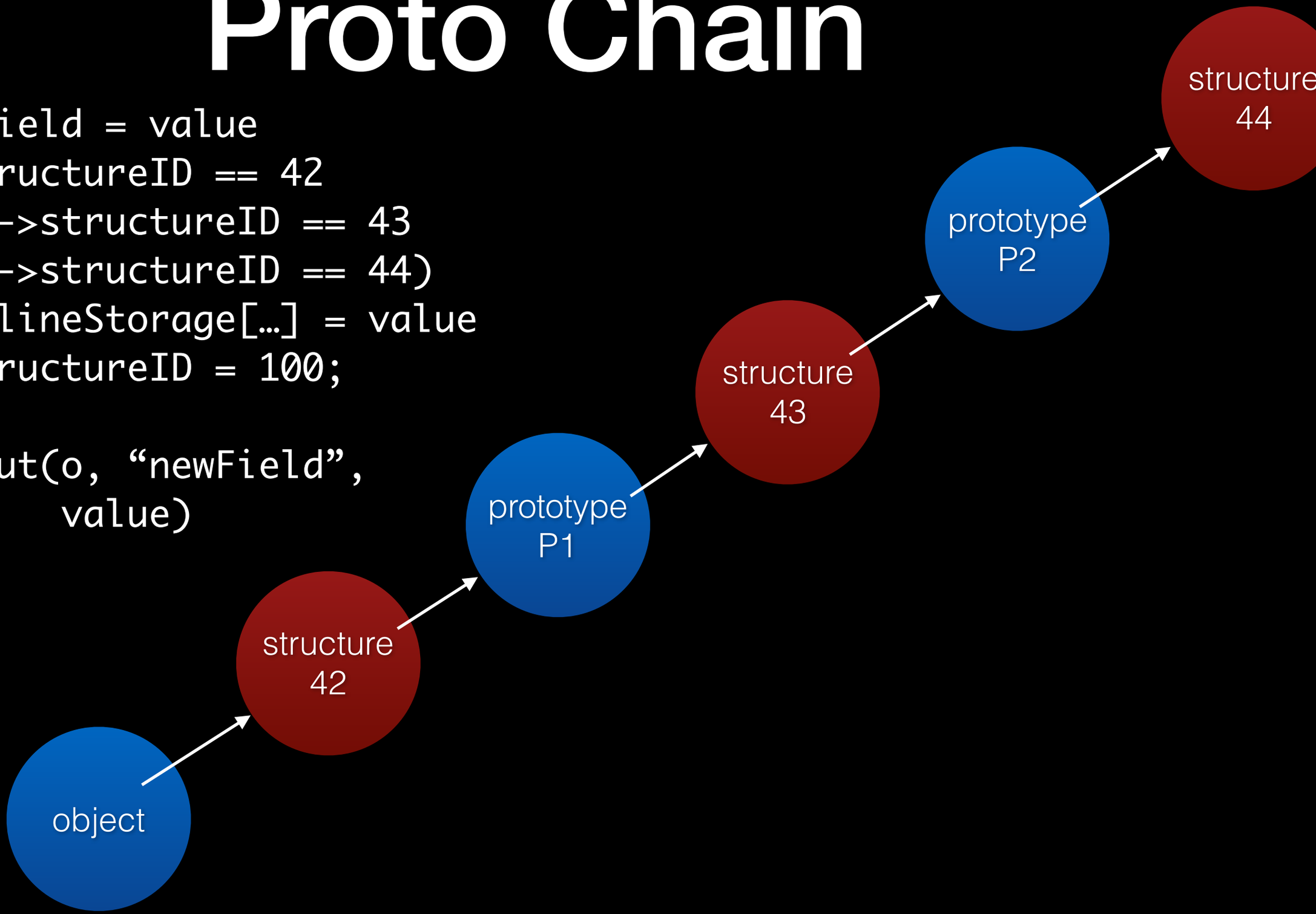
Proto Chain

```
// tmp = o.getF
if (o->structureID == 42
    && P1->structureID == 43
    && P2->structureID == 44)
     return P2->inlineStorage[…]
else
    slowGet(o, “getF”)
```



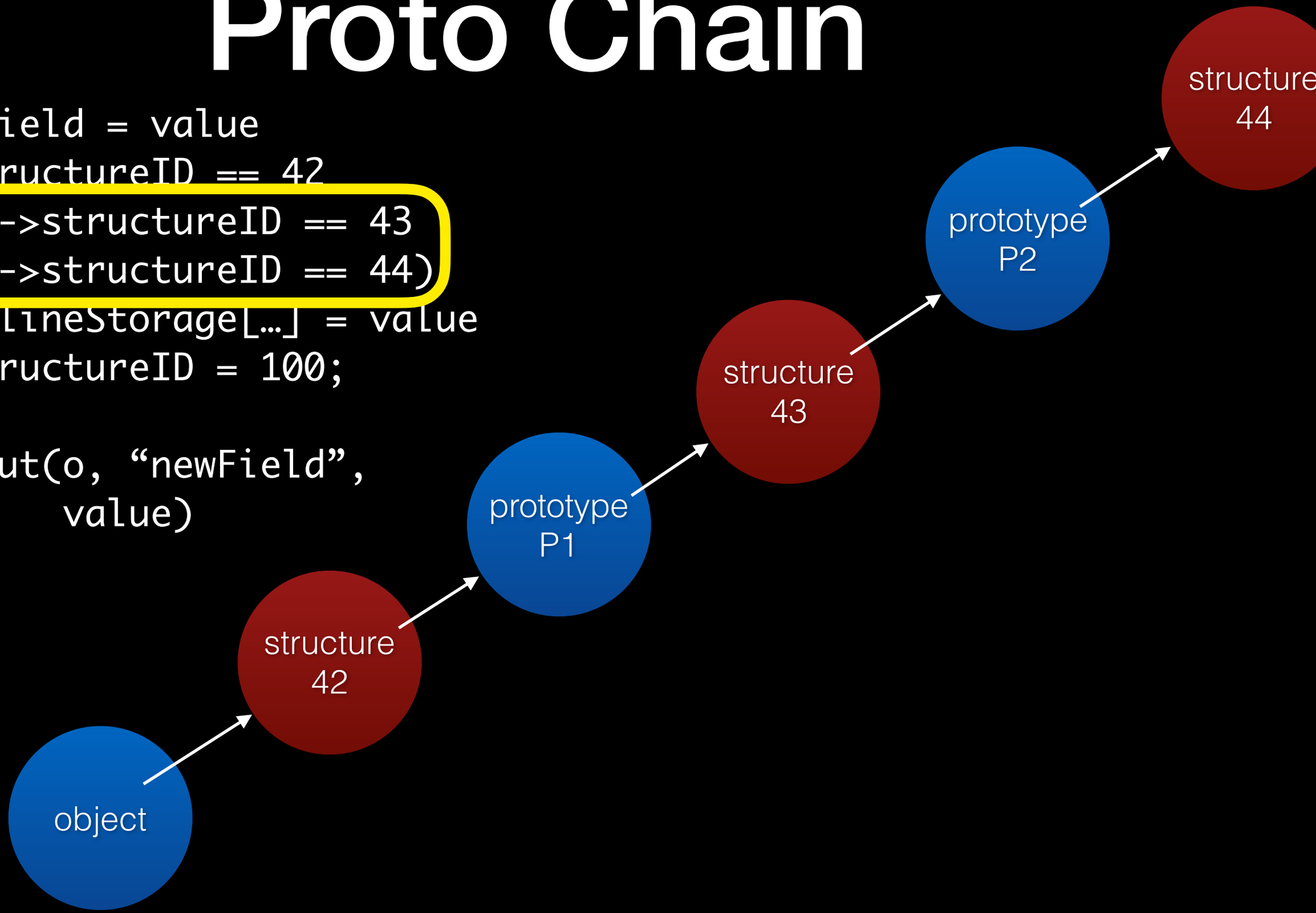
Proto Chain

```
// o.newField = value
if (o->structureID == 42
    && P1->structureID == 43
    && P2->structureID == 44)
    o->inlineStorage[…] = value
    o->structureID = 100;
else
    slowPut(o, “newField”,
            value)
```

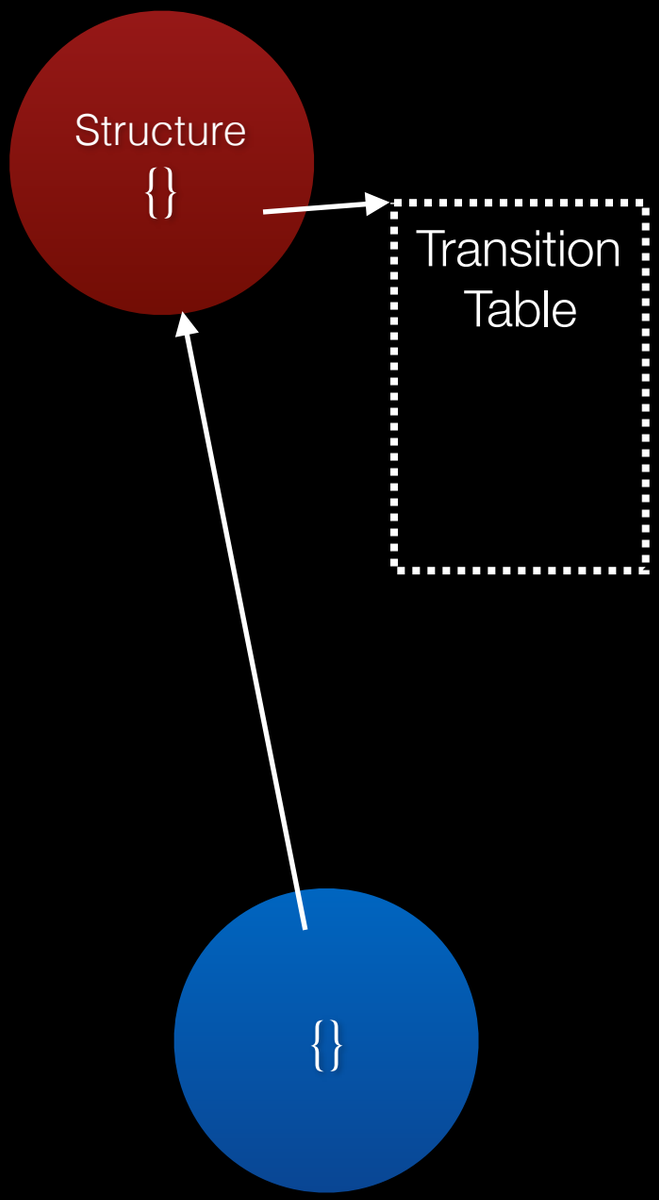


Proto Chain

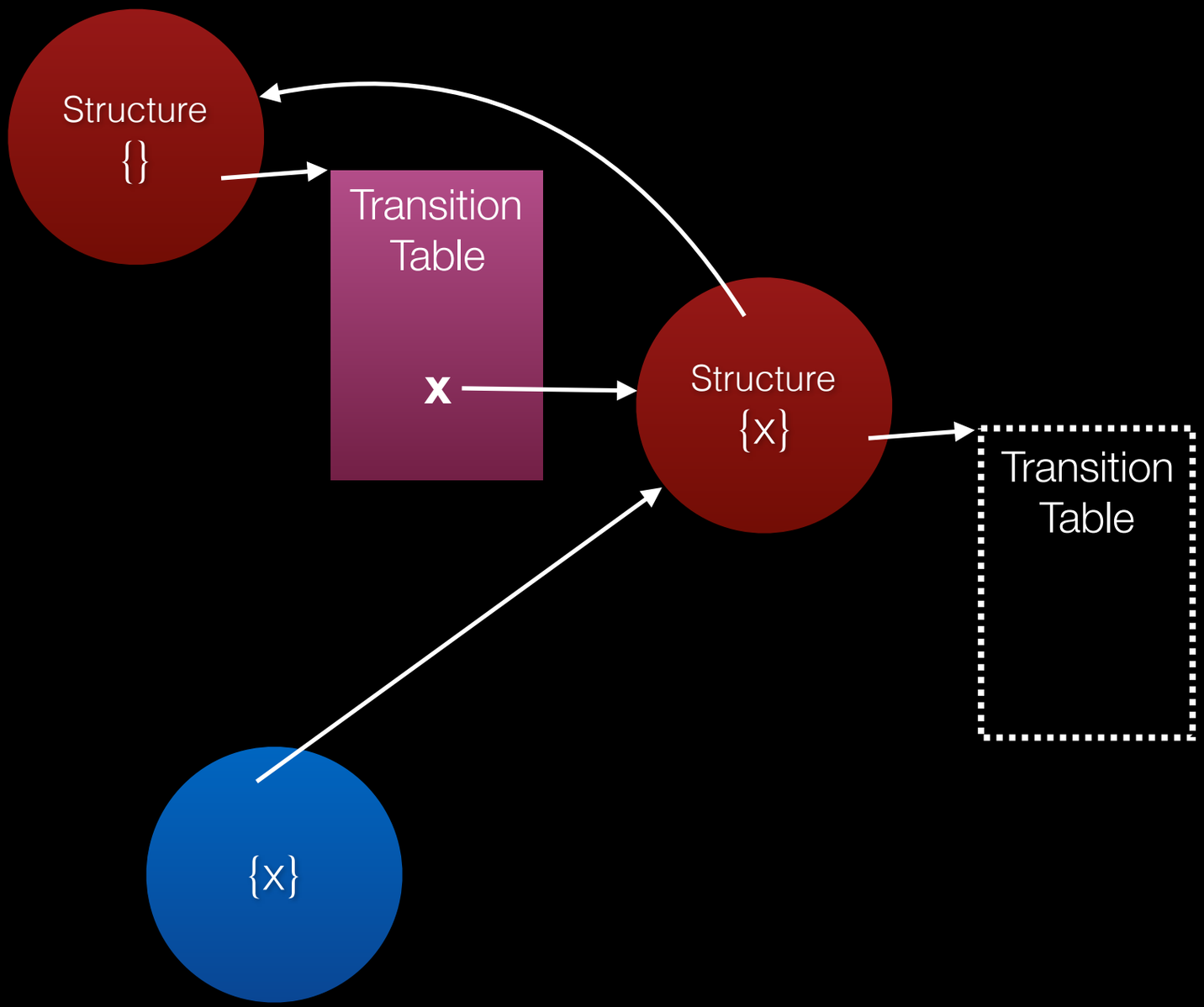
```
// o.newField = value
if (o->structureID == 42
    && P1->structureID == 43
    && P2->structureID == 44)
    o->inlineStorage[…] = value
    o->structureID = 100;
else
    slowPut(o, “newField”,
            value)
```



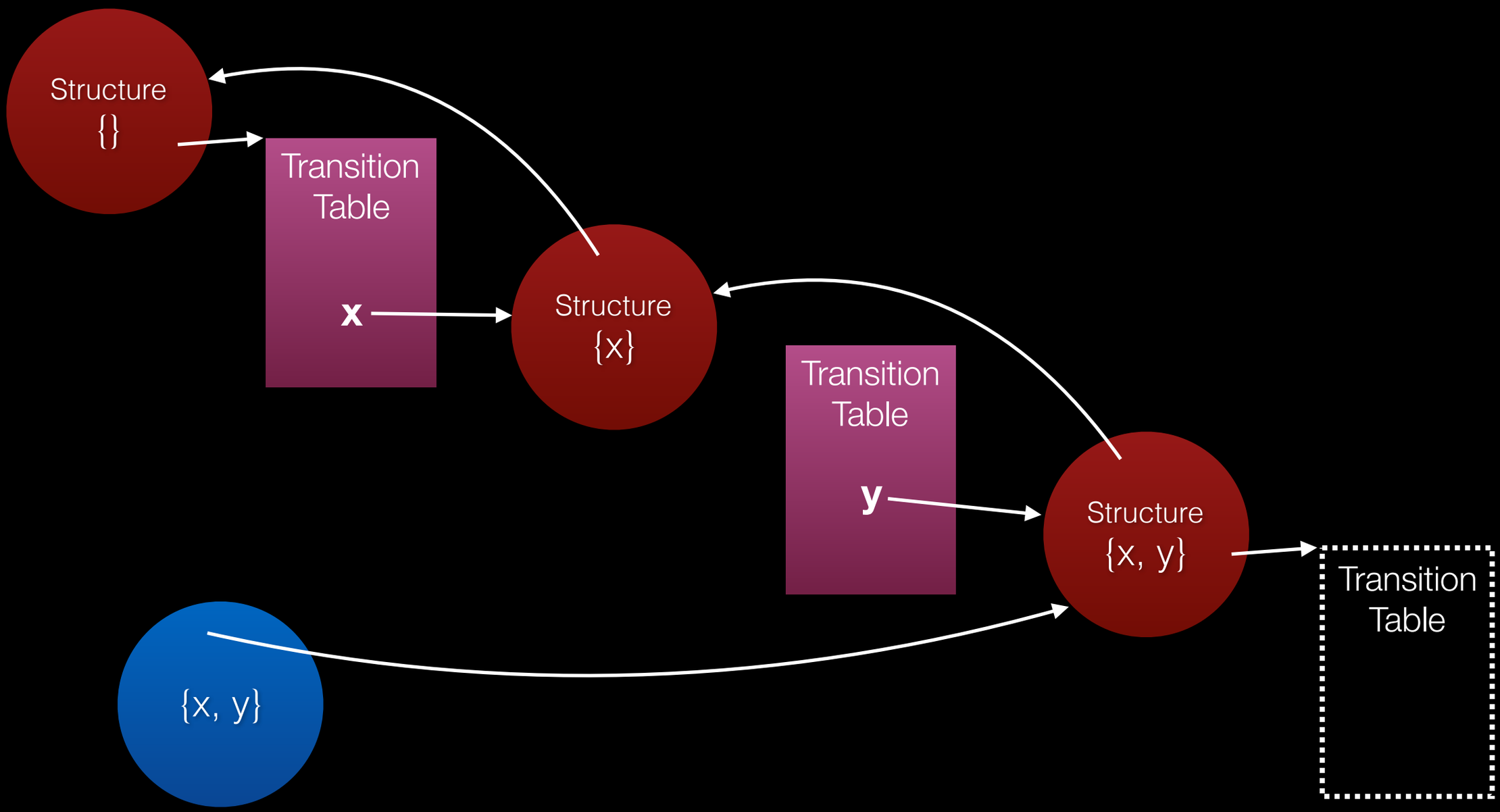
Transition Hash Consing



Transition Hash Consing

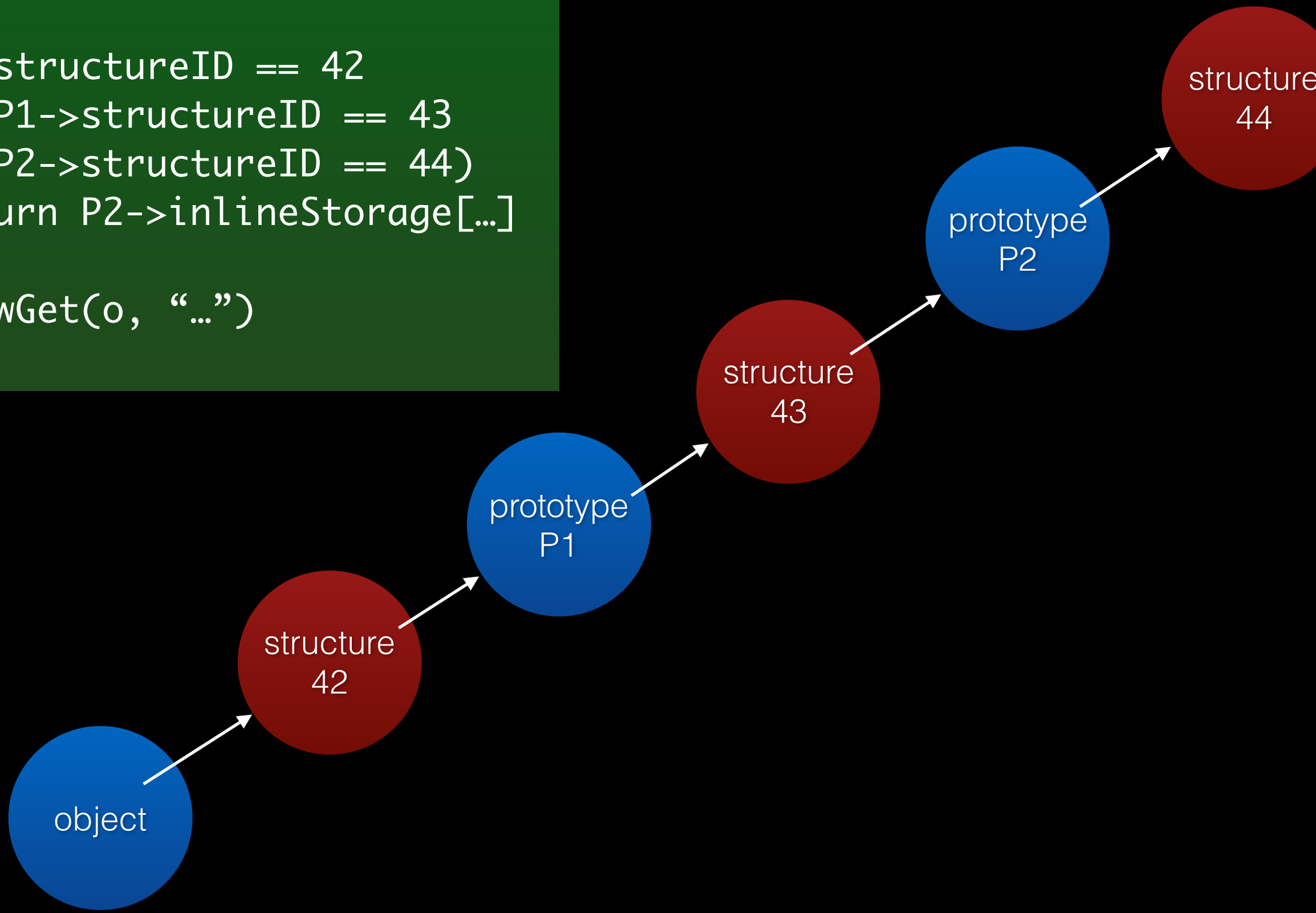


Transition Hash Consing



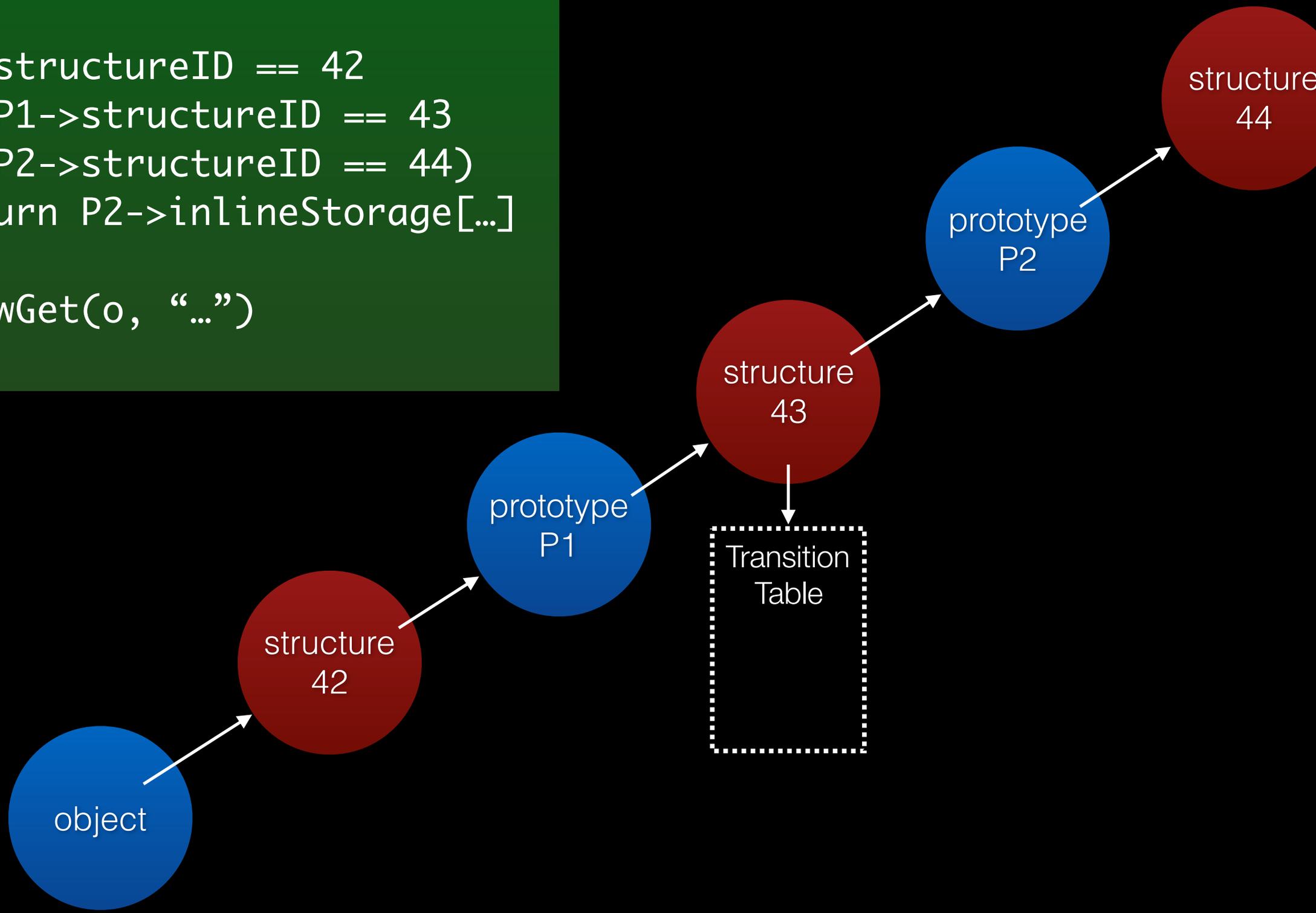
Optimized Code

```
if (o->structureID == 42
    && P1->structureID == 43
    && P2->structureID == 44)
    return P2->inlineStorage[…]
else
    slowGet(o, “…”)
```



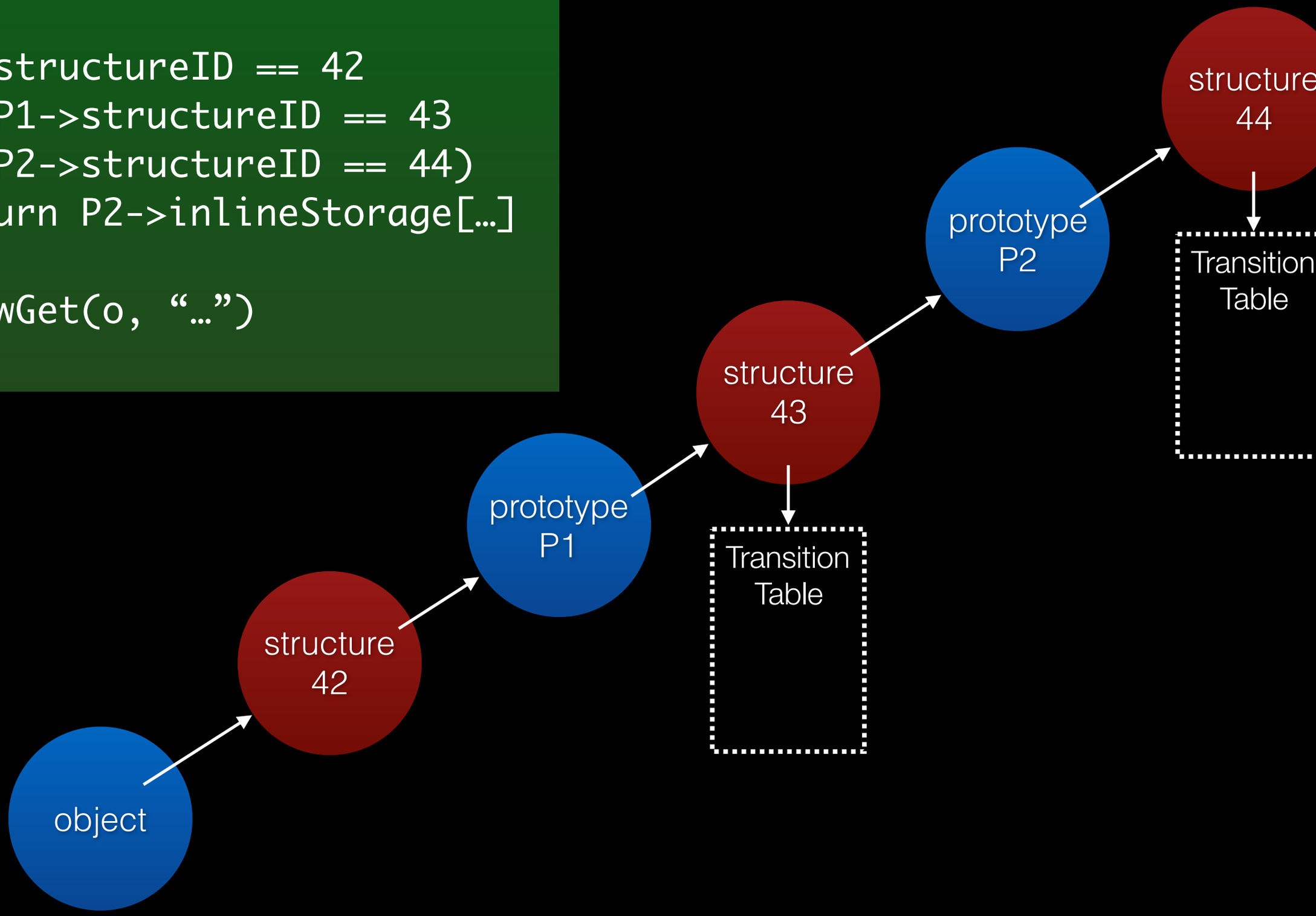
Optimized Code

```
if (o->structureID == 42
    && P1->structureID == 43
    && P2->structureID == 44)
    return P2->inlineStorage[…]
else
    slowGet(o, “…”)
```



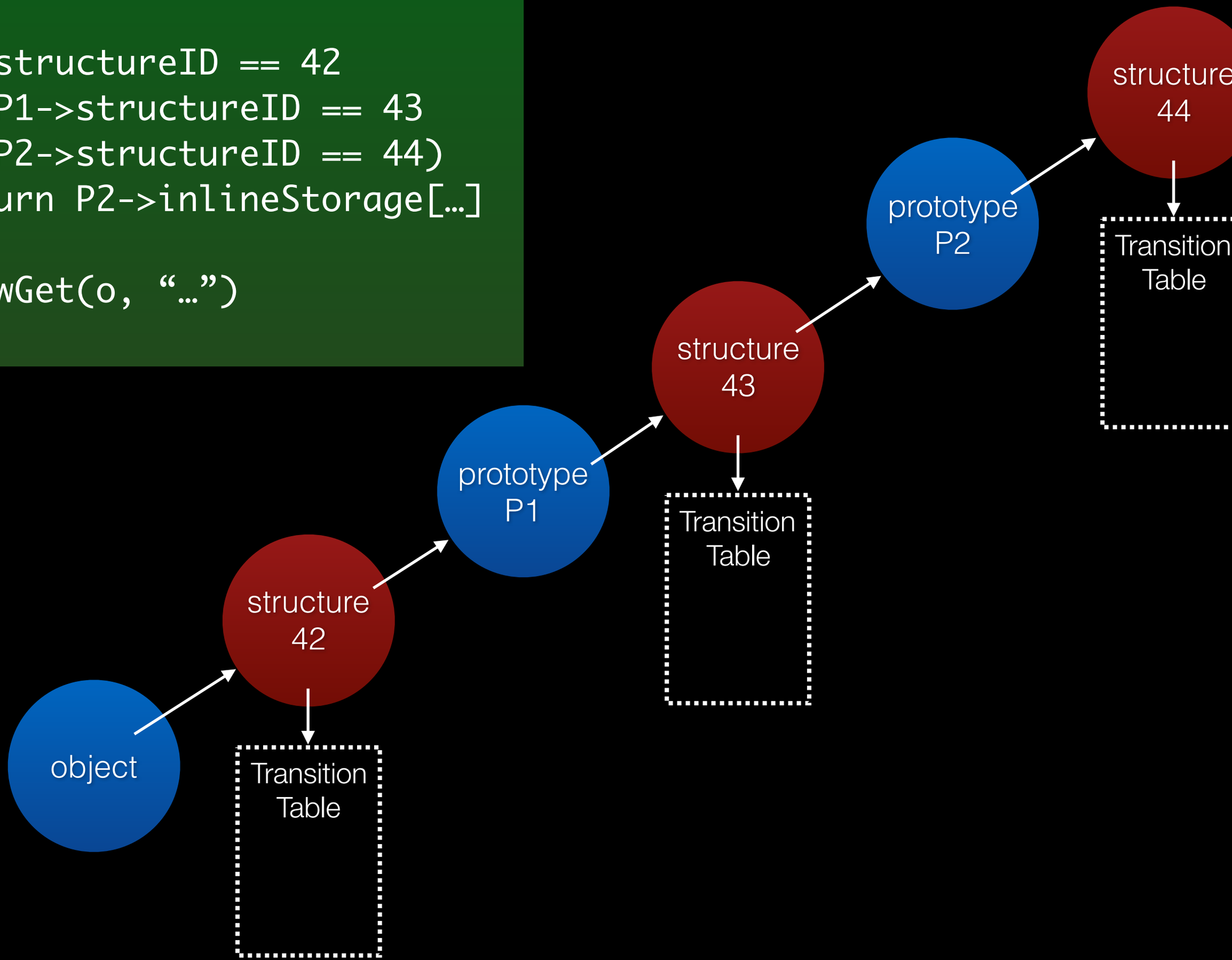
Optimized Code

```
if (o->structureID == 42
    && P1->structureID == 43
    && P2->structureID == 44)
    return P2->inlineStorage[…]
else
    slowGet(o, “…”)
```



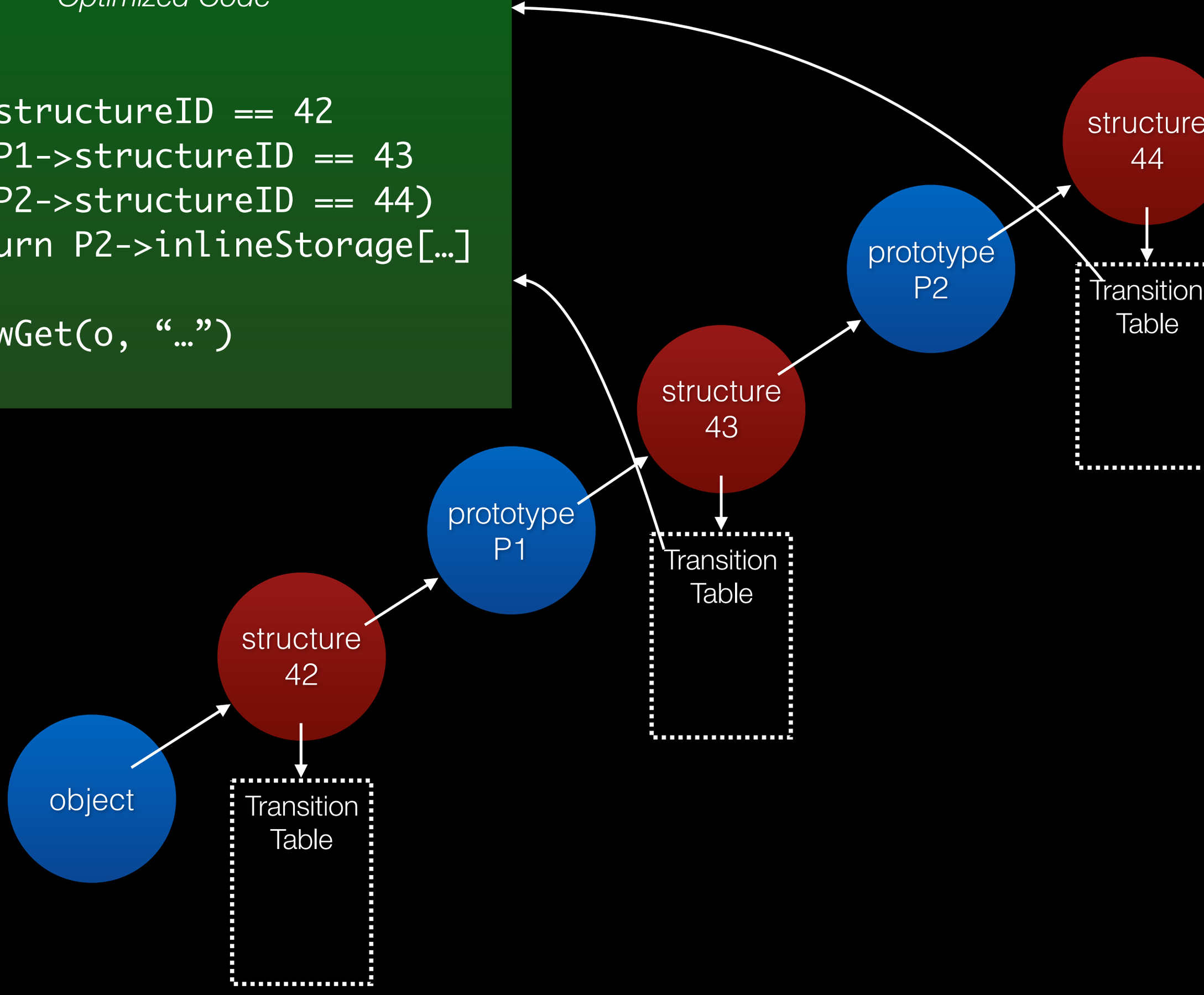
Optimized Code

```
if (o->structureID == 42
    && P1->structureID == 43
    && P2->structureID == 44)
    return P2->inlineStorage[…]
else
    slowGet(o, “…”)
```



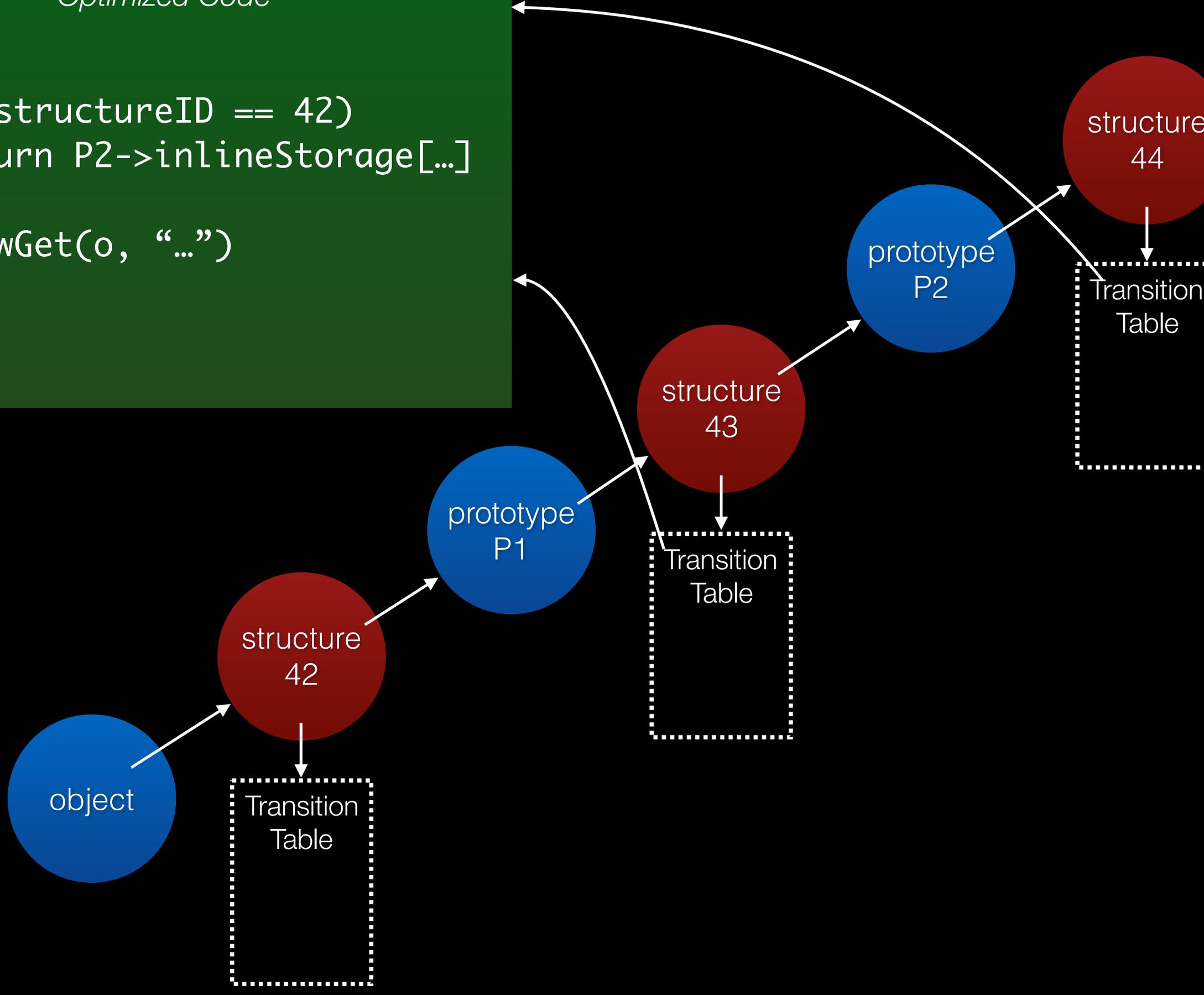
Optimized Code

```
if (o->structureID == 42
    && P1->structureID == 43
    && P2->structureID == 44)
    return P2->inlineStorage[…]
else
    slowGet(o, “…”)
```



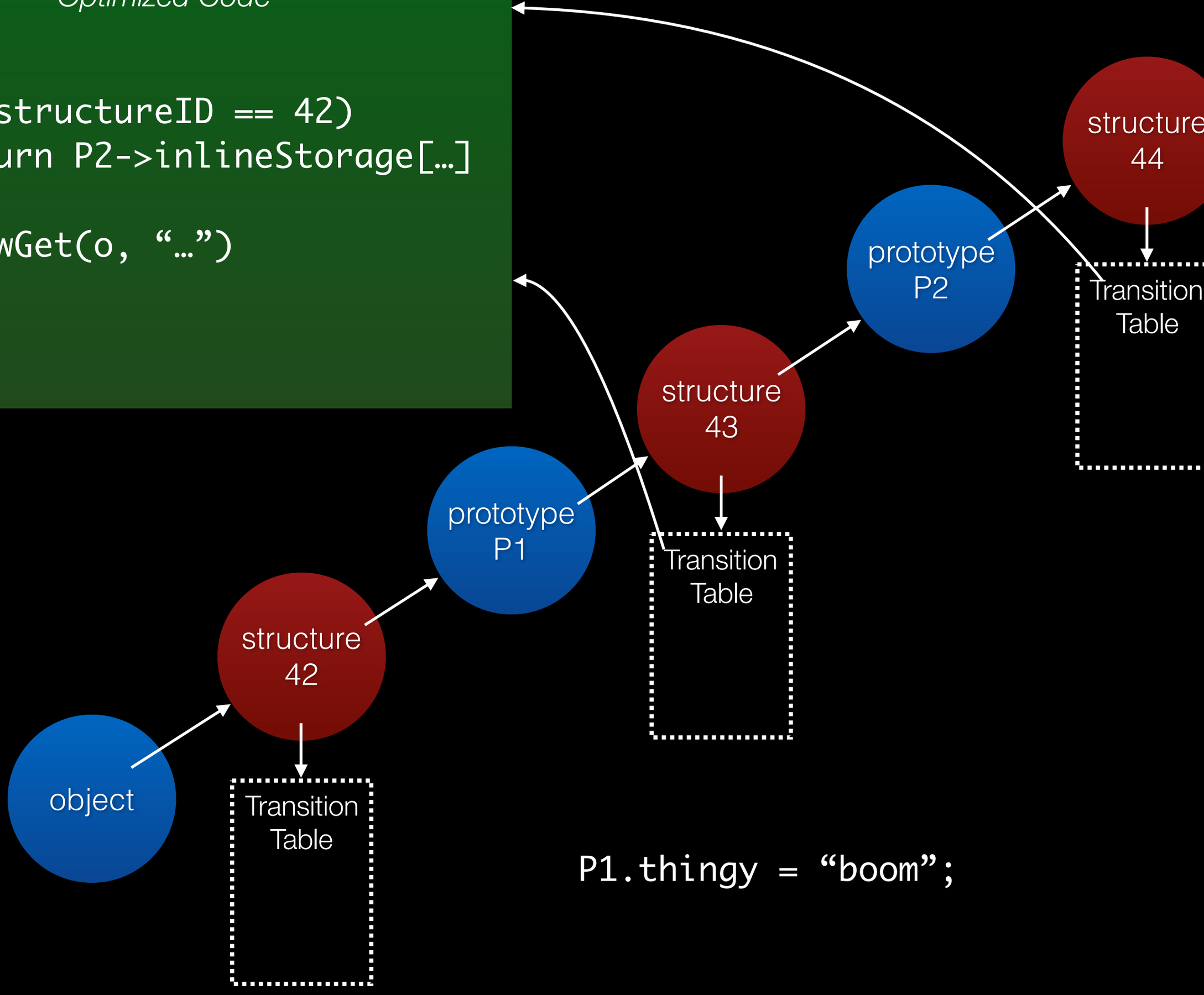
Optimized Code

```
if (o->structureID == 42)
    return P2->inlineStorage[…]
else
    slowGet(o, “…”)
```



Optimized Code

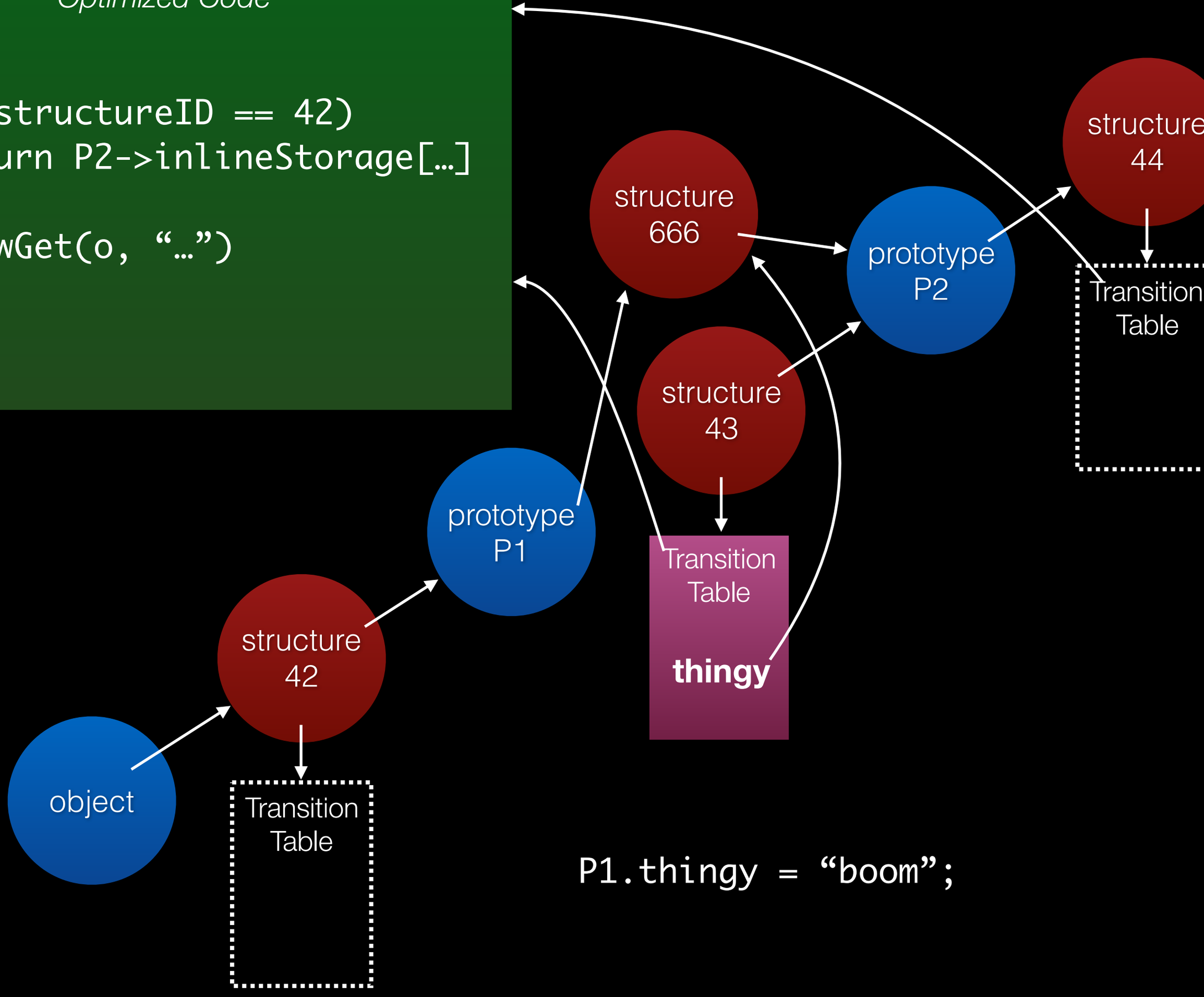
```
if (o->structureID == 42)
    return P2->inlineStorage[…]
else
    slowGet(o, “…”)
```



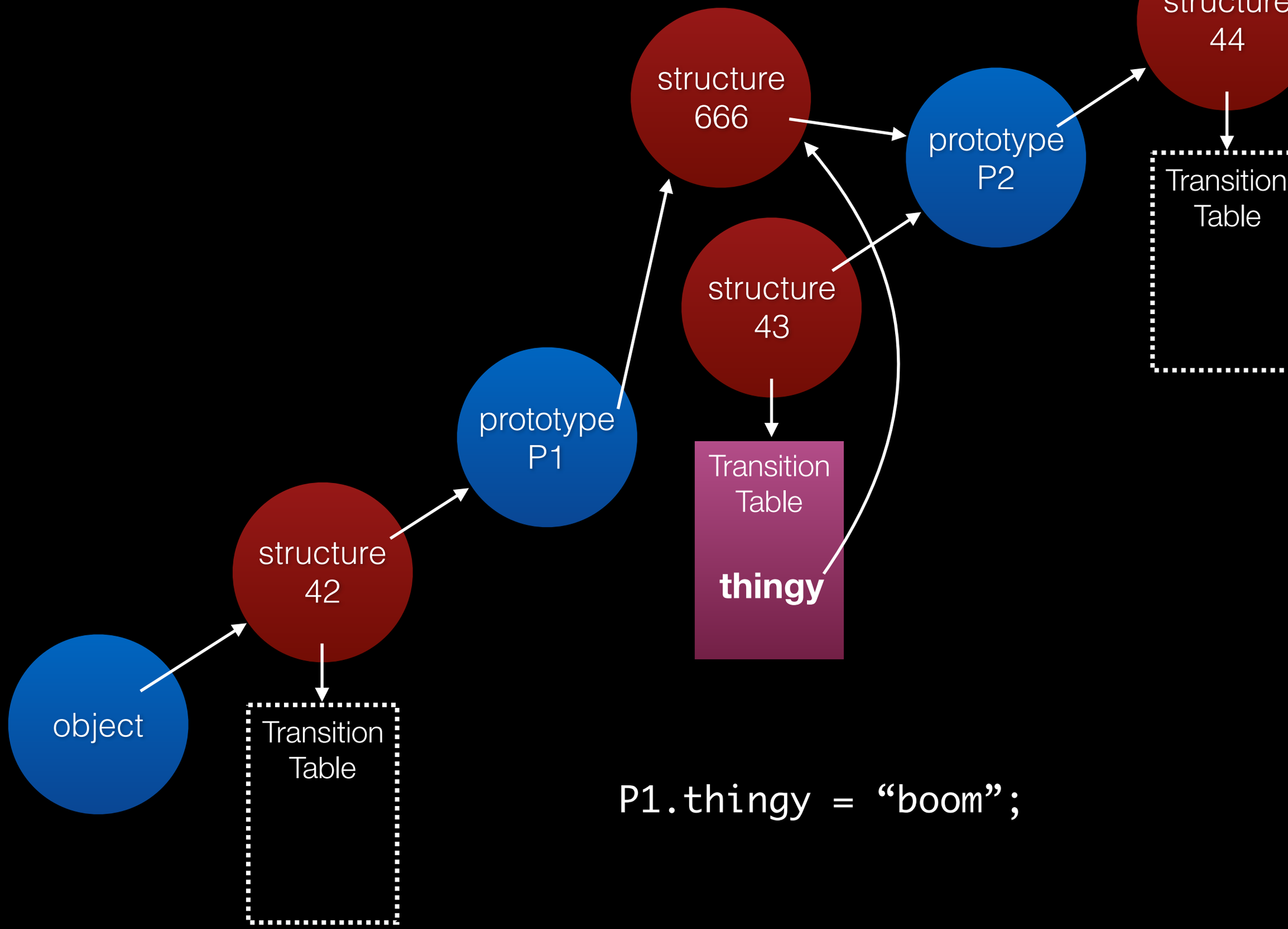

```
P1.thingy = “boom”;
```

Optimized Code

```
if (o->structureID == 42)
    return P2->inlineStorage[…]
else
    slowGet(o, “…”)
```




```
P1.thingy = “boom”;
```



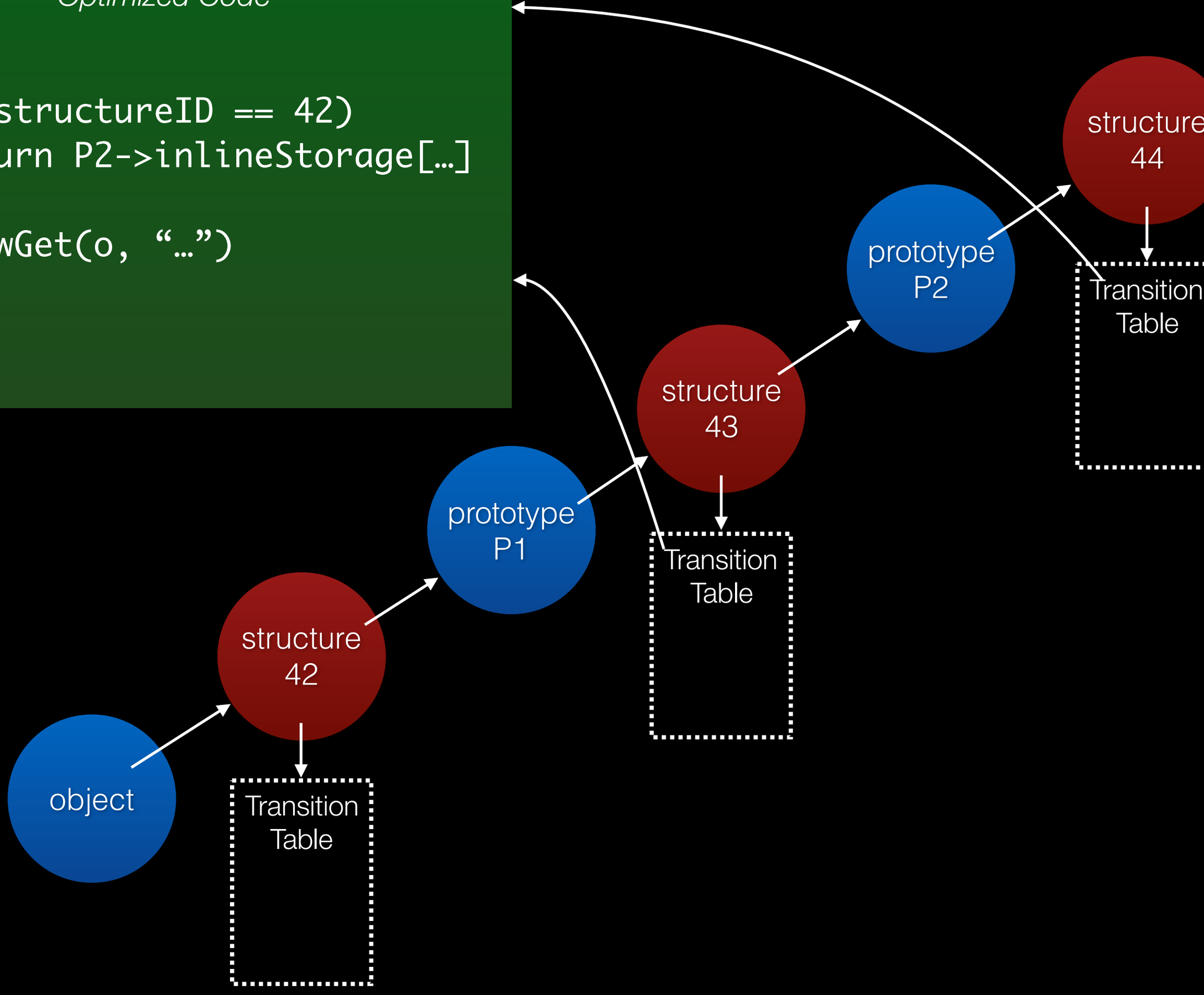

```
P1.thingy = “boom”;
```

Watchpoint

```
class Watchpoint {
public:
    virtual void fire() = 0;
};
```

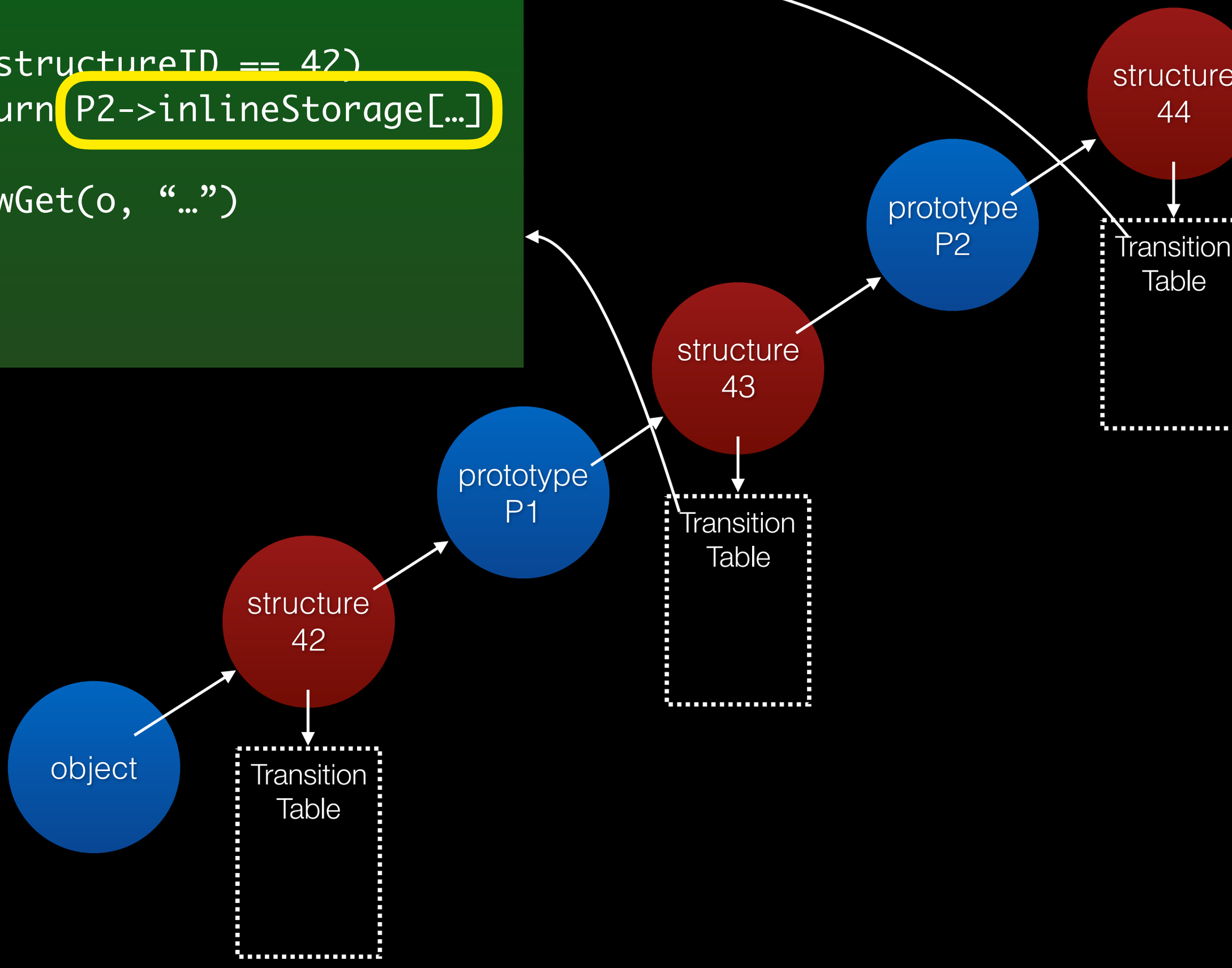
Optimized Code

```
if (o->structureID == 42)
    return P2->inlineStorage[…]
else
    slowGet(o, “…”)
```



Optimized Code

```
if (o->structureID == 42)
    return P2->inlineStorage[…]
else
    slowGet(o, “…”)
```



Optimized Code

```
if (o->structureID == 42)
    return P2->inlineStorage[…]
else
    slowGet(o, “…”)
```

Replacement Map

field

structure 44

prototype P2

Transition Table

structure 43

prototype P1

Transition Table

structure 42

object

Transition Table

Optimized Code

```
if (o->structureID == 42)
    return constant
else
    slowGet(o, "…")
```

Replacement Map

field

structure 44

prototype P2

Transition Table

structure 43

prototype P1

Transition Table

structure 42

object

Transition Table

Simple Inline Caching

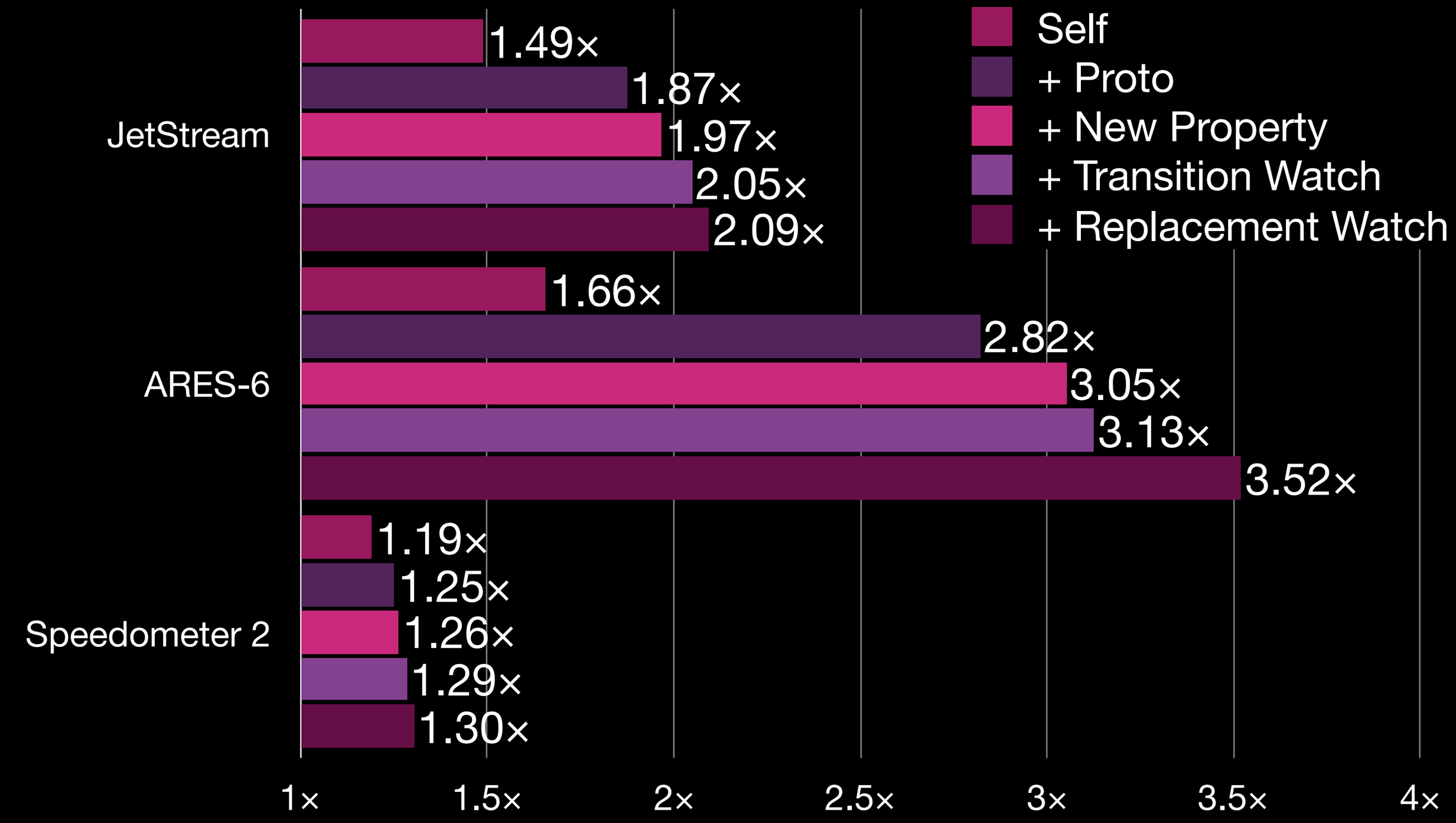
- tmp = o.f // self
- o.f = tmp // self, existing
- o.f = tmp // self, new
- tmp = o.f // prototype

Simple Inline Caching

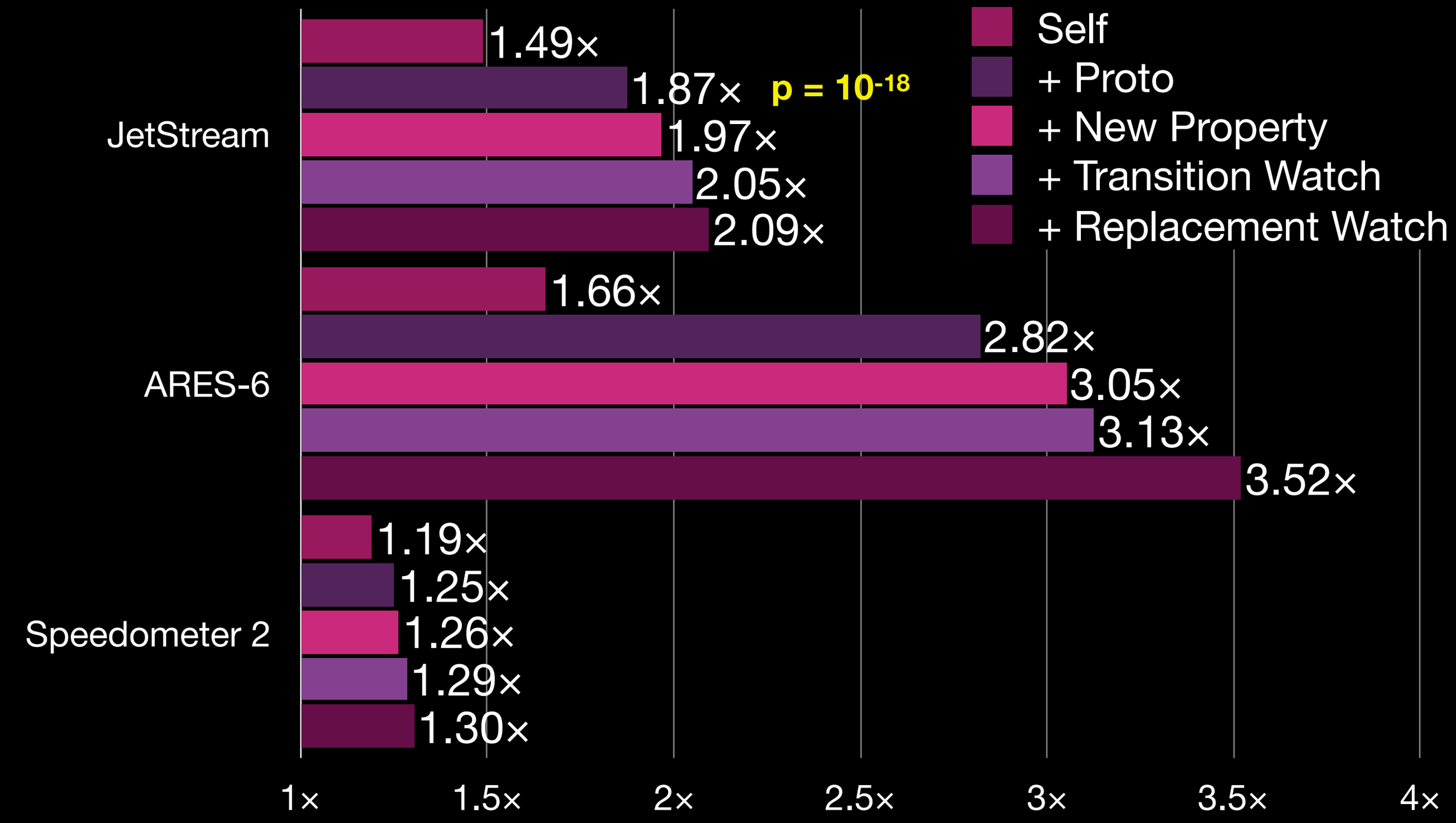
- tmp = o.f // self
- o.f = tmp // self, existing
- o.f = tmp // self, new
- tmp = o.f // prototype

only need one branch

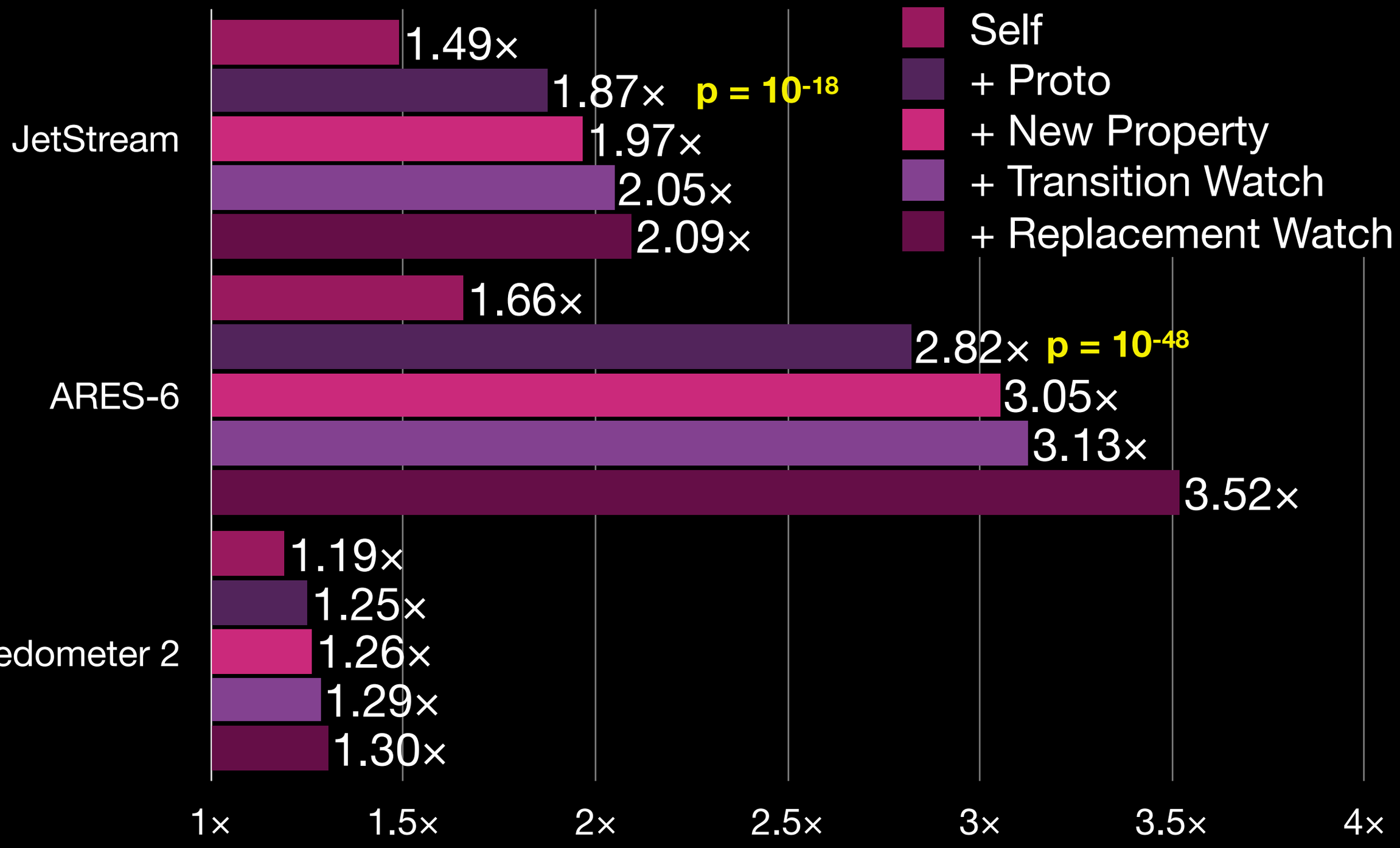
JIT Monomorphic IC speed-up



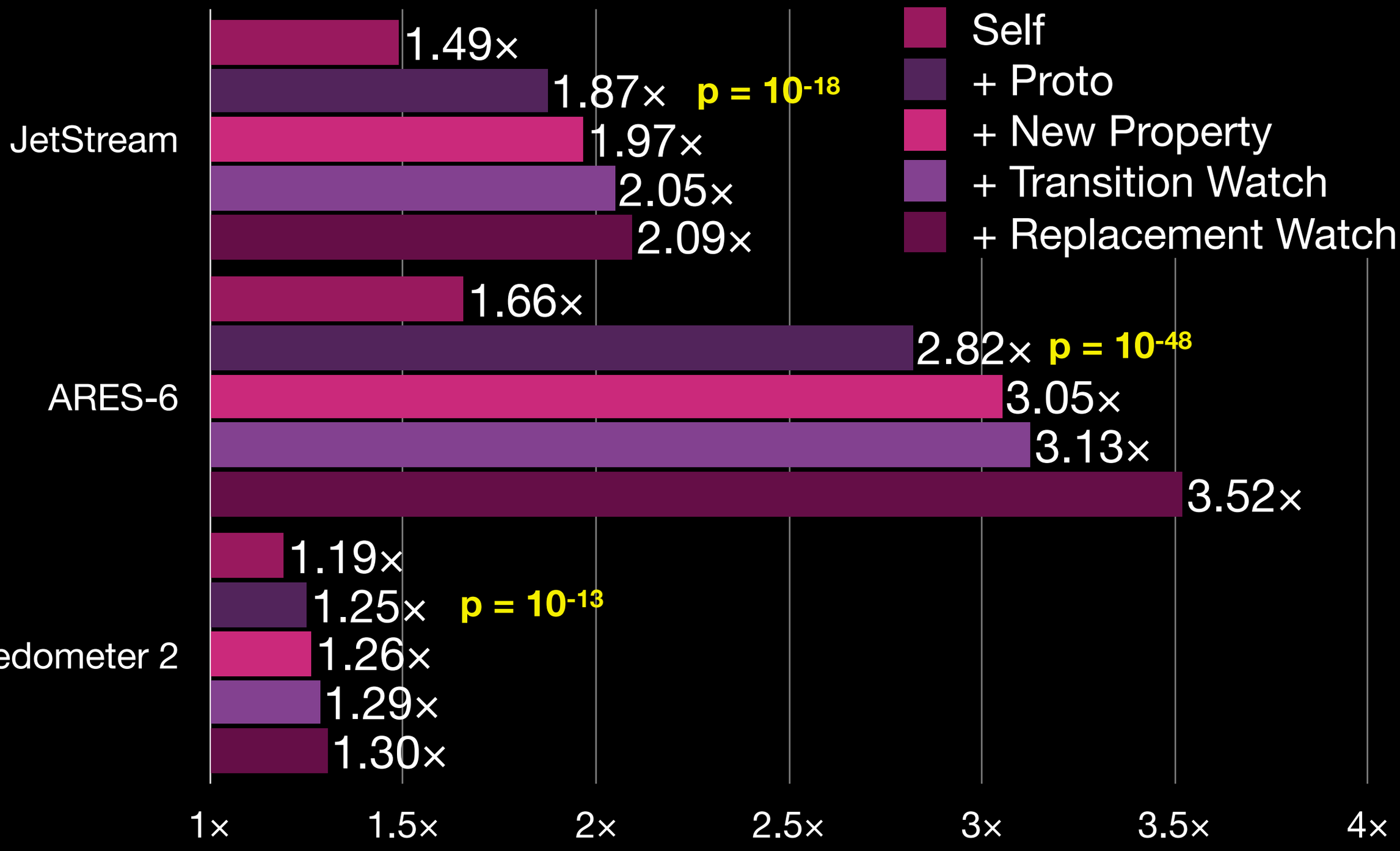
JIT Monomorphic IC speed-up



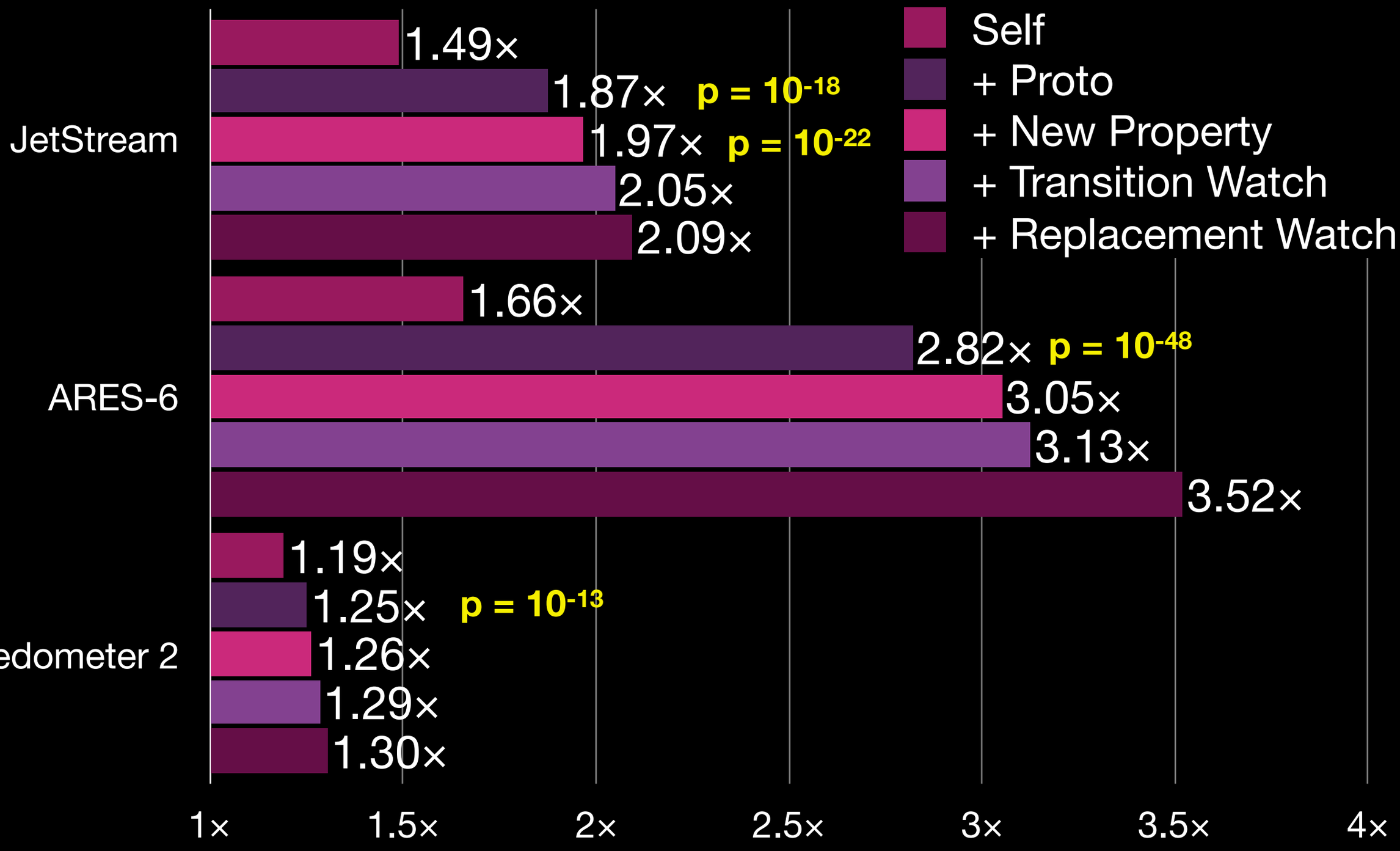
JIT Monomorphic IC speed-up



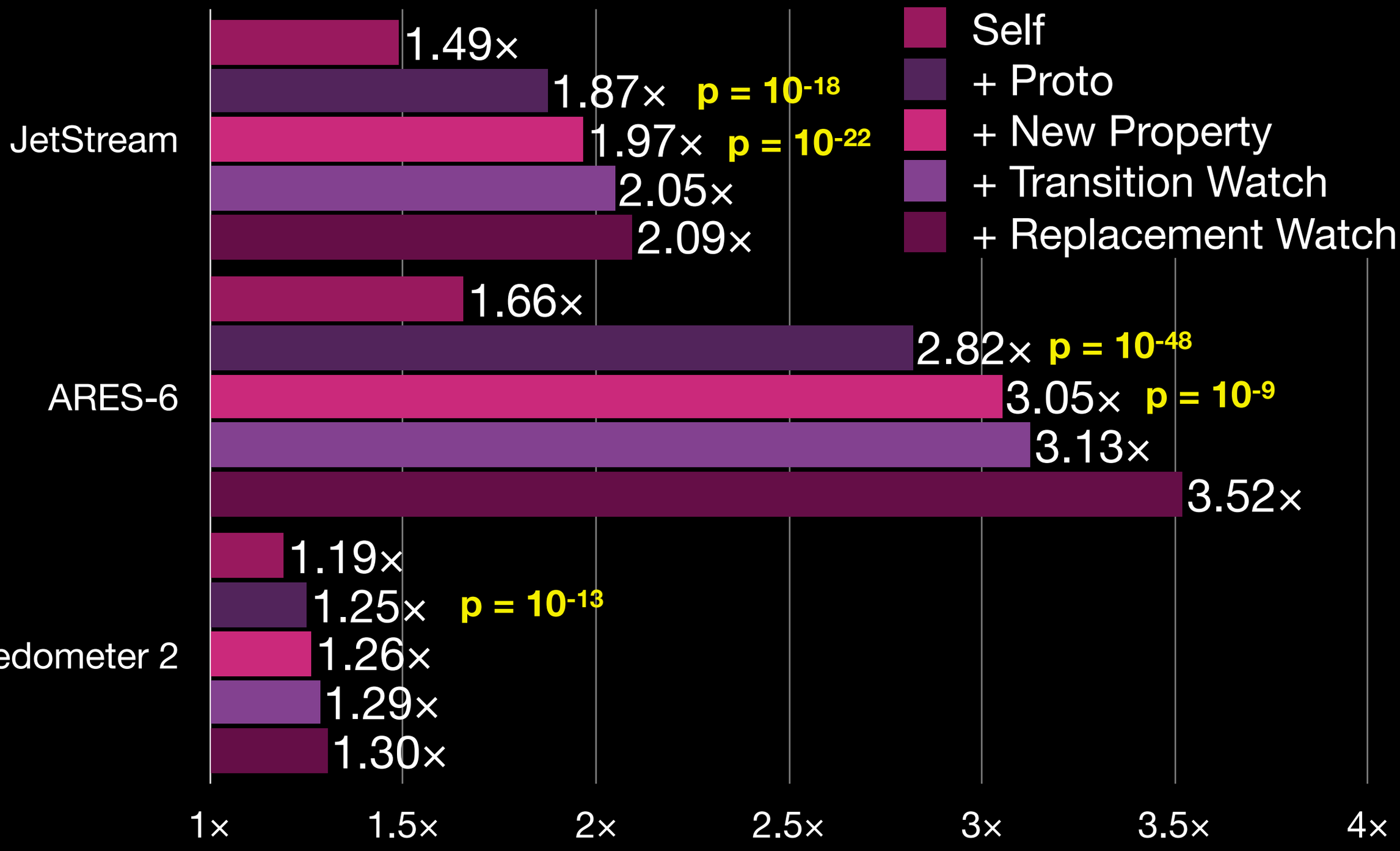
JIT Monomorphic IC speed-up



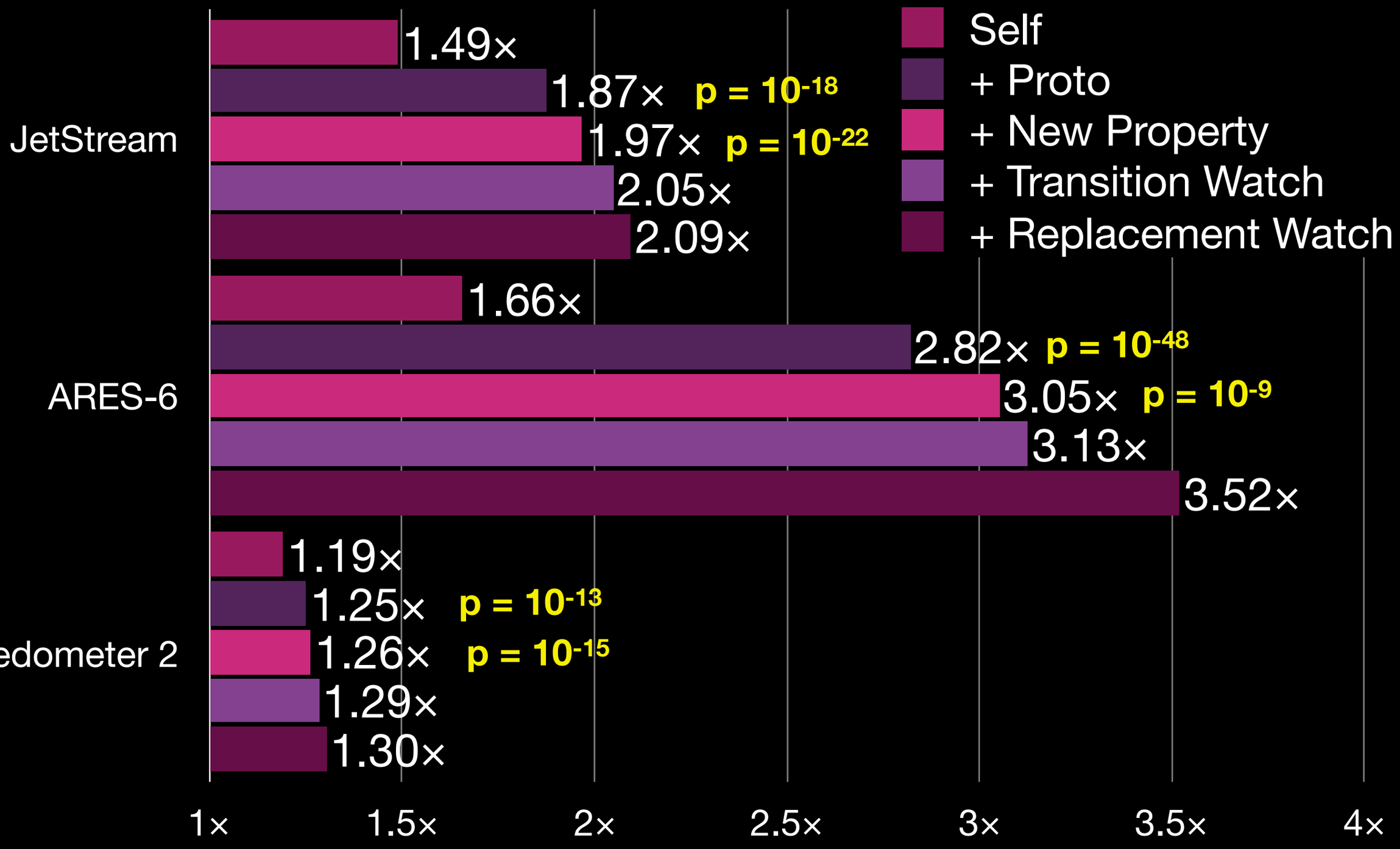
JIT Monomorphic IC speed-up



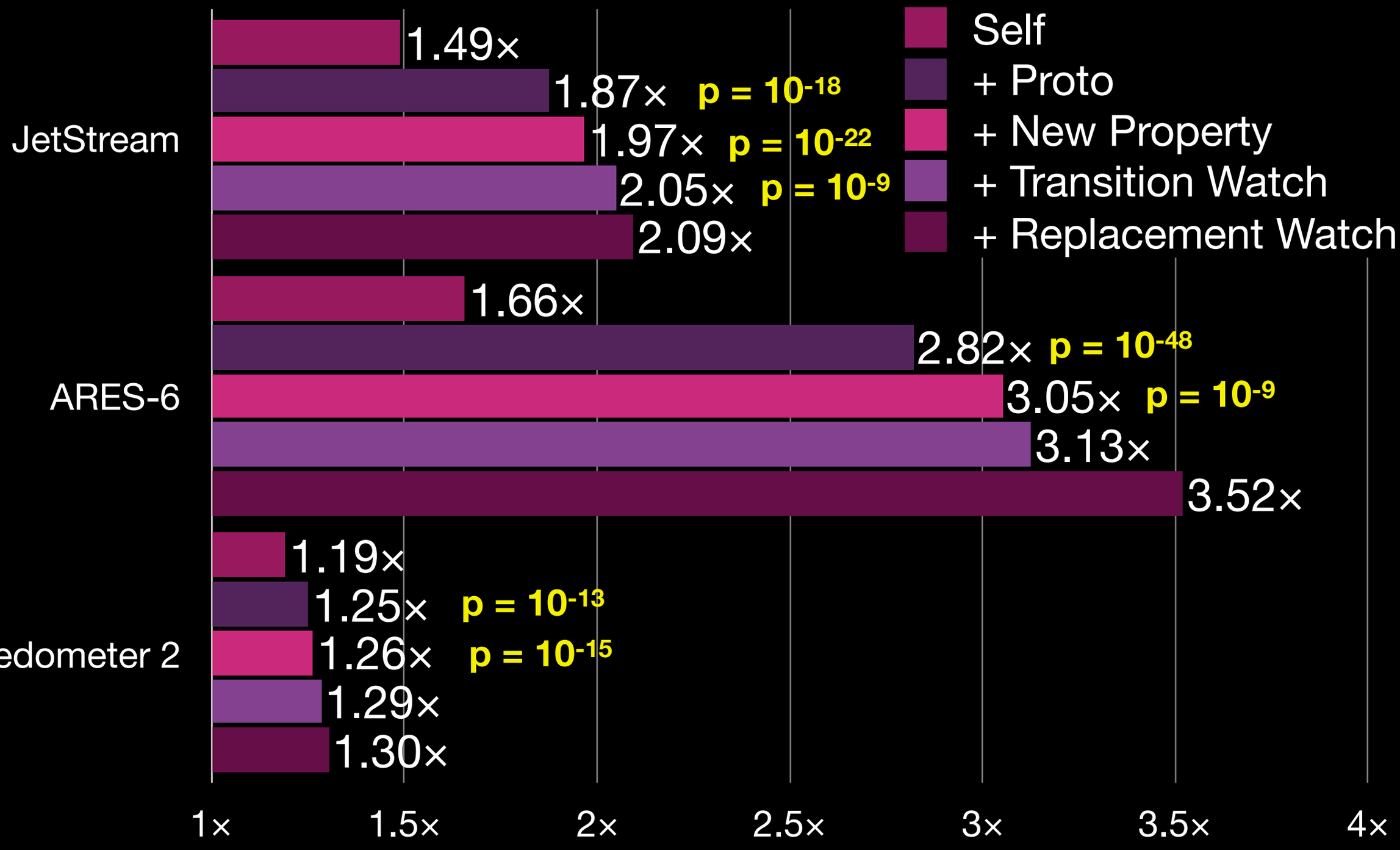
JIT Monomorphic IC speed-up



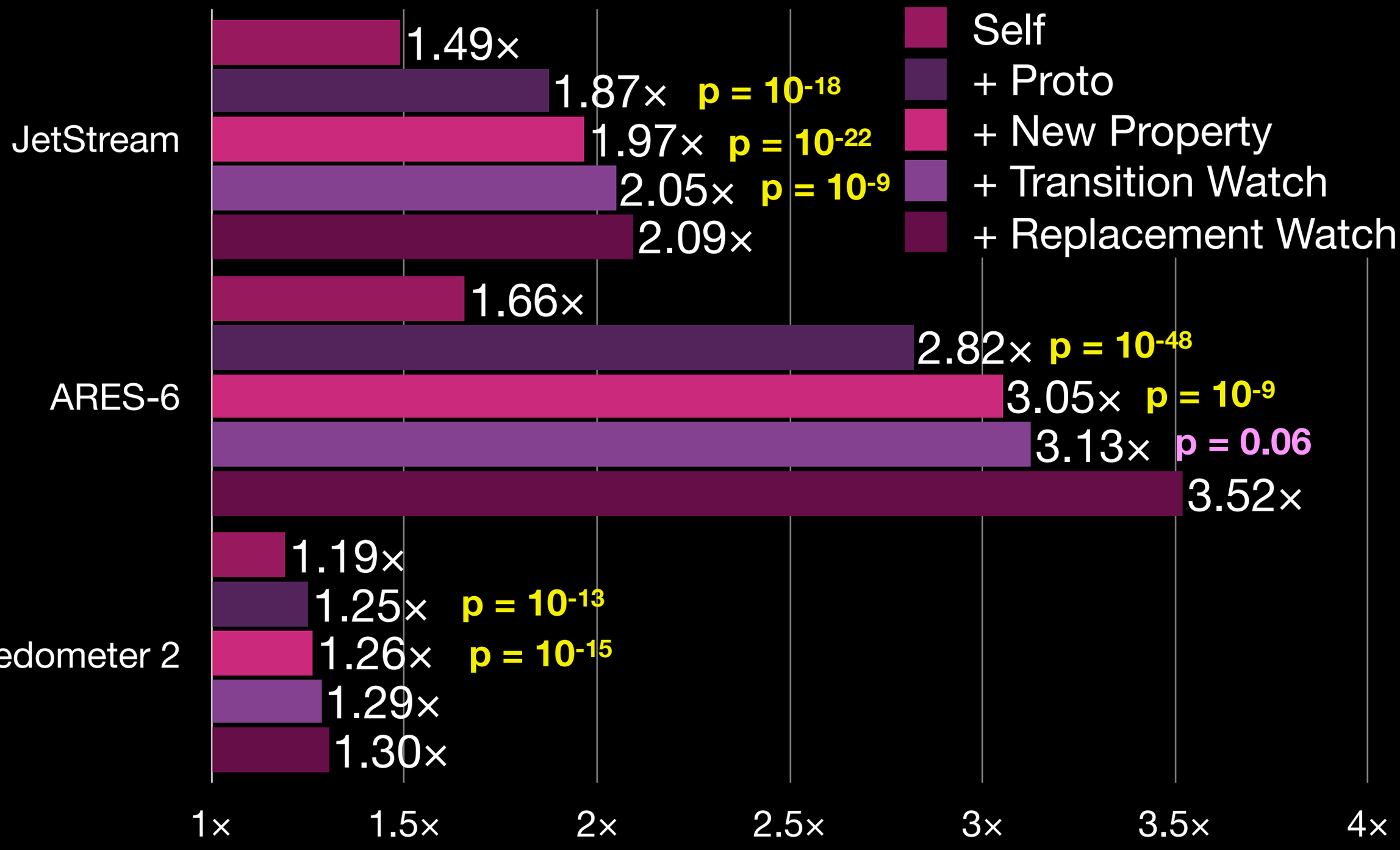
JIT Monomorphic IC speed-up



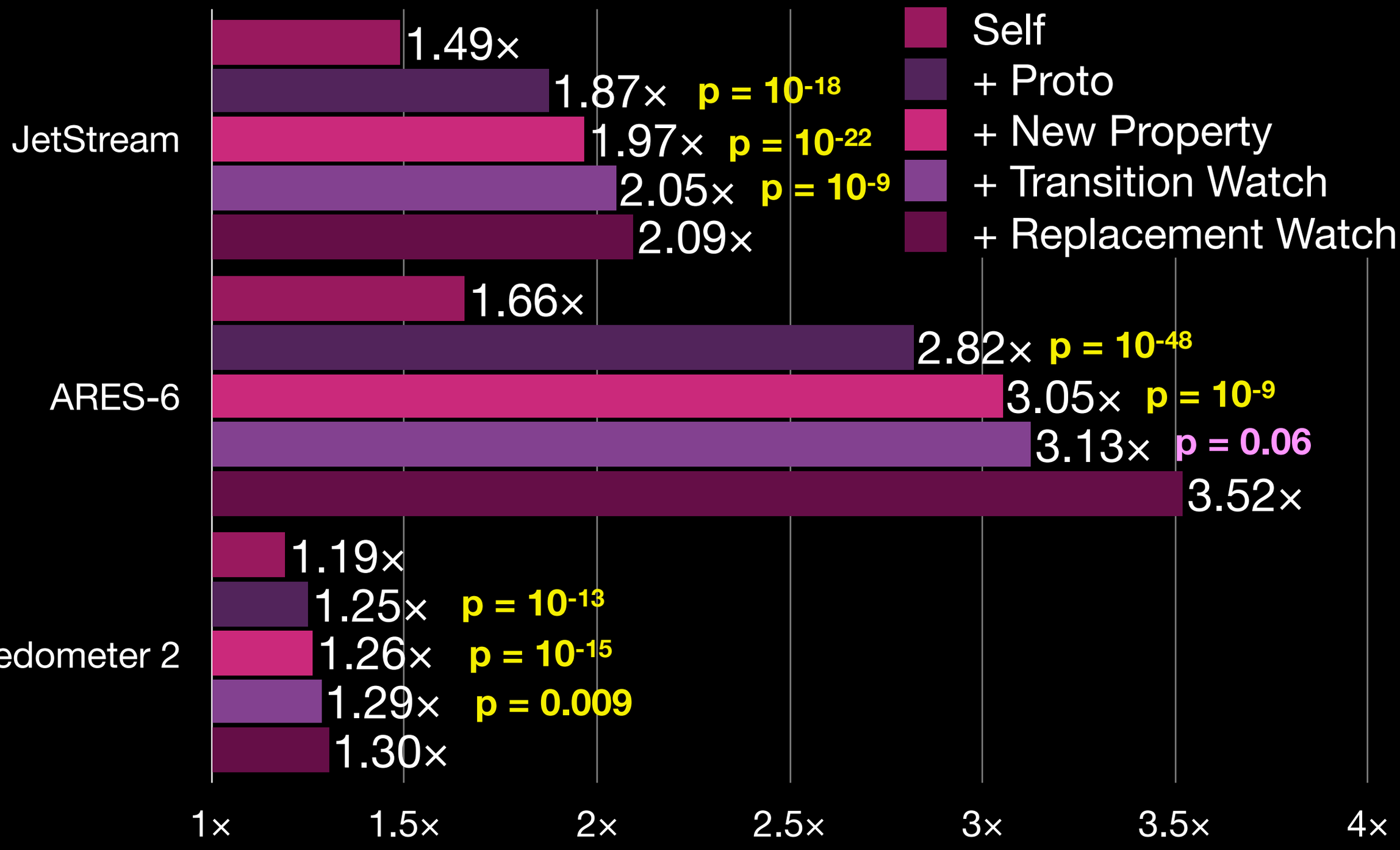
JIT Monomorphic IC speed-up



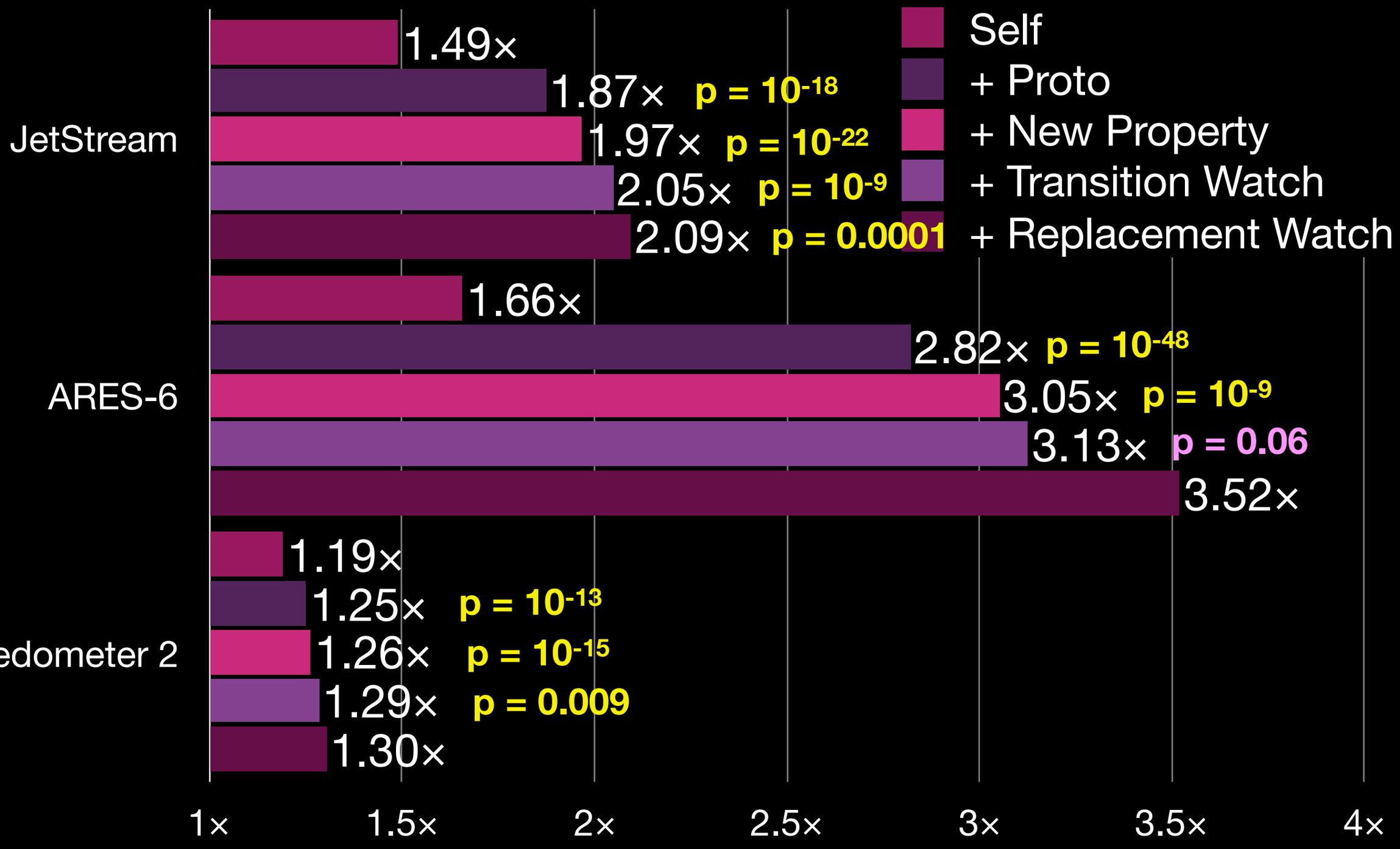
JIT Monomorphic IC speed-up



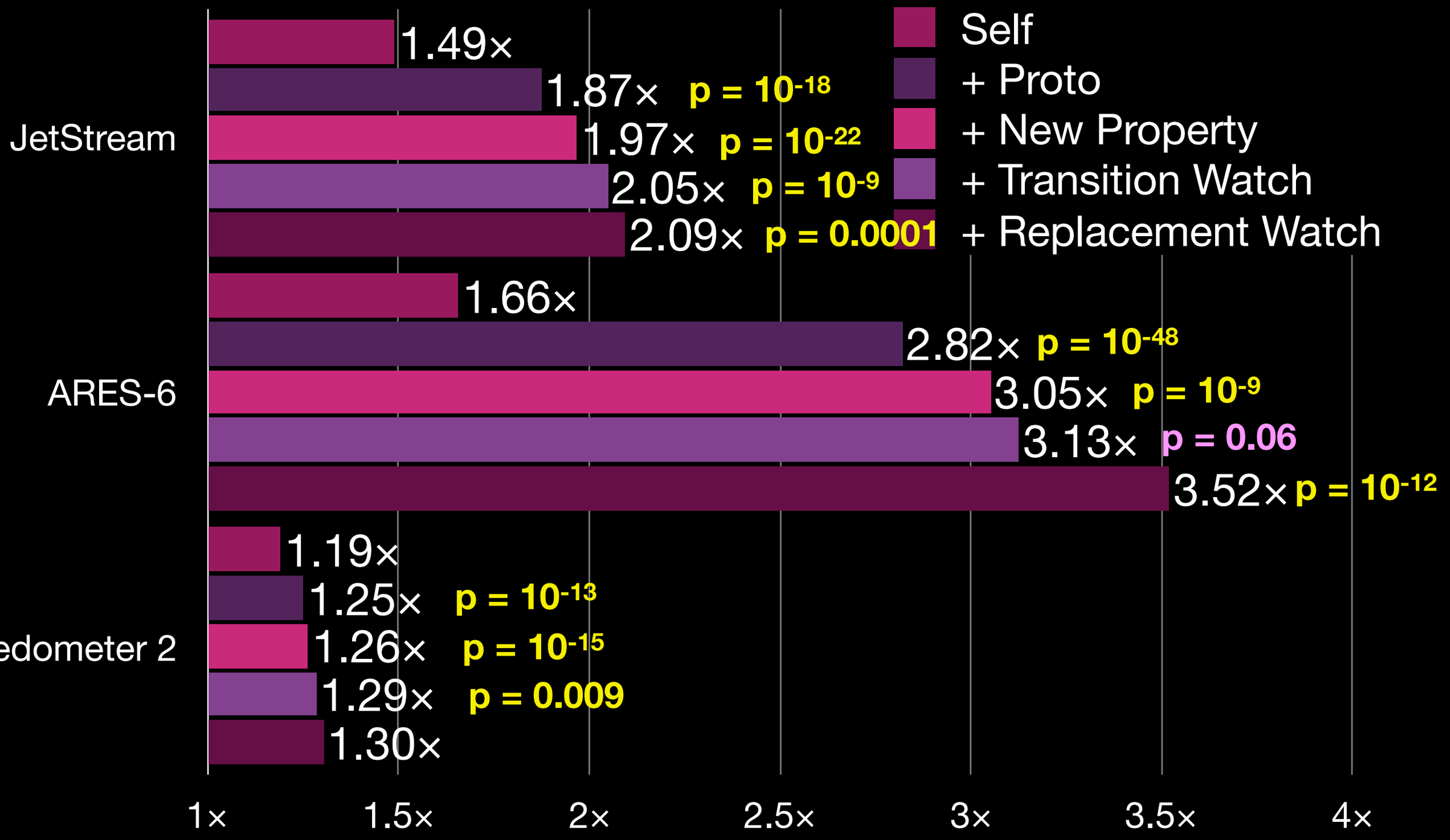
JIT Monomorphic IC speed-up



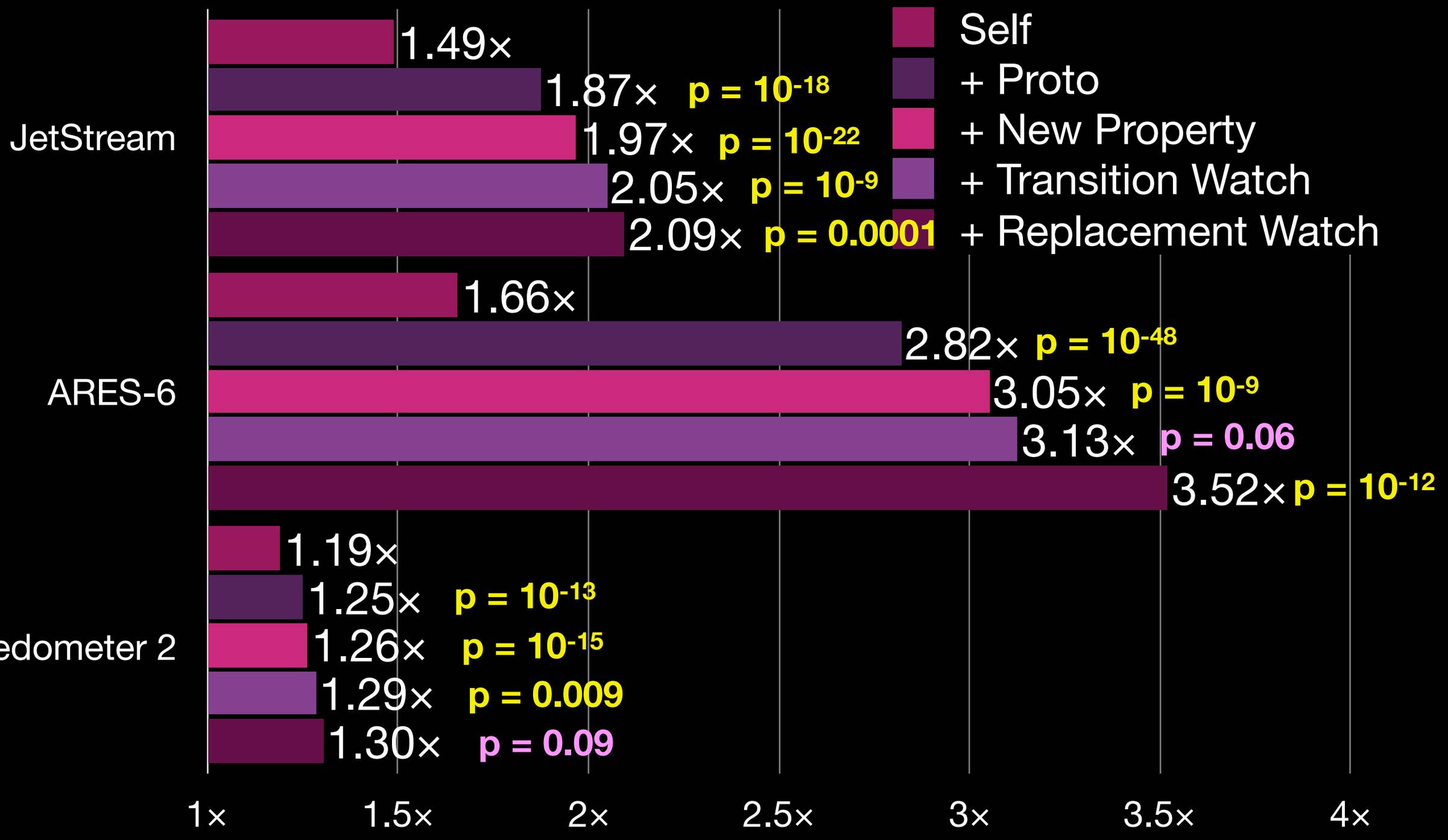
JIT Monomorphic IC speed-up



JIT Monomorphic IC speed-up



JIT Monomorphic IC speed-up



Advanced Topics

- Polymorphic Inline Caches
- Inlining Inline Caches
- Type Inference

Polymorphic Inline Caches

{f: 64, g:75}

{f: 64, g:75}

{f: 98, g:23}

{f: 7, g:32}

{f: 64, g:75}

{f: 98, g:23}

{f: 7, g:32}

{f: 64, g:75}

{f: 98, g:23}

{f: 342, g:5}

{f: 7, g:32}

{f: 64, g:75}

{f: 54, g:75, h:389}

{f: 98, g:23}

{f: 342, g:5}

{f: 6, g:18, h:83}

{f: 7, g:32}

{f: 64, g:75}

{f: 54, g:75, h:389}

{f: 98, g:23}

{f: 342, g:5}

{g: 37, f: 30}

{f: 6, g:18, h:83}

{f: 7, g:32}

{f: 64, g:75}

{f: 54, g:75, h:389}

{f: 98, g:23}

{f: 342, g:5}

{g: 37, f: 30}

{f: 6, g:18, h:83}

{f: 7, g:32}

{f: 64, g:75}

{f: 54, g:75, h:389}

{f: 98, g:23}

{g: 835, f: 0}

{f: 342, g:5}

{f: 6, g:18, h:83}

{g: 37, f: 30}

{f: 7, g:32}

{f, g}

{f, g, h}

{f: 64, g:75}

{f: 54, g:75, h:389}

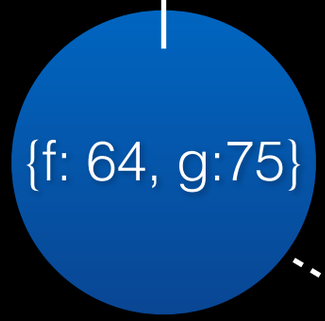
{g, f}

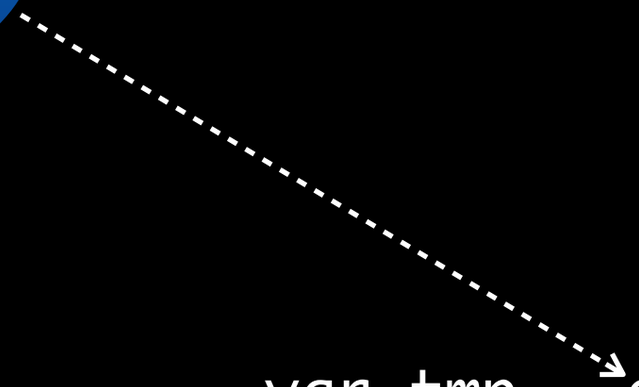
{f: 98, g:23}

{f: 342, g:5}

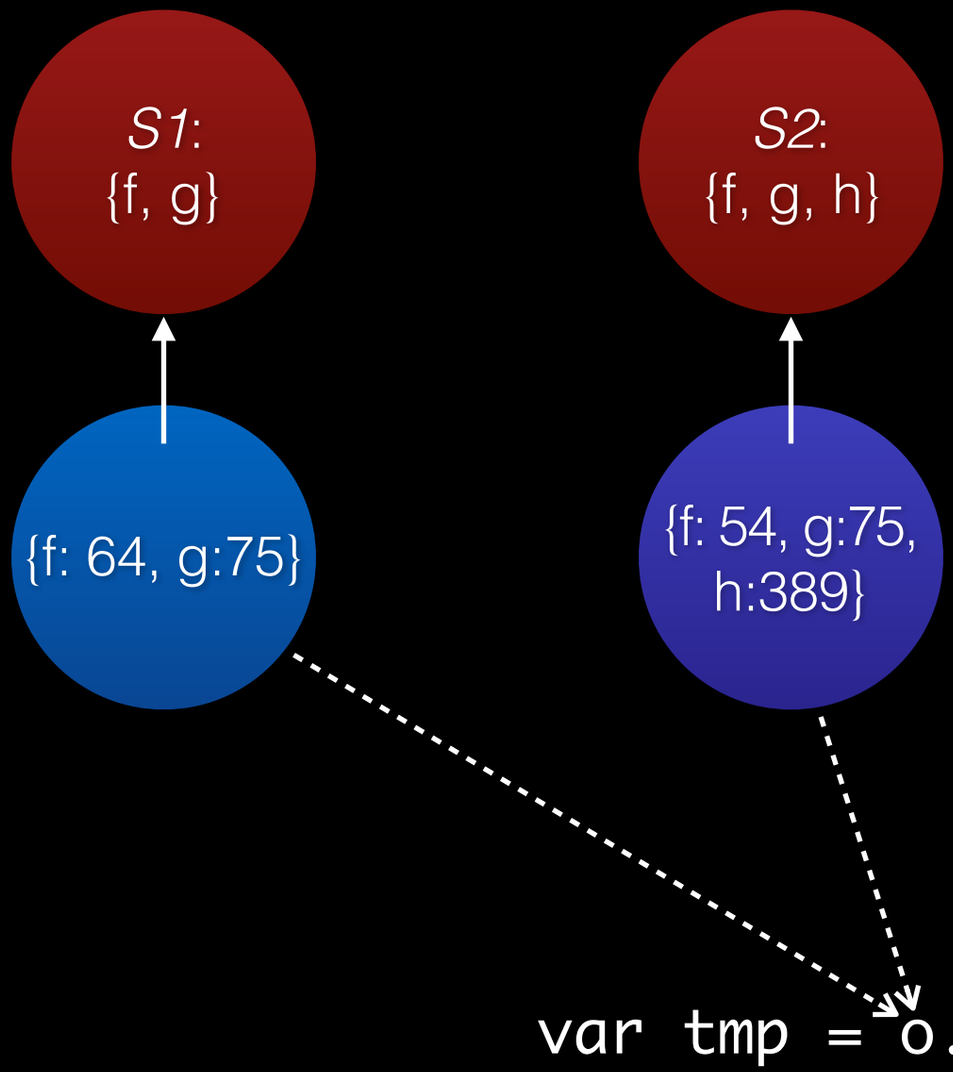
{g: 835, f: 0}

```
var tmp = o.f;
```

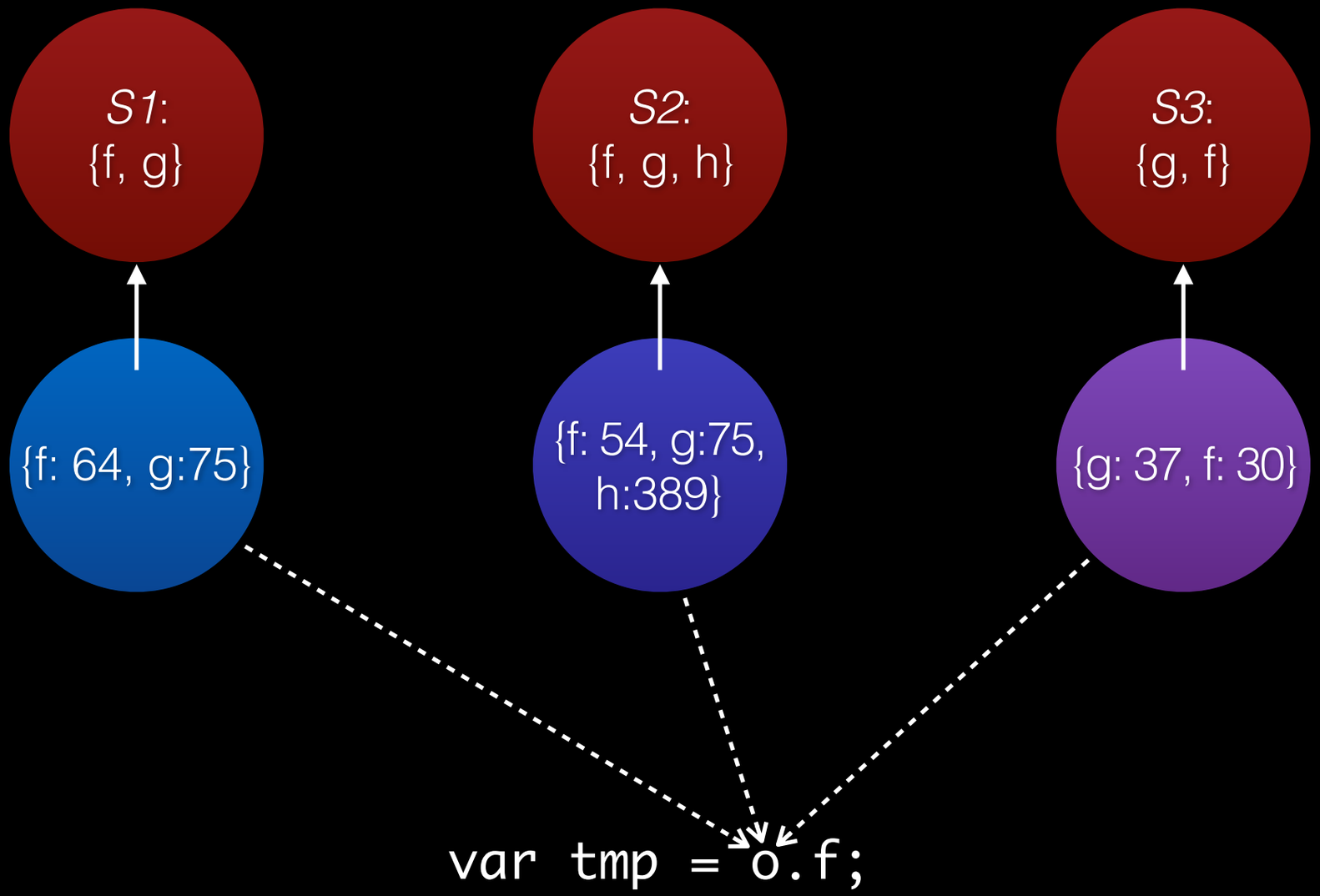


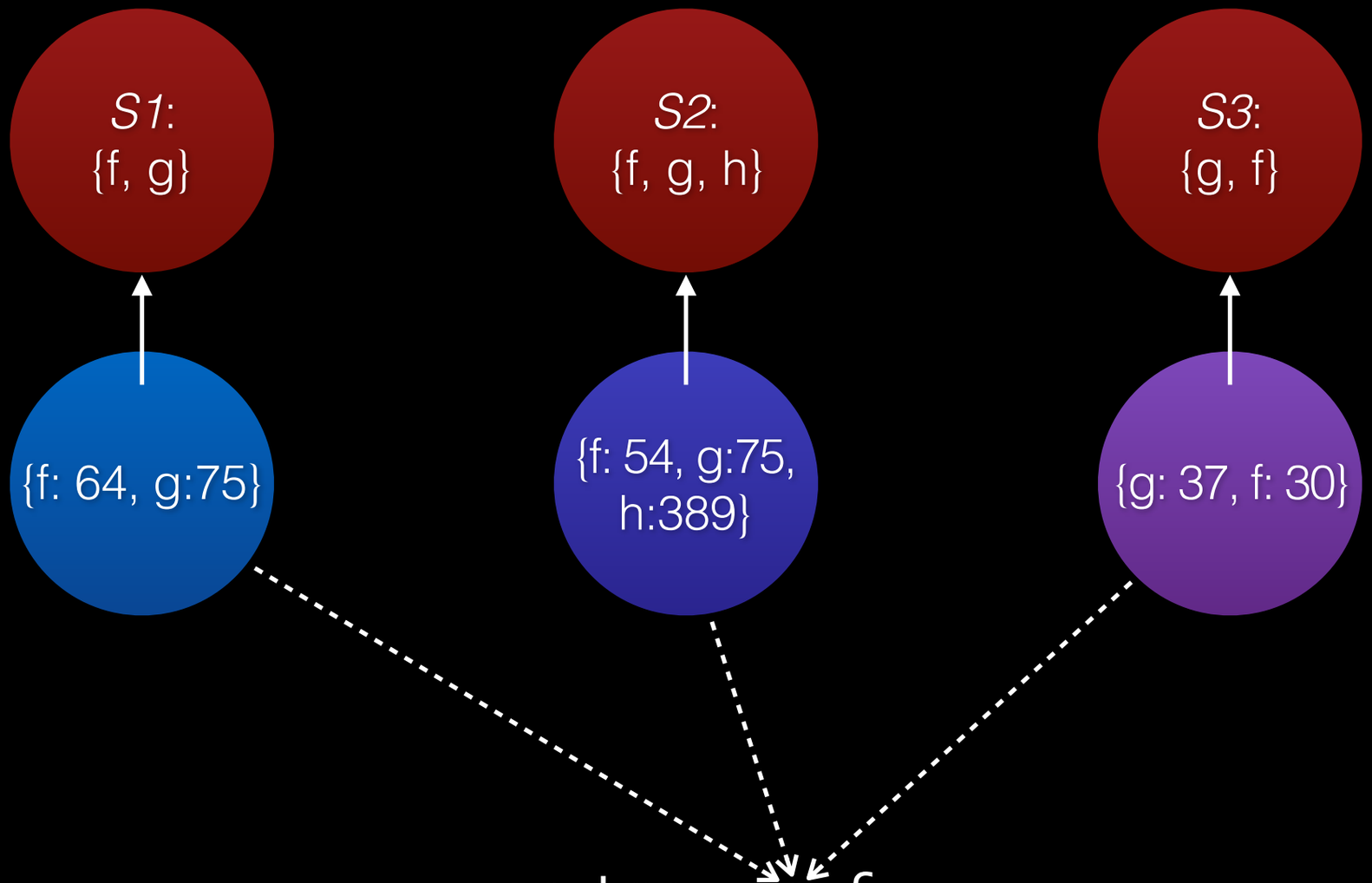


```
var tmp = o.f;
```




```
var tmp = o.f;
```


S1:
{f, g}
S2:
{f, g, h}
S3:
{g, f}
{f: 64, g:75}
{f: 54, g:75,
h:389}
{g: 37, f: 30}
var tmp = o.f;




```
if (o->structureID == S1 || o->structureID == S2)
    o->inlineStorage[0]
else if (o->structureID == S3)
    o->inlineStorage[1]
else
    slowGet(o, “f”)
```

```
function foo(o) {
    return o.f;
}

noInline(foo);

for (var i = 0; i < 10000; ++i) {
    foo({f: 1, g: 2});
    foo({f: 1, g: 2, h:3});
    foo({g: 2, f: 1});
}
```


```
0x456923529be: cmp $0x129, (%rax)
0x456923529c4: jnz 0x45692352a2d
0x456923529ca: mov 0x10(%rax), %rax
0x456923529ce: nop 0x200(%rax)
```

```
function foo(o) {
    return o.f;
}

noInline(foo);

for (var i = 0; i < 10000; ++i) {
    foo({f: 1, g: 2});
    foo({f: 1, g: 2, h:3});
    foo({g: 2, f: 1});
}
```


```
0x45692352be0: mov (%rax), %esi
0x45692352be2: cmp $0x129, %esi
0x45692352be8: jnz 0x45692352bf7
0x45692352bee: mov 0x10(%rax), %rax
0x45692352bf2: jmp 0x456923529d5
0x45692352bf7: cmp $0x126, %esi
0x45692352bfd: jnz 0x45692352c0c
0x45692352c03: mov 0x10(%rax), %rax
0x45692352c07: jmp 0x456923529d5
0x45692352c0c: cmp $0x12b, %esi
0x45692352c12: jnz 0x45692352a2d
0x45692352c18: mov 0x18(%rax), %rax
0x45692352c1c: jmp 0x456923529d5
0x45692352c21: jmp 0x45692352a2d
```

S1:
{f, g}

S2:
{f, g, h}

S3:
{g, f}

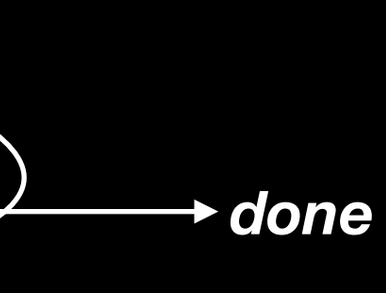
```
0x45692352be0: mov (%rax), %esi
0x45692352be2: cmp $0x129, %esi
0x45692352be8: jnz 0x45692352bf7
0x45692352bee: mov 0x10(%rax), %rax
0x45692352bf2: jmp 0x456923529d5
0x45692352bf7: cmp $0x126, %esi
0x45692352bfd: jnz 0x45692352c0c
0x45692352c03: mov 0x10(%rax), %rax
0x45692352c07: jmp 0x456923529d5
0x45692352c0c: cmp $0x12b, %esi
0x45692352c12: jnz 0x45692352a2d
0x45692352c18: mov 0x18(%rax), %rax
0x45692352c1c: jmp 0x456923529d5
0x45692352c21: jmp 0x45692352a2d
```

S1:
{f, g}

S2:
{f, g, h}

S3:
{g, f}

```
0x45692352be0: mov (%rax), %esi
0x45692352be2: cmp $0x129, %esi
0x45692352be8: jnz 0x45692352bf7
0x45692352bee: mov 0x10(%rax), %rax
0x45692352bf2: jmp 0x456923529d5
0x45692352bf7: cmp $0x126, %esi
0x45692352bfd: jnz 0x45692352c0c
0x45692352c03: mov 0x10(%rax), %rax
0x45692352c07: jmp 0x456923529d5
0x45692352c0c: cmp $0x12b, %esi
0x45692352c12: jnz 0x45692352a2d
0x45692352c18: mov 0x18(%rax), %rax
0x45692352c1c: jmp 0x456923529d5
0x45692352c21: jmp 0x45692352a2d
```

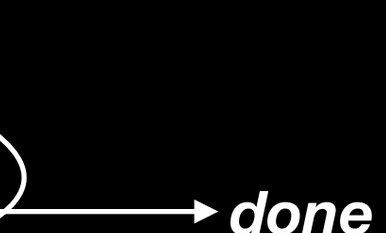


S1:
{f, g}

S2:
{f, g, h}

S3:
{g, f}

```
0x45692352be0: mov (%rax), %esi
0x45692352be2: cmp $0x129, %esi
0x45692352be8: jnz 0x45692352bf7
0x45692352bee: mov 0x10(%rax), %rax
0x45692352bf2: jmp 0x456923529d5
0x45692352bf7: cmp $0x126, %esi
0x45692352bfd: jnz 0x45692352c0c
0x45692352c03: mov 0x10(%rax), %rax
0x45692352c07: jmp 0x456923529d5
0x45692352c0c: cmp $0x12b, %esi
0x45692352c12: jnz 0x45692352a2d
0x45692352c18: mov 0x18(%rax), %rax
0x45692352c1c: jmp 0x456923529d5
0x45692352c21: jmp 0x45692352a2d
```

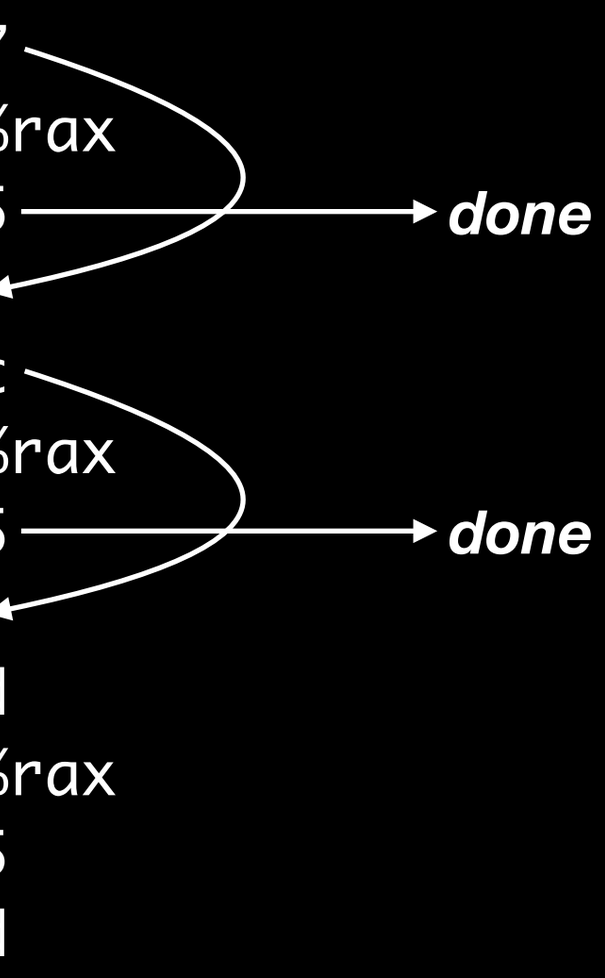


S1:
{f, g}

S2:
{f, g, h}

S3:
{g, f}

```
0x45692352be0: mov (%rax), %esi
0x45692352be2: cmp $0x129, %esi
0x45692352be8: jnz 0x45692352bf7
0x45692352bee: mov 0x10(%rax), %rax
0x45692352bf2: jmp 0x456923529d5
0x45692352bf7: cmp $0x126, %esi
0x45692352bfd: jnz 0x45692352c0c
0x45692352c03: mov 0x10(%rax), %rax
0x45692352c07: jmp 0x456923529d5
0x45692352c0c: cmp $0x12b, %esi
0x45692352c12: jnz 0x45692352a2d
0x45692352c18: mov 0x18(%rax), %rax
0x45692352c1c: jmp 0x456923529d5
0x45692352c21: jmp 0x45692352a2d
```

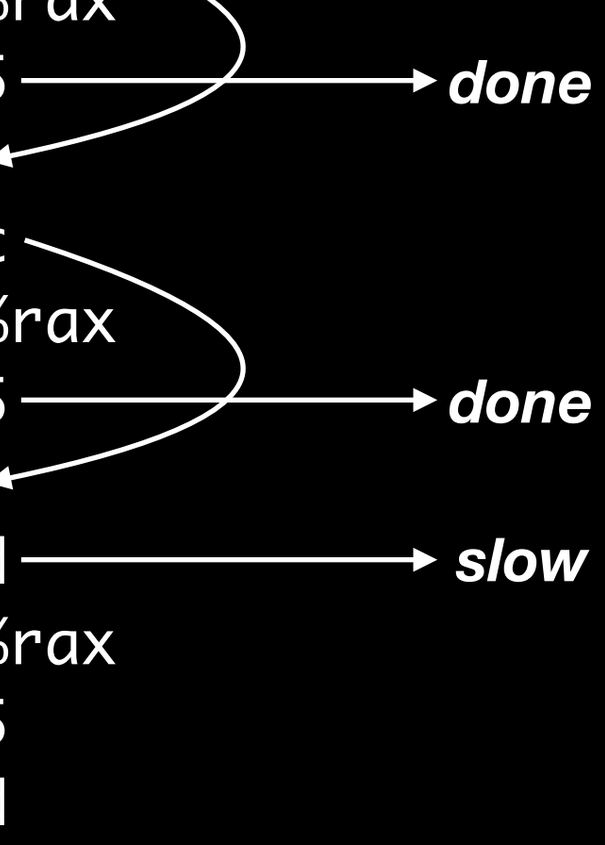


S1:
{f, g}

S2:
{f, g, h}

S3:
{g, f}

```
0x45692352be0: mov (%rax), %esi
0x45692352be2: cmp $0x129, %esi
0x45692352be8: jnz 0x45692352bf7
0x45692352bee: mov 0x10(%rax), %rax
0x45692352bf2: jmp 0x456923529d5
0x45692352bf7: cmp $0x126, %esi
0x45692352bfd: jnz 0x45692352c0c
0x45692352c03: mov 0x10(%rax), %rax
0x45692352c07: jmp 0x456923529d5
0x45692352c0c: cmp $0x12b, %esi
0x45692352c12: jnz 0x45692352a2d
0x45692352c18: mov 0x18(%rax), %rax
0x45692352c1c: jmp 0x456923529d5
0x45692352c21: jmp 0x45692352a2d
```



S1:
{f, g}

S2:
{f, g, h}

S3:
{g, f}

```
0x45692352be0: mov (%rax), %esi
0x45692352be2: cmp $0x129, %esi
0x45692352be8: jnz 0x45692352bf7
0x45692352bee: mov 0x10(%rax), %rax
0x45692352bf2: jmp 0x456923529d5
0x45692352bf7: cmp $0x126, %esi
0x45692352bfd: jnz 0x45692352c0c
0x45692352c03: mov 0x10(%rax), %rax
0x45692352c07: jmp 0x456923529d5
0x45692352c0c: cmp $0x12b, %esi
0x45692352c12: jnz 0x45692352a2d
0x45692352c18: mov 0x18(%rax), %rax
0x45692352c1c: jmp 0x456923529d5
0x45692352c21: jmp 0x45692352a2d
```

done

done

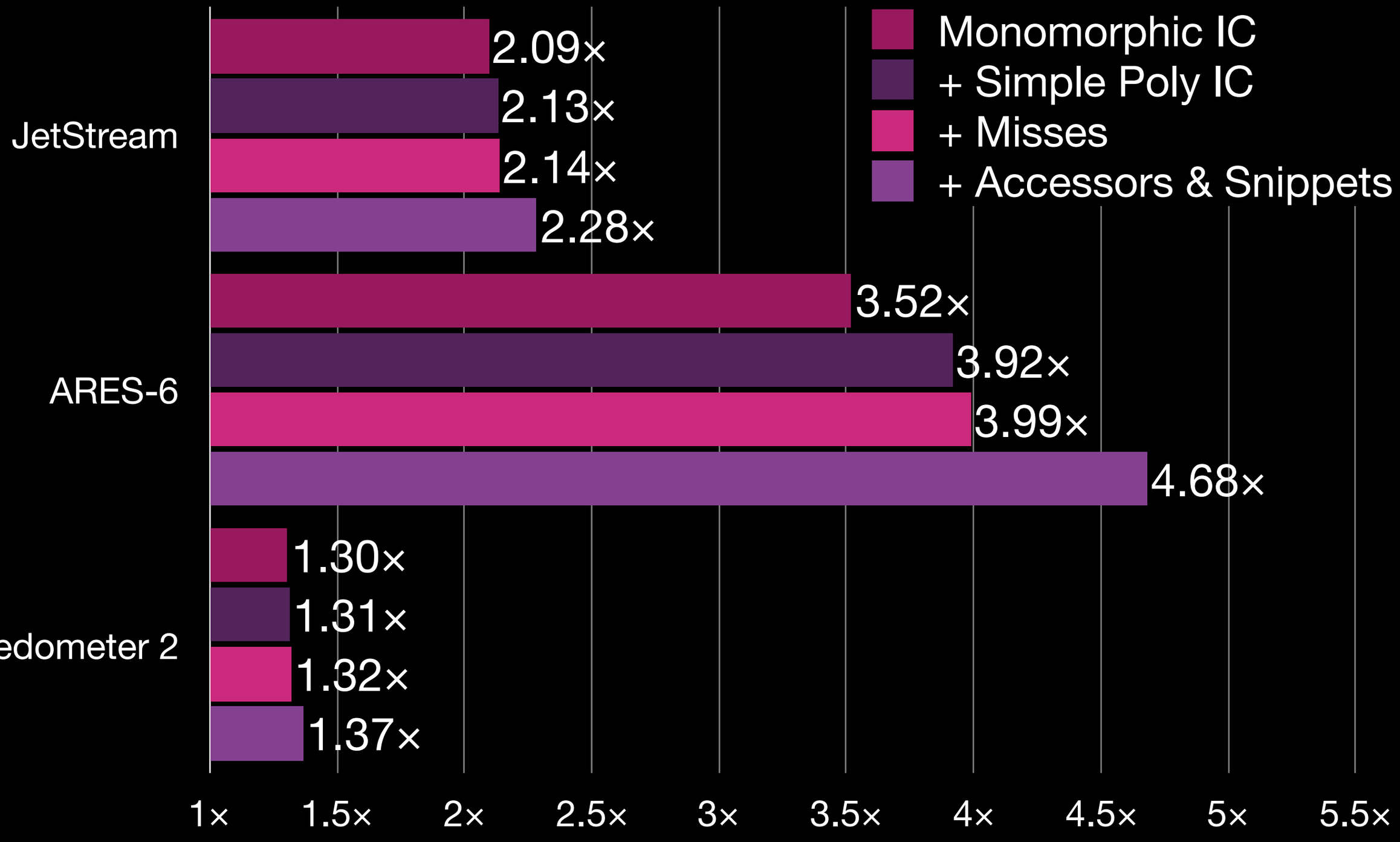
slow

done

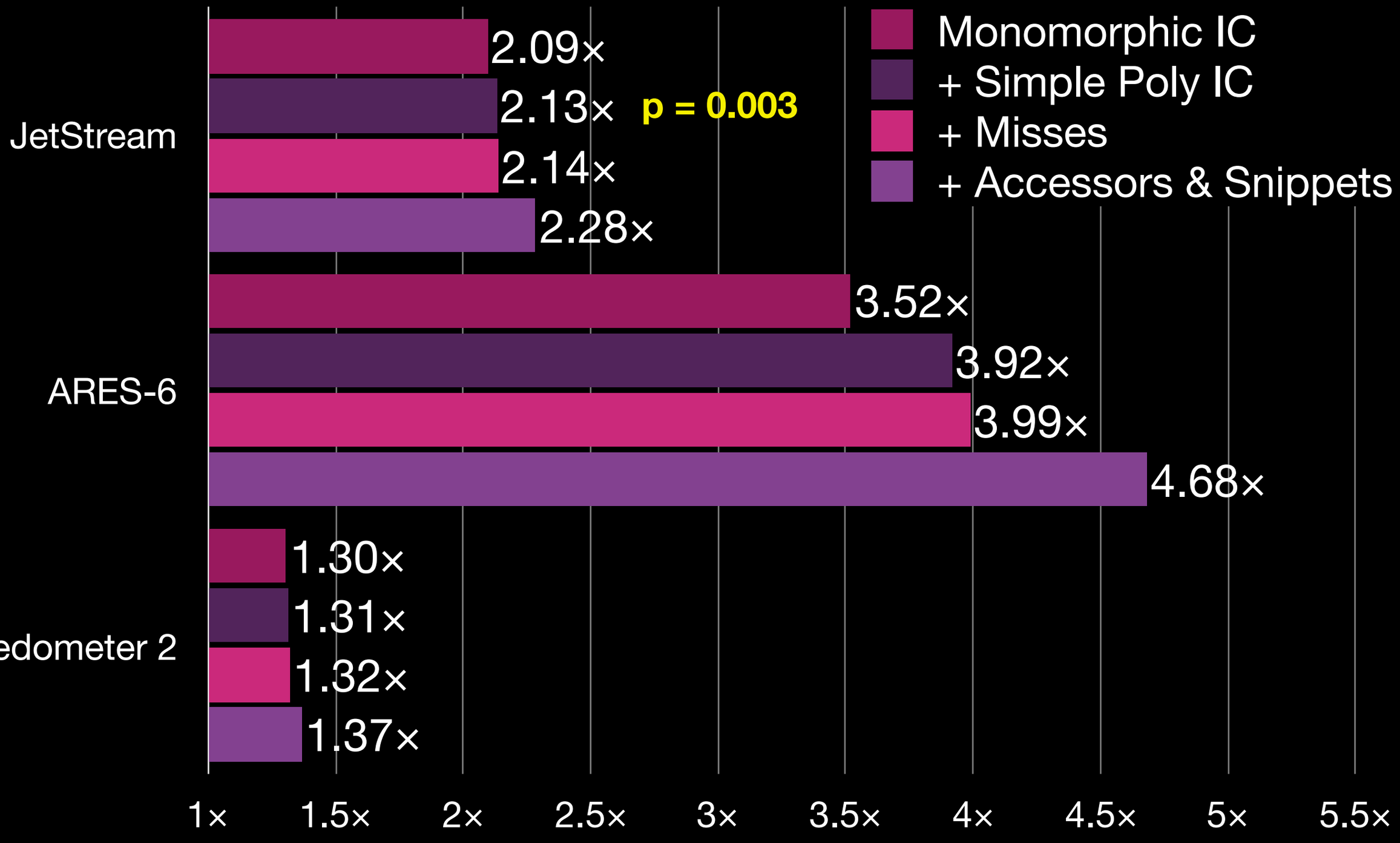
Polymorphic Access JIT

- Self
- Prototype
- Transition (new property)
- Getters
- Setters
- Custom accessors
- Snippets (DOM JIT)

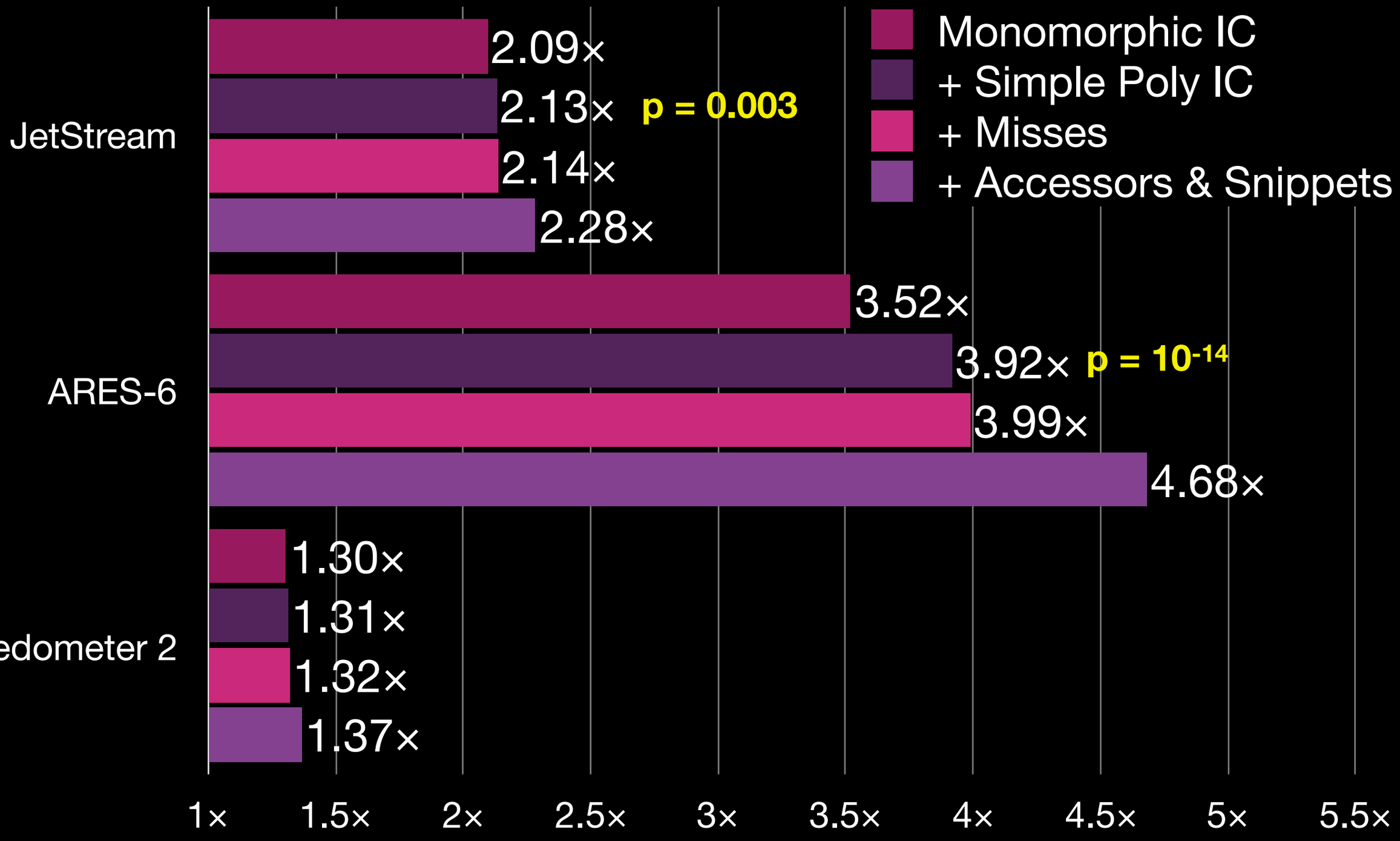
JIT Polymorphic IC speed-up



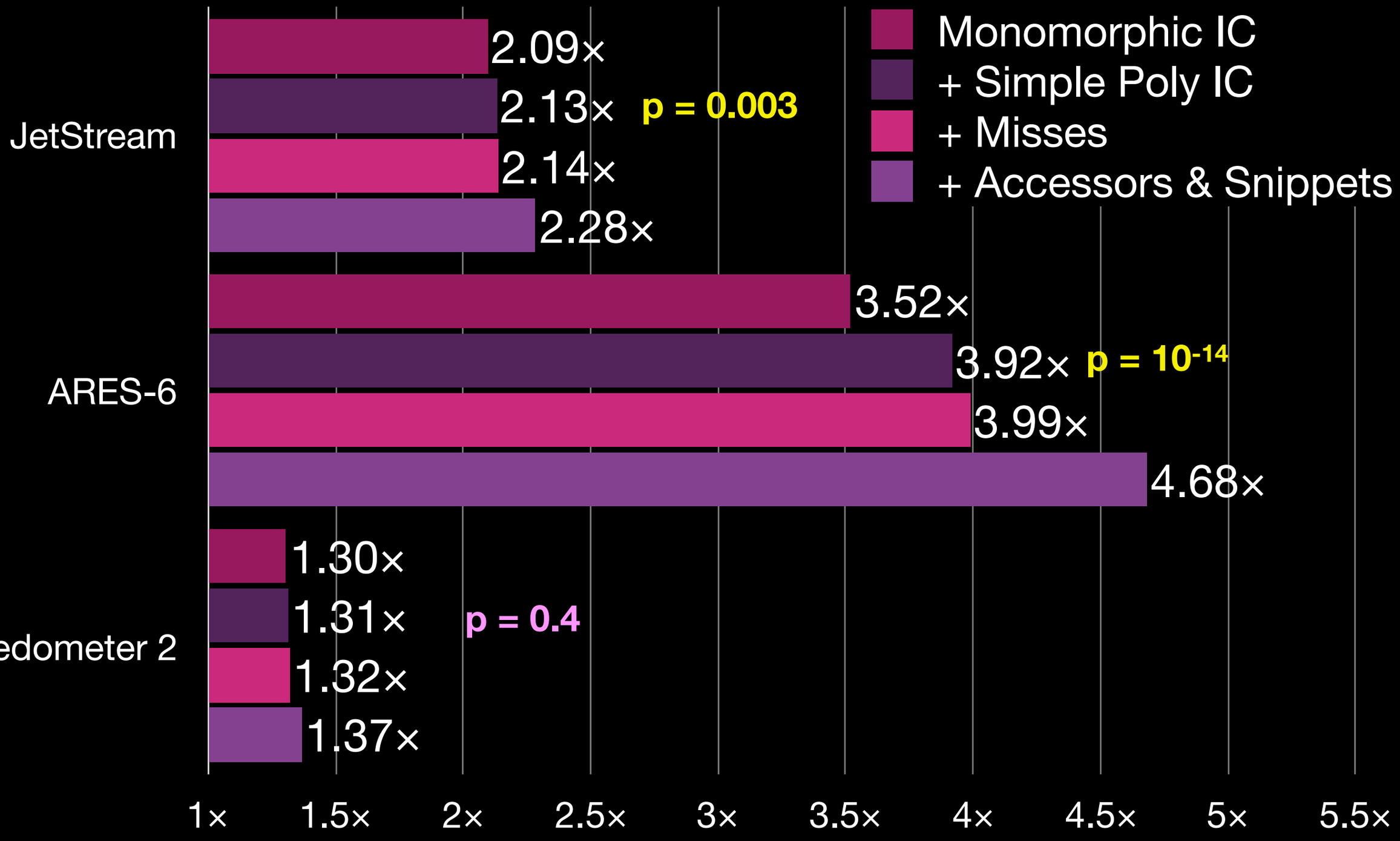
JIT Polymorphic IC speed-up



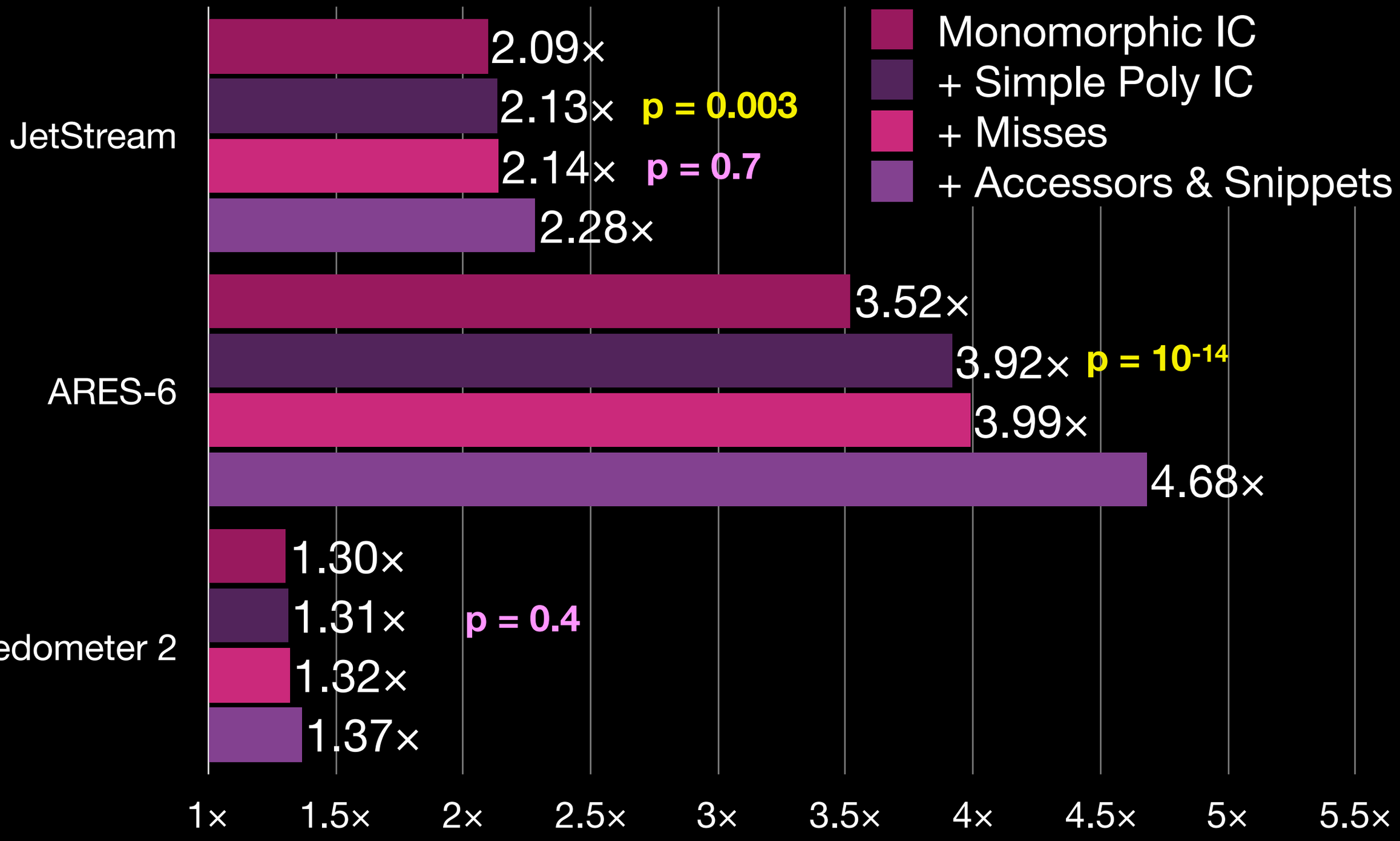
JIT Polymorphic IC speed-up



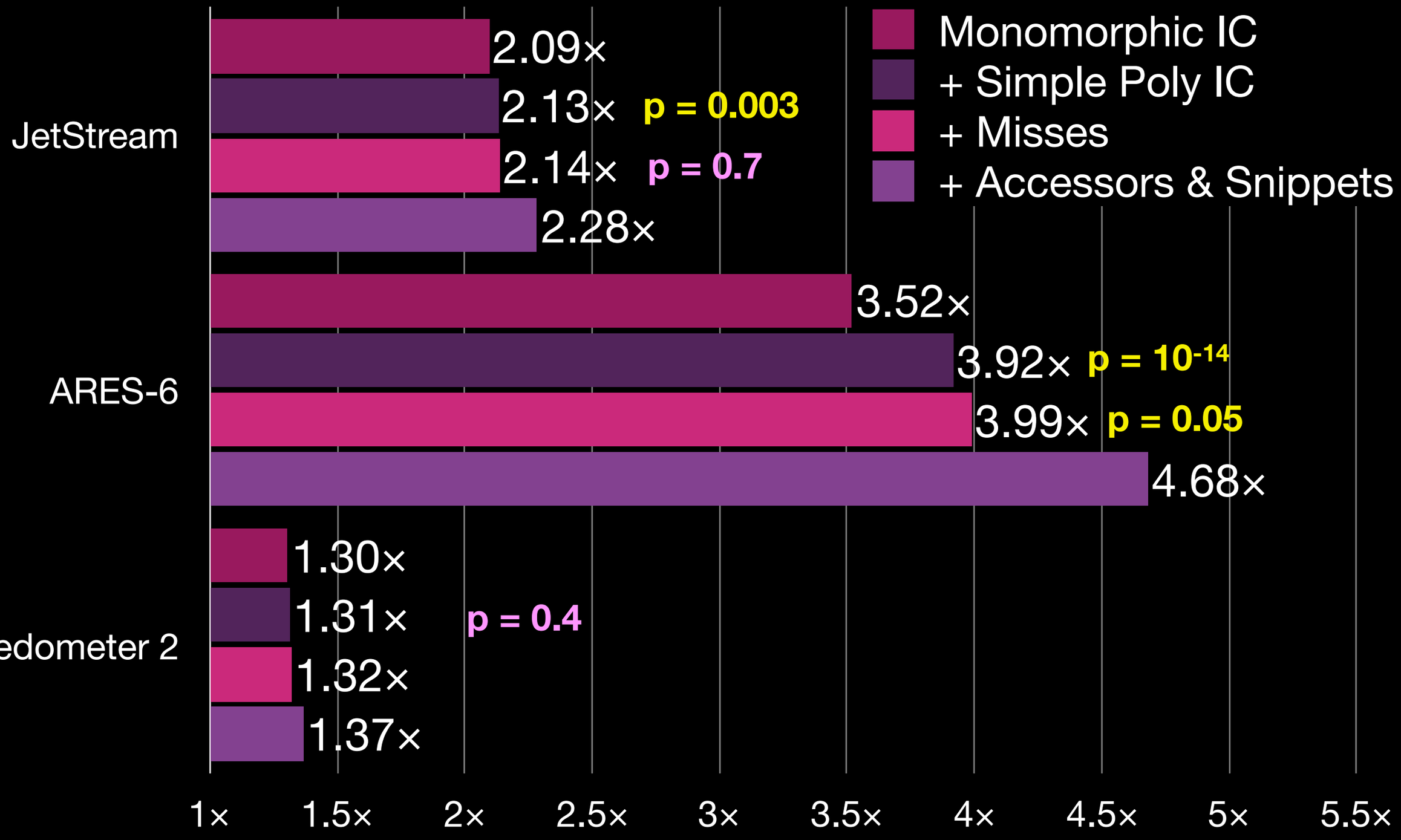
JIT Polymorphic IC speed-up



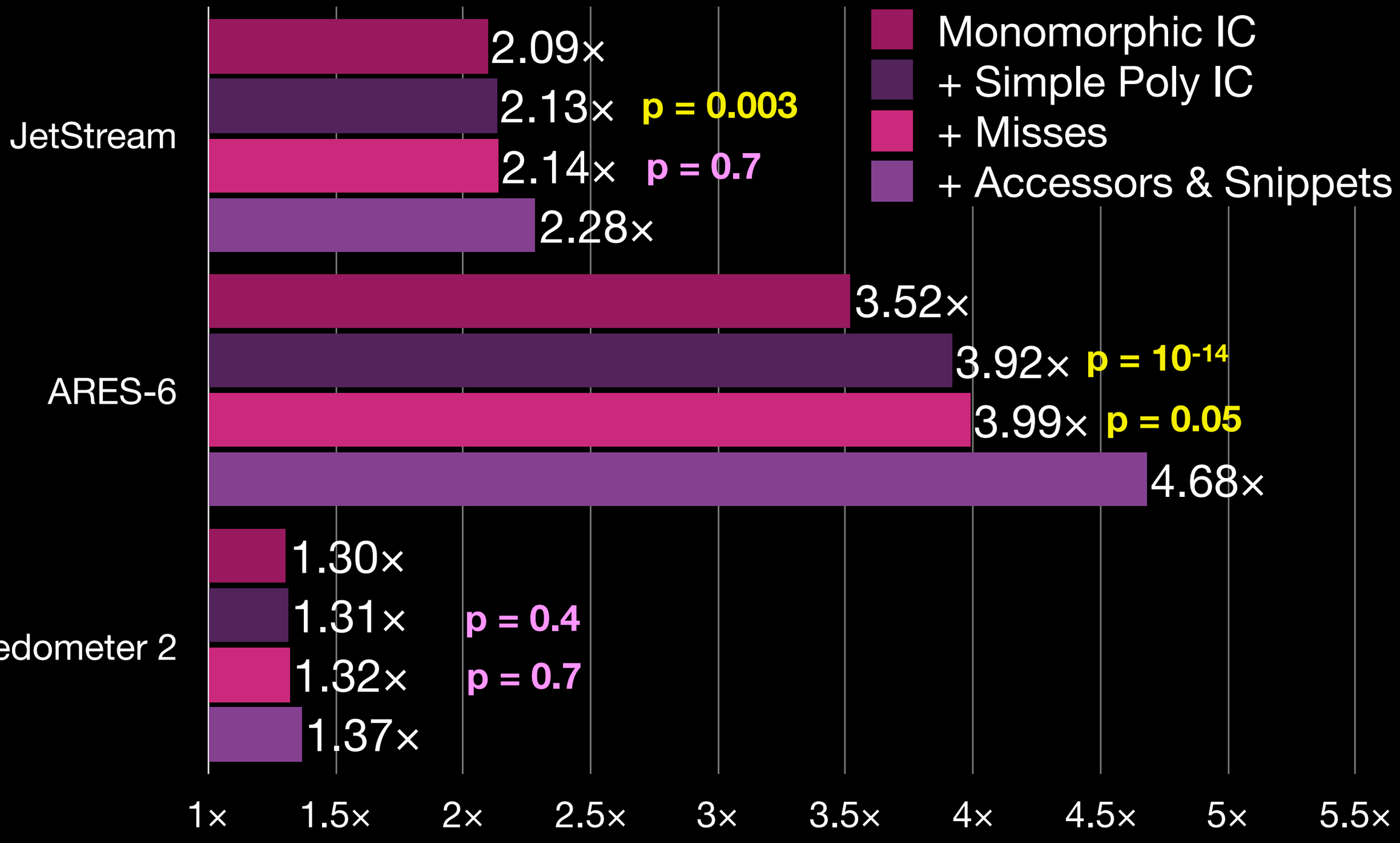
JIT Polymorphic IC speed-up



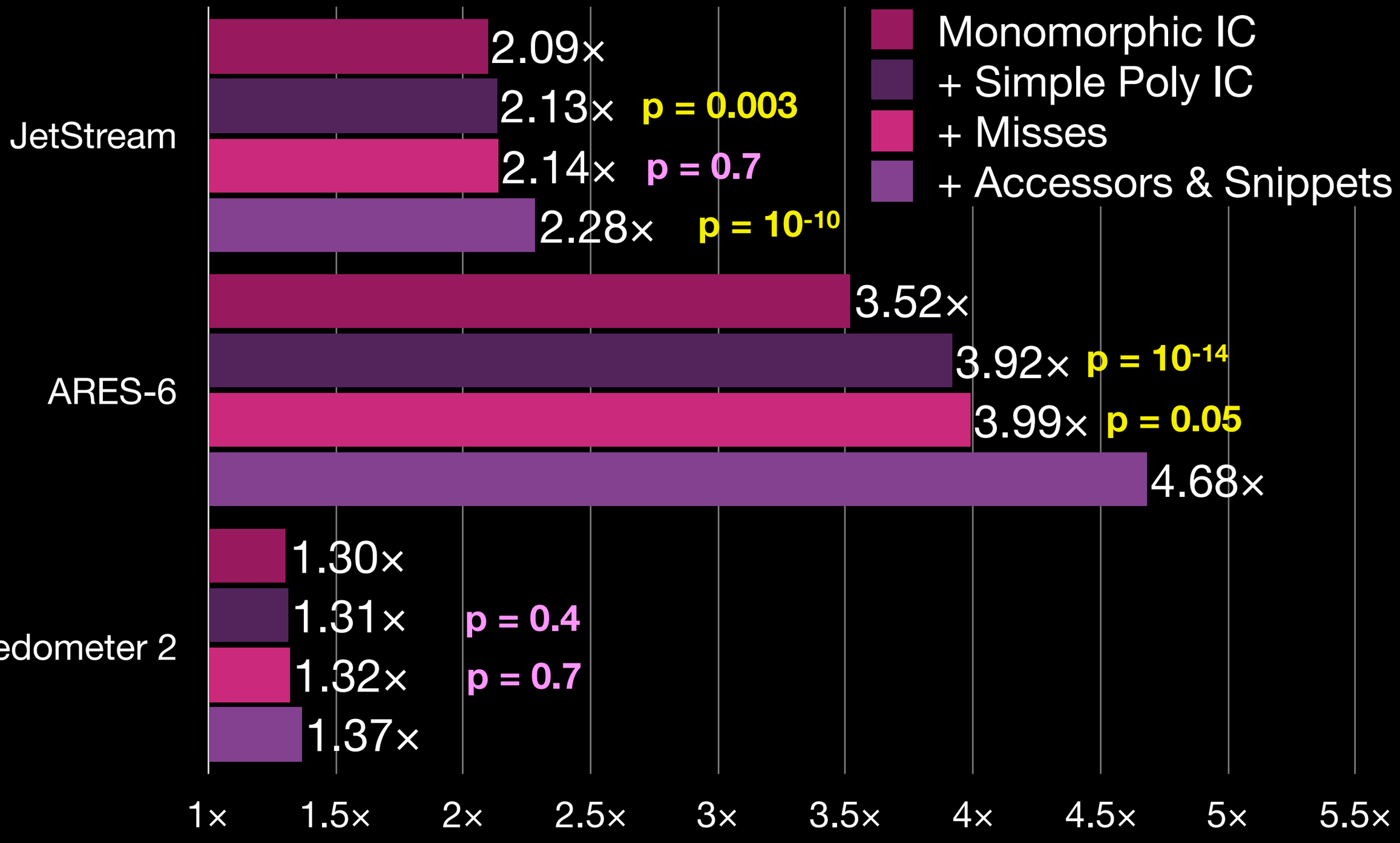
JIT Polymorphic IC speed-up



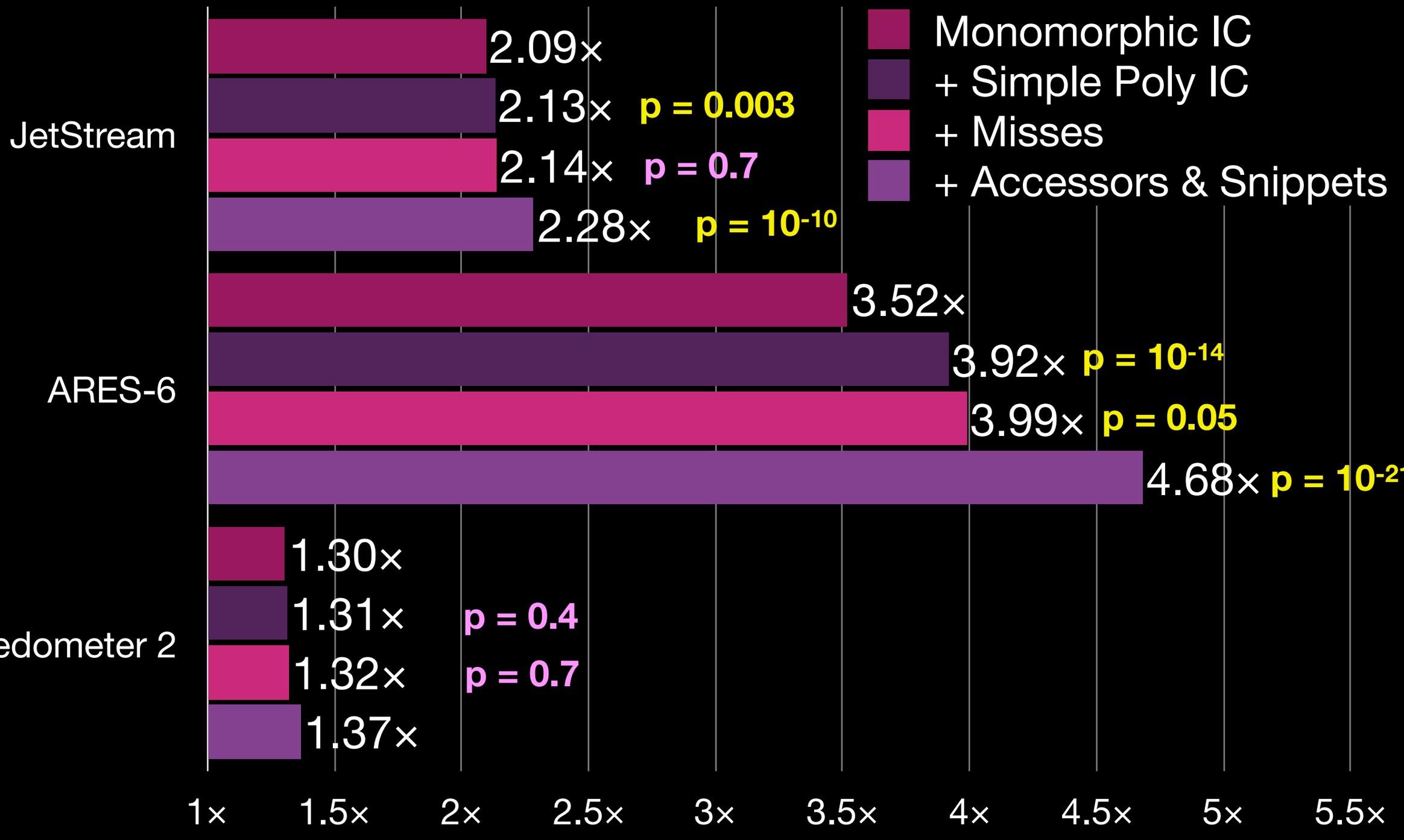
JIT Polymorphic IC speed-up



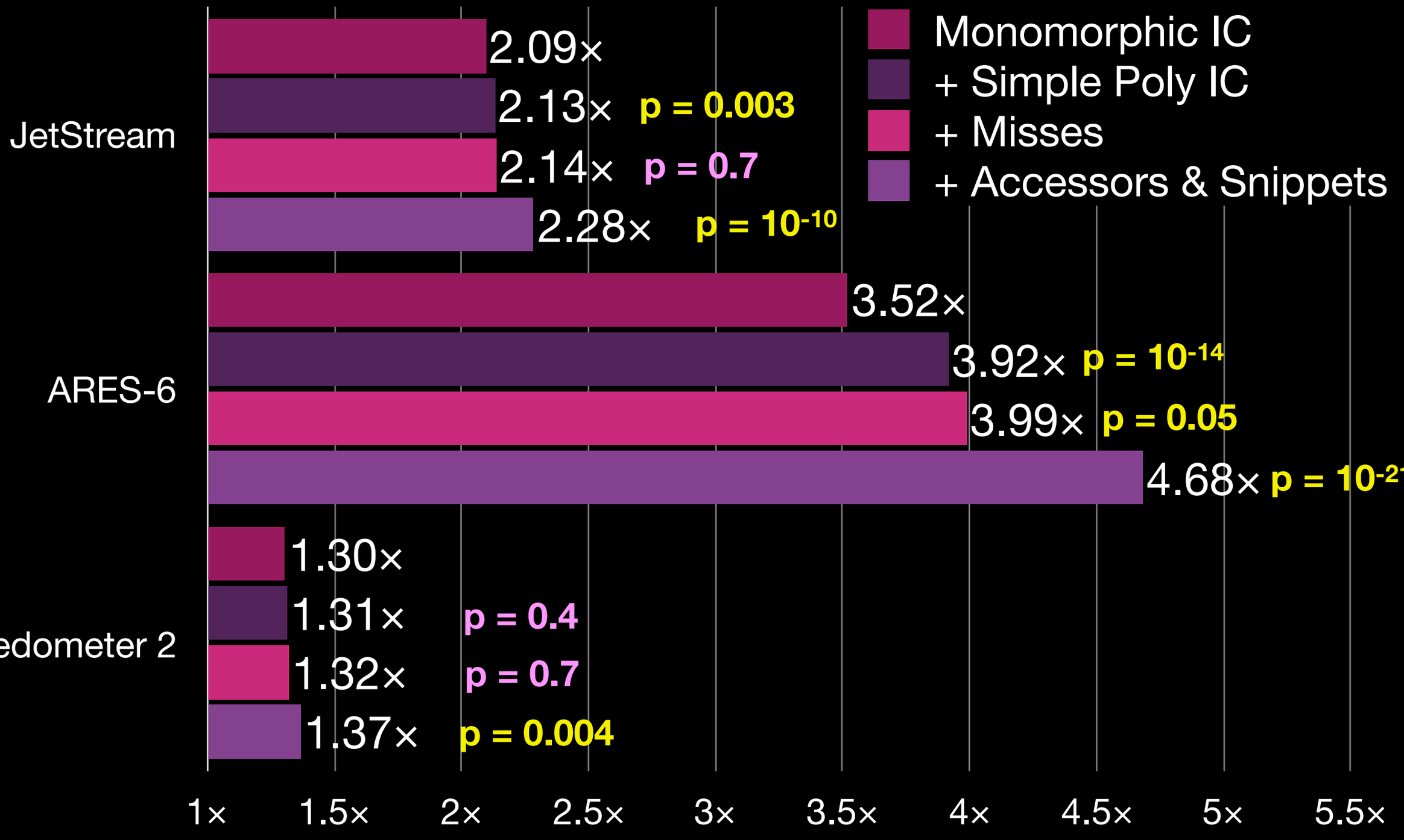
JIT Polymorphic IC speed-up



JIT Polymorphic IC speed-up

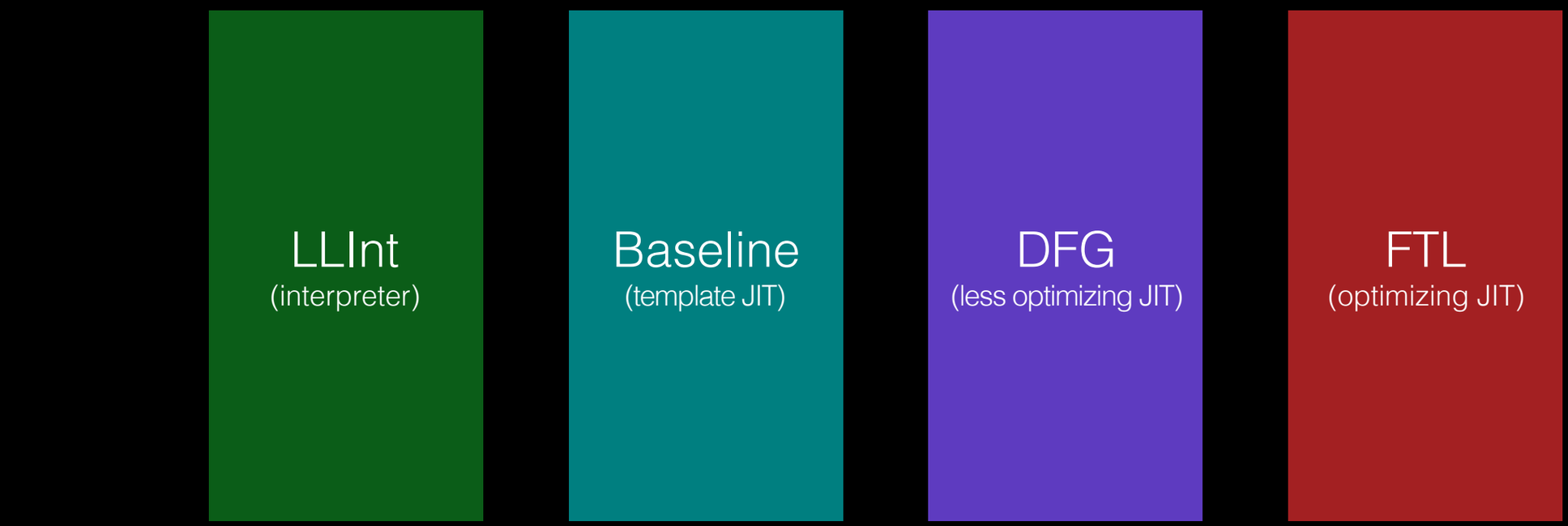


JIT Polymorphic IC speed-up



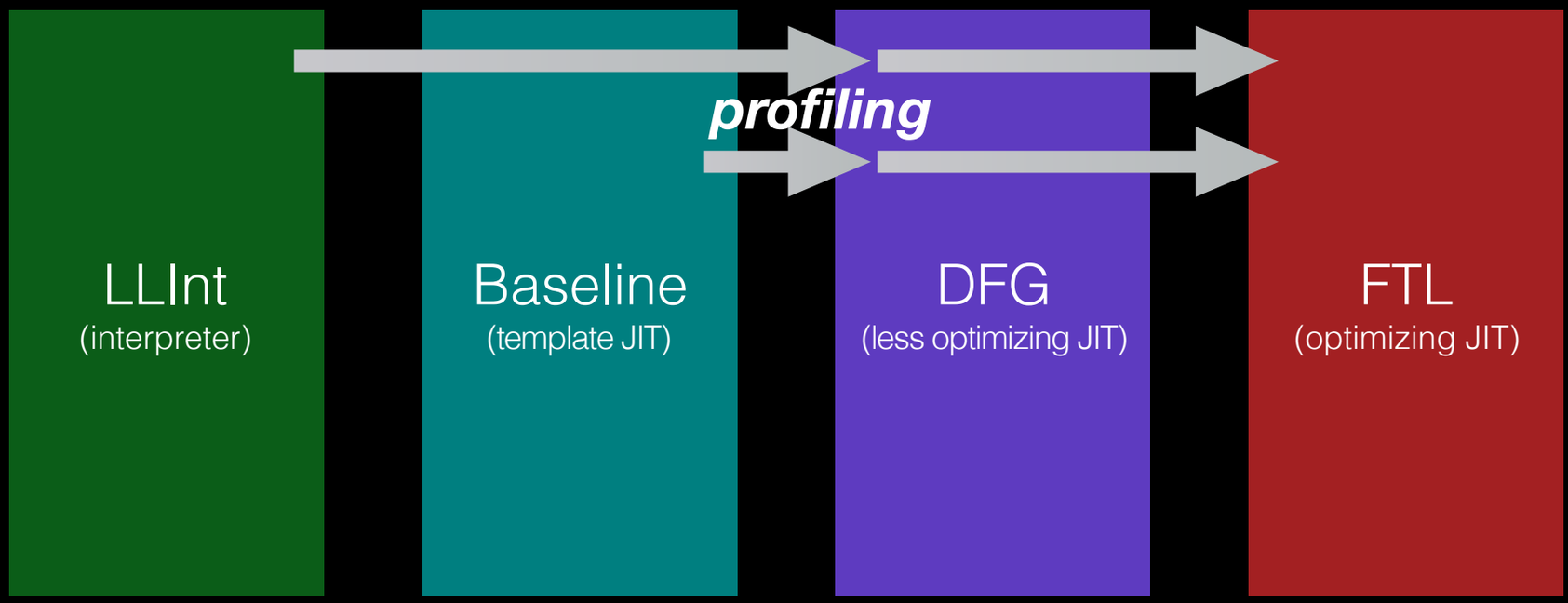
Inlining Inline Caches

Four Tiers


LLInt
(interpreter)
Baseline
(template JIT)
DFG
(less optimizing JIT)
FTL
(optimizing JIT)

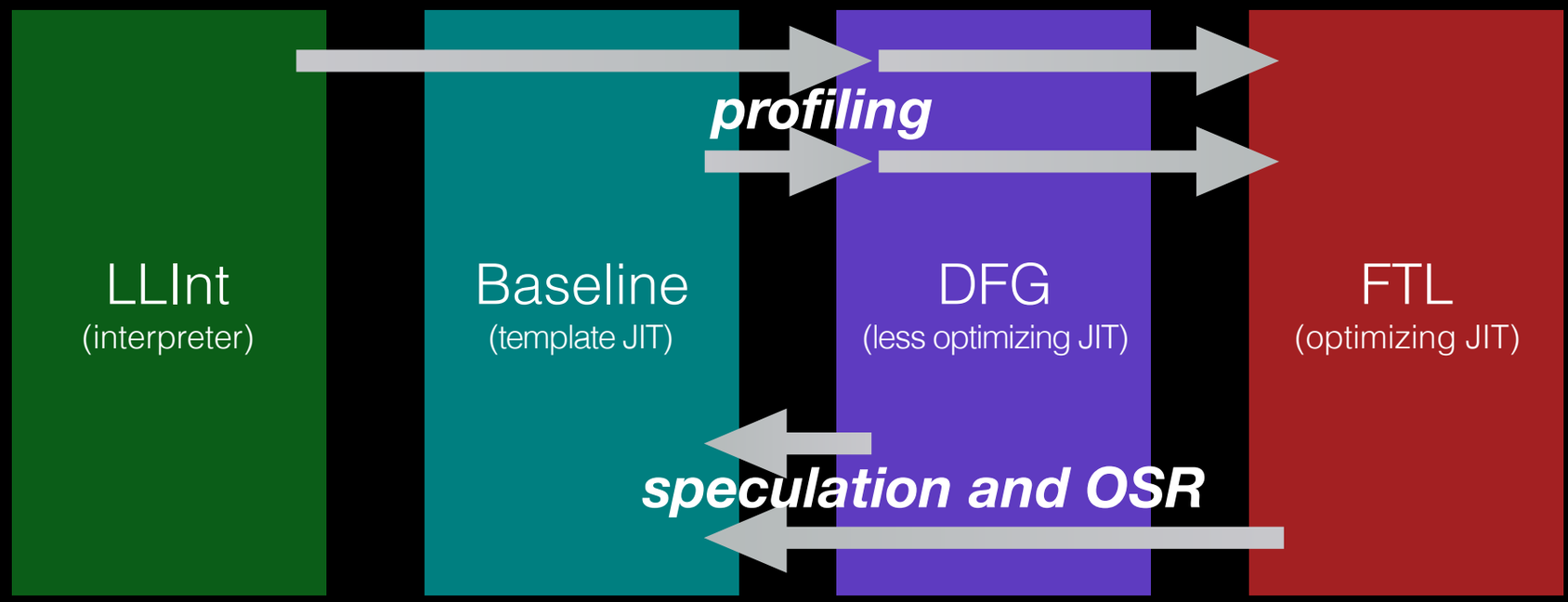

latency
throughput

Four Tiers



latency *throughput*

Four Tiers



latency *throughput*

LLInt
(interpreter)

get_by_id

Baseline
(template JIT)

jmp Lslow

DFG
(less optimizing JIT)

jmp Lslow

FTL
(optimizing JIT)

jmp Lslow

LLInt
(interpreter)

```
get_by_id
…, S1, 0
```

Baseline
(template JIT)

```
jmp Lslow
```

DFG
(less optimizing JIT)

```
jmp Lslow
```

FTL
(optimizing JIT)

```
jmp Lslow
```

LLInt
(interpreter)

```
get_by_id
…, S1, 0
```

Baseline
(template JIT)

```
cmp S1,
    (%rax)
jnz Lslow
mov 10(%rax),
    %rax
```

DFG
(less optimizing JIT)

```
jmp Lslow
```

FTL
(optimizing JIT)

```
jmp Lslow
```

LLInt	Baseline	DFG	FTL
(interpreter)	(template JIT)	(less optimizing JIT)	(optimizing JIT)

`get_by_id …, S1, 0`	`cmp S1, (%rax)`
	`jnz Lslow`
	`mov 10(%rax), %rax`
		`cmp S1, (%rax)`
		`jnz Lslow`
		`mov 10(%rax), %rax`
			`jmp Lslow`

LLInt
(interpreter)

```
get_by_id
 …, S1, 0
```

Baseline
(template JIT)

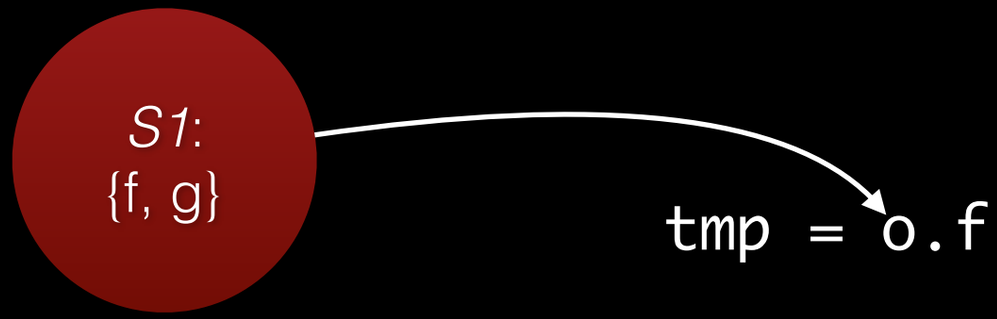
```
cmp S1,
    (%rax)
jnz Lslow
mov 10(%rax),
    %rax
```

DFG
(less optimizing JIT)

```
cmp S1,
    (%rax)
jnz Lslow
mov 10(%rax),
    %rax
```

FTL
(optimizing JIT)

```
cmp S1,
    (%rax)
jnz Lslow
mov 10(%rax),
    %rax
```



LLInt
(interpreter)

get_by_id

LLInt
(interpreter)

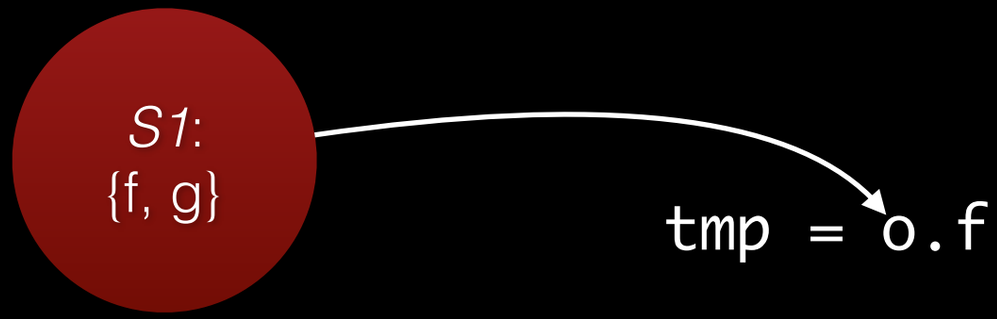
```
get_by_id
…, S1, 0
```

LLInt
(interpreter)

```
get_by_id
…, S1, 0
```

Baseline
(template JIT)

```
jmp Lslow
```



LLInt
(interpreter)

```
get_by_id
…, S1, 0
```

Baseline
(template JIT)

```
cmp S1,
    (%rax)
jnz Lslow
mov 10(%rax),
    %rax
```

LLInt
(interpreter)

```
get_by_id
…, S1, 0
```

Baseline
(template JIT)

```
cmp S1,
    (%rax)
jnz Lslow
mov 10(%rax),
    %rax
```

DFG
(less optimizing JIT)

LLInt
(interpreter)

```
get_by_id
…, S1, 0
```

Baseline
(template JIT)

```
cmp S1,
    (%rax)
jnz Lslow
mov 10(%rax),
    %rax
```

DFG
(less optimizing JIT)

```
CheckStructure(@o, S1)
GetByOffset(@o, “f”, 0)
```

LLInt
(interpreter)

```
get_by_id
…, S1, 0
```

Baseline
(template JIT)

```
cmp S1,
    (%rax)
jnz Lslow
mov 10(%rax),
    %rax
```

DFG
(less optimizing JIT)

```
CheckStructure(@o, S1)
GetByOffset(@o, “f”, 0)
```

FTL
(optimizing JIT)

LLInt
(interpreter)

```
get_by_id
…, S1, 0
```

Baseline
(template JIT)

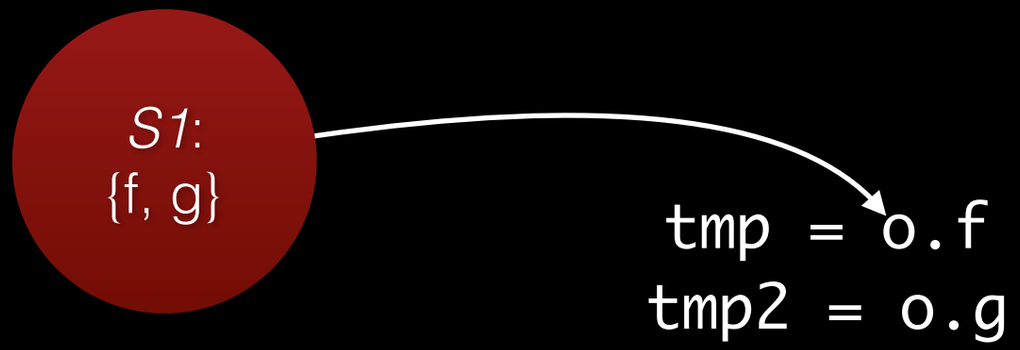
```
cmp S1,
    (%rax)
jnz Lslow
mov 10(%rax),
    %rax
```

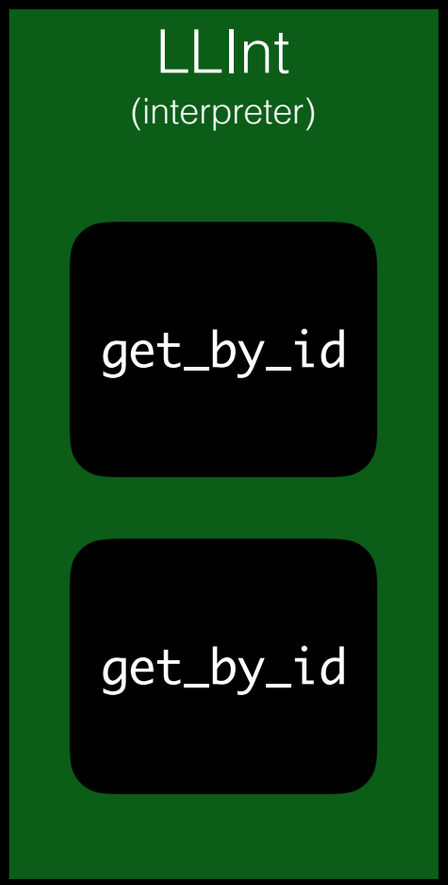
DFG
(less optimizing JIT)

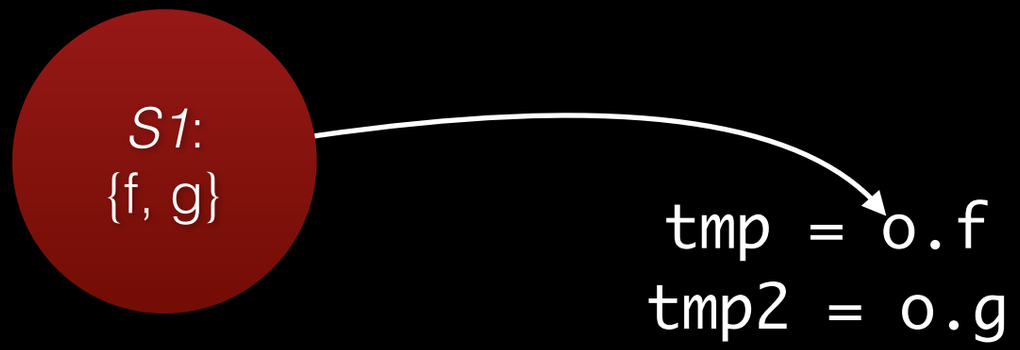
```
CheckStructure(@o, S1)
GetByOffset(@o, “f”, 0)
```

FTL
(optimizing JIT)

```
CheckStructure(@o, S1)
GetByOffset(@o, “f”, 0)
```


S1:
{f, g}
tmp = o.f
tmp2 = o.g

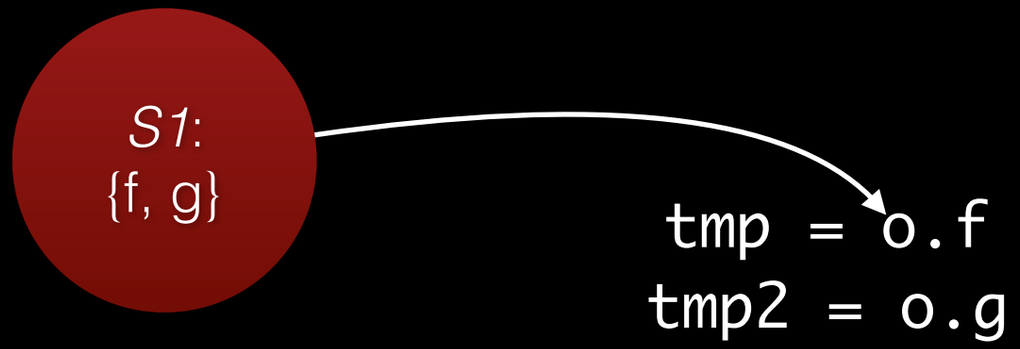

LLInt
(interpreter)
get_by_id
get_by_id



LLInt
(interpreter)

```
get_by_id
…, S1, 0
```

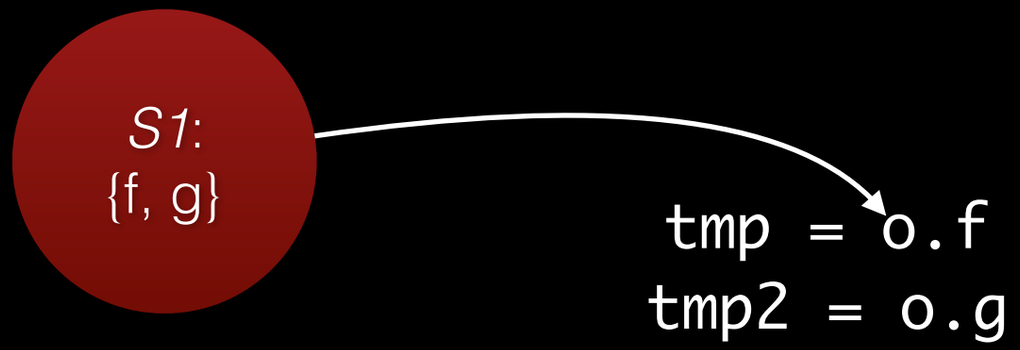
```
get_by_id
```



LLInt
(interpreter)

```
get_by_id
…, S1, 0
```

```
get_by_id
…, S1, 1
```



LLInt
(interpreter)

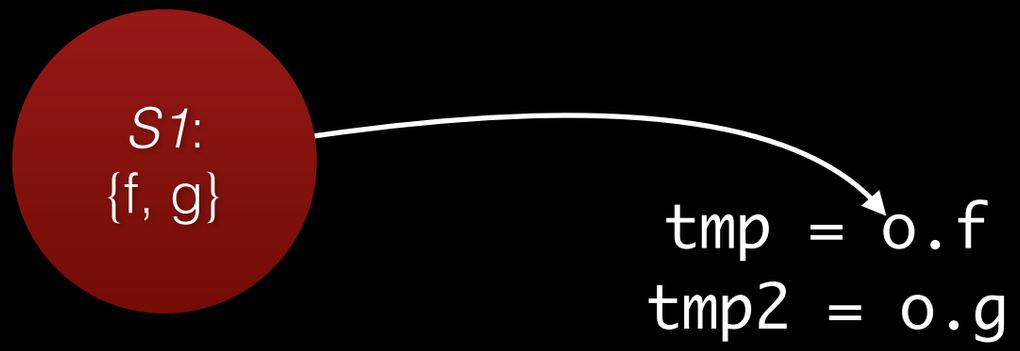
```
get_by_id
…, S1, 0
```

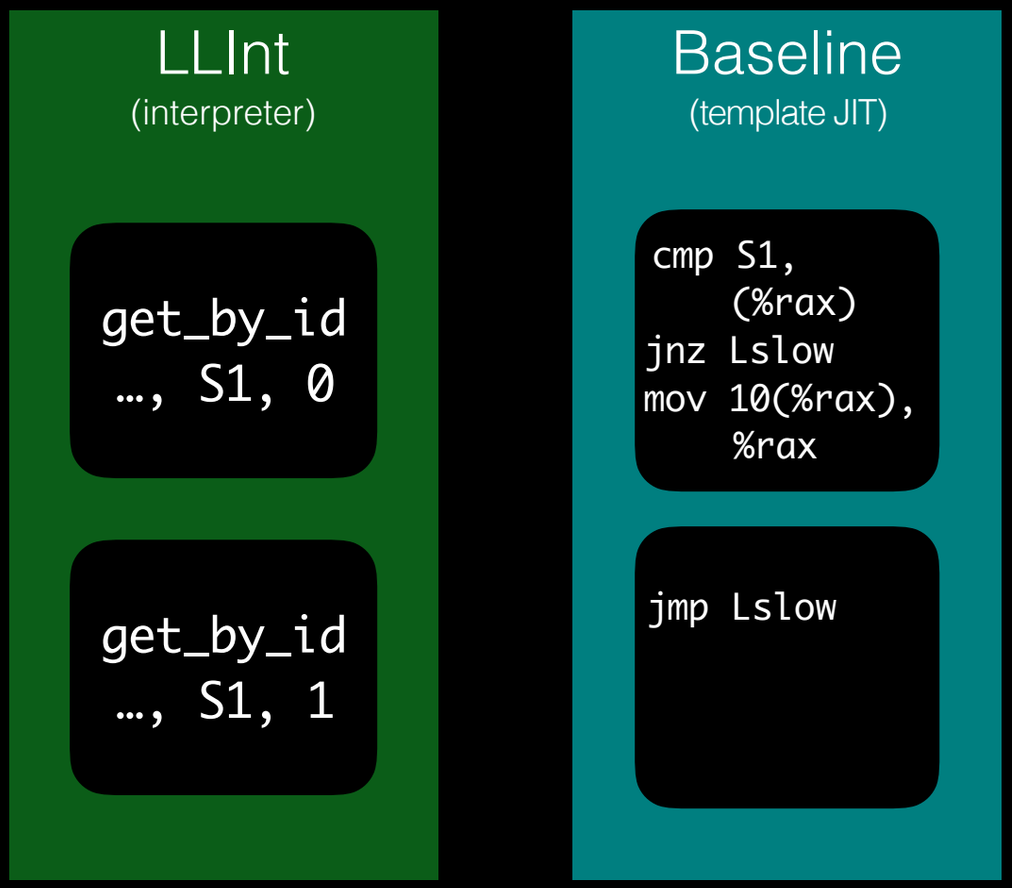
```
get_by_id
…, S1, 1
```

Baseline
(template JIT)

```
jmp Lslow
```

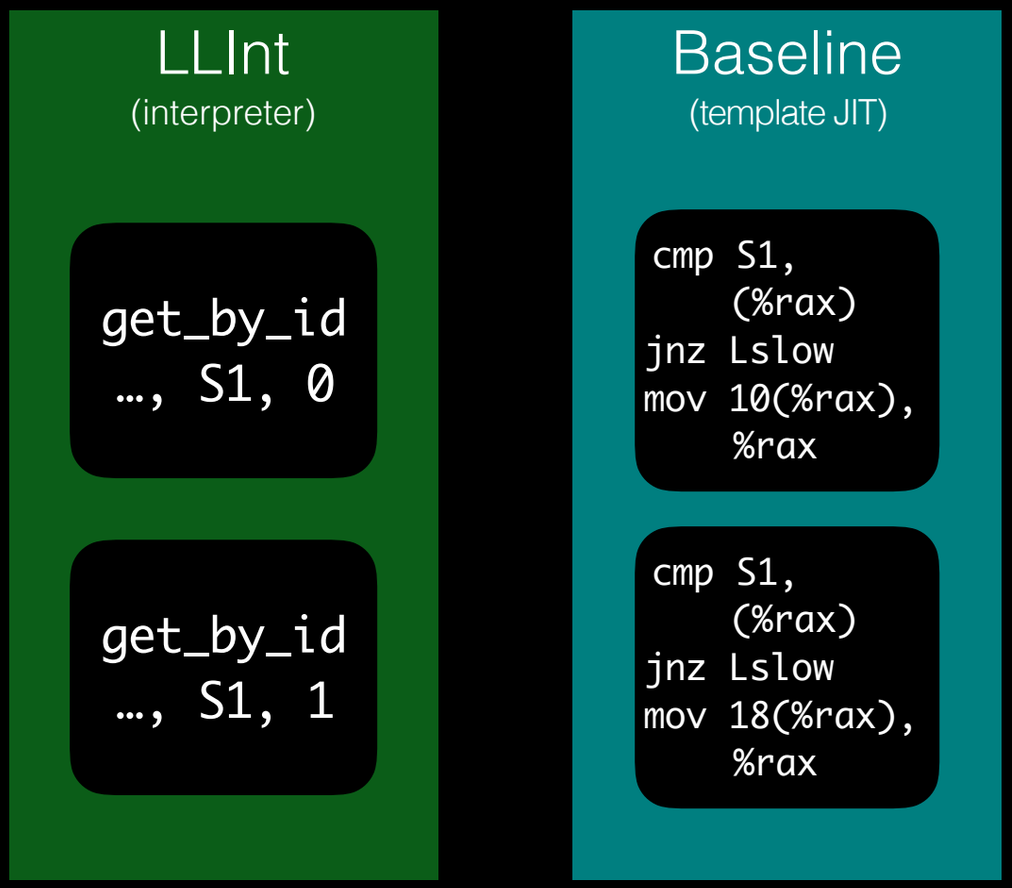
```
jmp Lslow
```

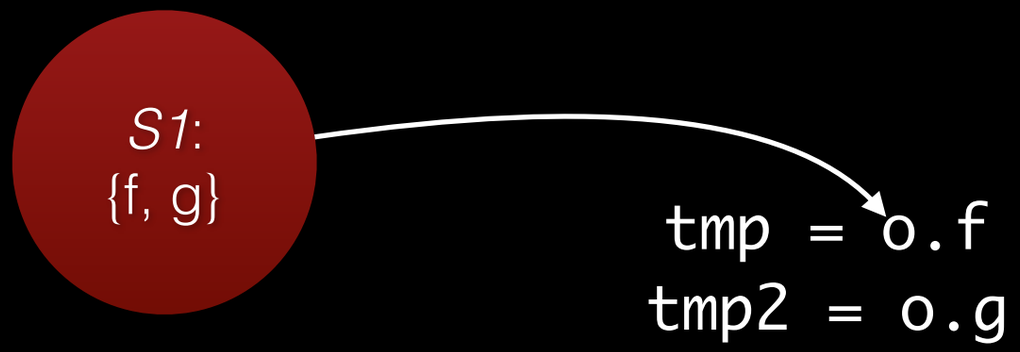

S1:
{f, g}
tmp = o.f
tmp2 = o.g


LLInt
(interpreter)
get_by_id
…, S1, 0
get_by_id
…, S1, 1

Baseline
(template JIT)
cmp S1,
(%rax)
jnz Lslow
mov 10(%rax),
%rax
jmp Lslow


S1:
{f, g}
tmp = o.f
tmp2 = o.g


LLInt
(interpreter)
get_by_id
…, S1, 0
get_by_id
…, S1, 1
Baseline
(template JIT)
cmp S1,
(%rax)
jnz Lslow
mov 10(%rax),
%rax
cmp S1,
(%rax)
jnz Lslow
mov 18(%rax),
%rax



LLInt
(interpreter)

```
get_by_id
…, S1, 0
```

```
get_by_id
…, S1, 1
```

Baseline
(template JIT)

```
cmp S1,
    (%rax)
jnz Lslow
mov 10(%rax),
    %rax
```

```
cmp S1,
    (%rax)
jnz Lslow
mov 18(%rax),
    %rax
```

DFG
(less optimizing JIT)

S1:
{f, g}

```
tmp = o.f
tmp2 = o.g
```

LLInt
(interpreter)

```
get_by_id
…, S1, 0
```

```
get_by_id
…, S1, 1
```

Baseline
(template JIT)

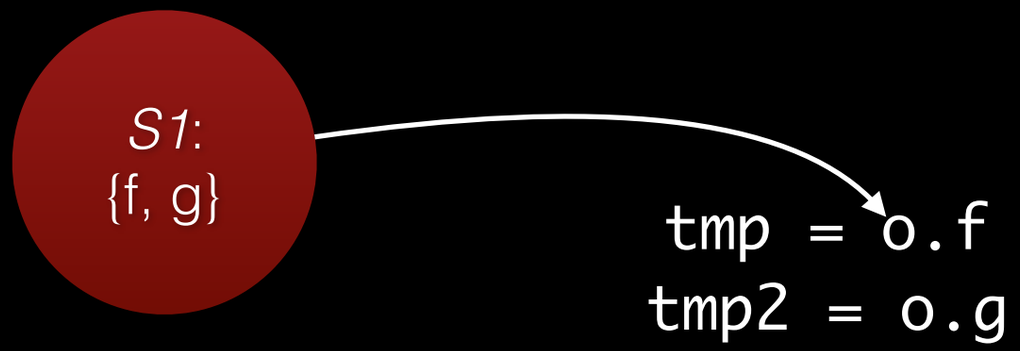
```
cmp S1,
    (%rax)
jnz Lslow
mov 10(%rax),
    %rax
```

```
cmp S1,
    (%rax)
jnz Lslow
mov 18(%rax),
    %rax
```

DFG
(less optimizing JIT)

```
CheckStructure(@o, S1)
GetByOffset(@o, “f”, 0)
```

```
CheckStructure(@o, S1)
GetByOffset(@o, “g”, 1)
```



LLInt
(interpreter)

```
get_by_id
…, S1, 0
```

```
get_by_id
…, S1, 1
```

Baseline
(template JIT)

```
cmp S1,
    (%rax)
jnz Lslow
mov 10(%rax),
    %rax
```

```
cmp S1,
    (%rax)
jnz Lslow
mov 18(%rax),
    %rax
```

DFG
(less optimizing JIT)

```
CheckStructure(@o, S1)
GetByOffset(@o, “f”, 0)
```

```
GetByOffset(@o, “g”, 1)
```

S1:
{f, g}

```
tmp = o.f
tmp2 = o.g
```

LLInt
(interpreter)

```
get_by_id
…, S1, 0
```

```
get_by_id
…, S1, 1
```

Baseline
(template JIT)

```
cmp S1,
    (%rax)
jnz Lslow
mov 10(%rax),
    %rax
```

```
cmp S1,
    (%rax)
jnz Lslow
mov 18(%rax),
    %rax
```

DFG
(less optimizing JIT)

```
CheckStructure(@o, S1)
GetByOffset(@o, “f”, 0)
```

```
GetByOffset(@o, “g”, 1)
```

FTL
(optimizing JIT)

S1:
{f, g}

```
tmp = o.f
tmp2 = o.g
```

LLInt
(interpreter)

```
get_by_id
…, S1, 0
```

```
get_by_id
…, S1, 1
```

Baseline
(template JIT)

```
cmp S1,
    (%rax)
jnz Lslow
mov 10(%rax),
    %rax
```

```
cmp S1,
    (%rax)
jnz Lslow
mov 18(%rax),
    %rax
```

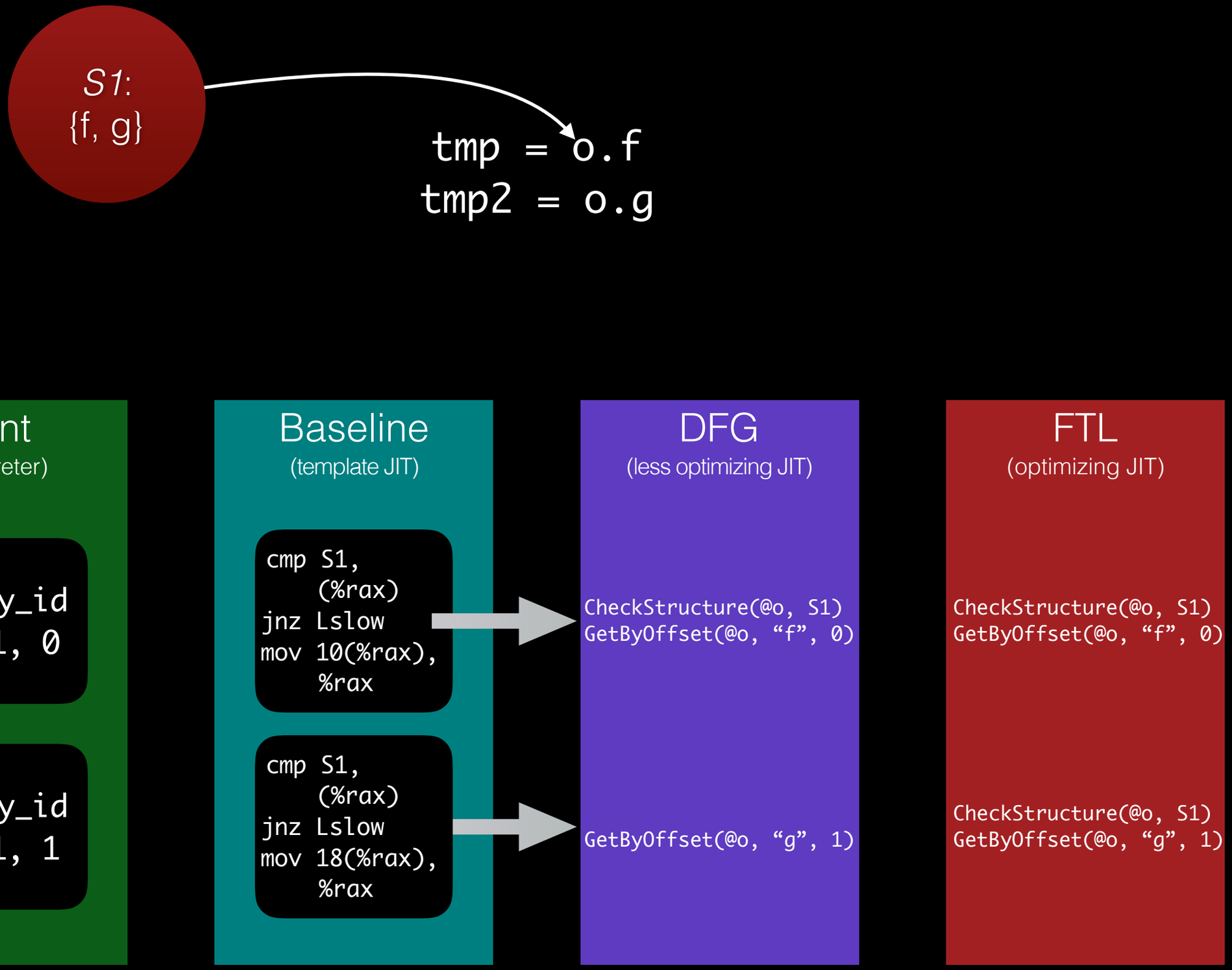
DFG
(less optimizing JIT)

```
CheckStructure(@o, S1)
GetByOffset(@o, “f”, 0)
```

```
GetByOffset(@o, “g”, 1)
```

FTL
(optimizing JIT)

```
CheckStructure(@o, S1)
GetByOffset(@o, “f”, 0)
```


S1:
{f, g}
tmp = o.f
tmp2 = o.g
LLInt
(interpreter)
get_by_id
…, S1, 0
get_by_id
…, S1, 1
Baseline
(template JIT)
cmp S1,
(%rax)
jnz Lslow
mov 10(%rax),
%rax
cmp S1,
(%rax)
jnz Lslow
mov 18(%rax),
%rax
DFG
(less optimizing JIT)
CheckStructure(@o, S1)
GetByOffset(@o, “f”, 0)
GetByOffset(@o, “g”, 1)
FTL
(optimizing JIT)
CheckStructure(@o, S1)
GetByOffset(@o, “f”, 0)
CheckStructure(@o, S1)
GetByOffset(@o, “g”, 1)

S1:
{f, g}

```
tmp = o.f
tmp2 = o.g
```

LLInt
(interpreter)

```
get_by_id
…, S1, 0
```

```
get_by_id
…, S1, 1
```

Baseline
(template JIT)

```
cmp S1,
    (%rax)
jnz Lslow
mov 10(%rax),
    %rax
```

```
cmp S1,
    (%rax)
jnz Lslow
mov 18(%rax),
    %rax
```

DFG
(less optimizing JIT)

```
CheckStructure(@o, S1)
GetByOffset(@o, “f”, 0)
```

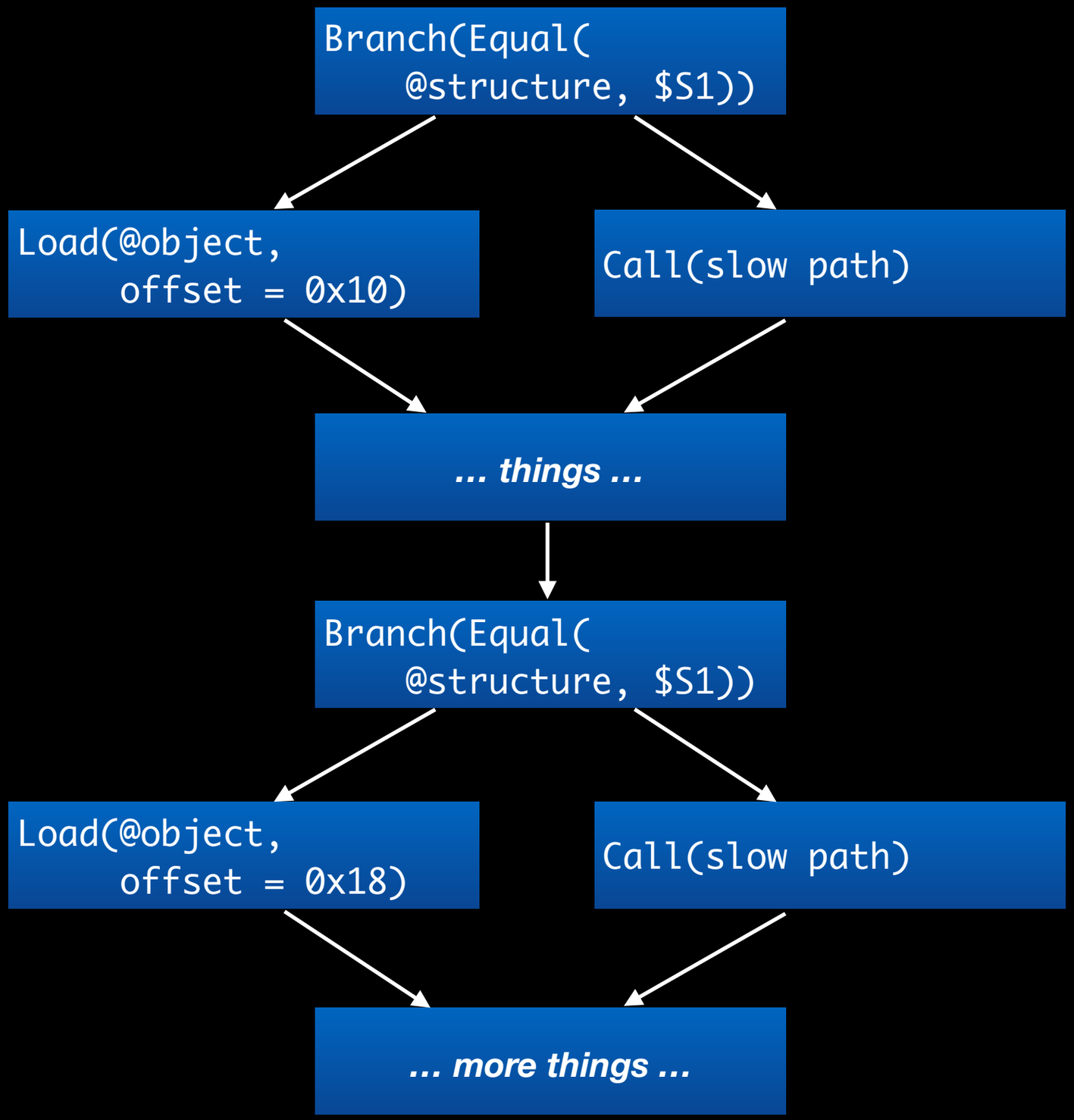
```
GetByOffset(@o, “g”, 1)
```

FTL
(optimizing JIT)

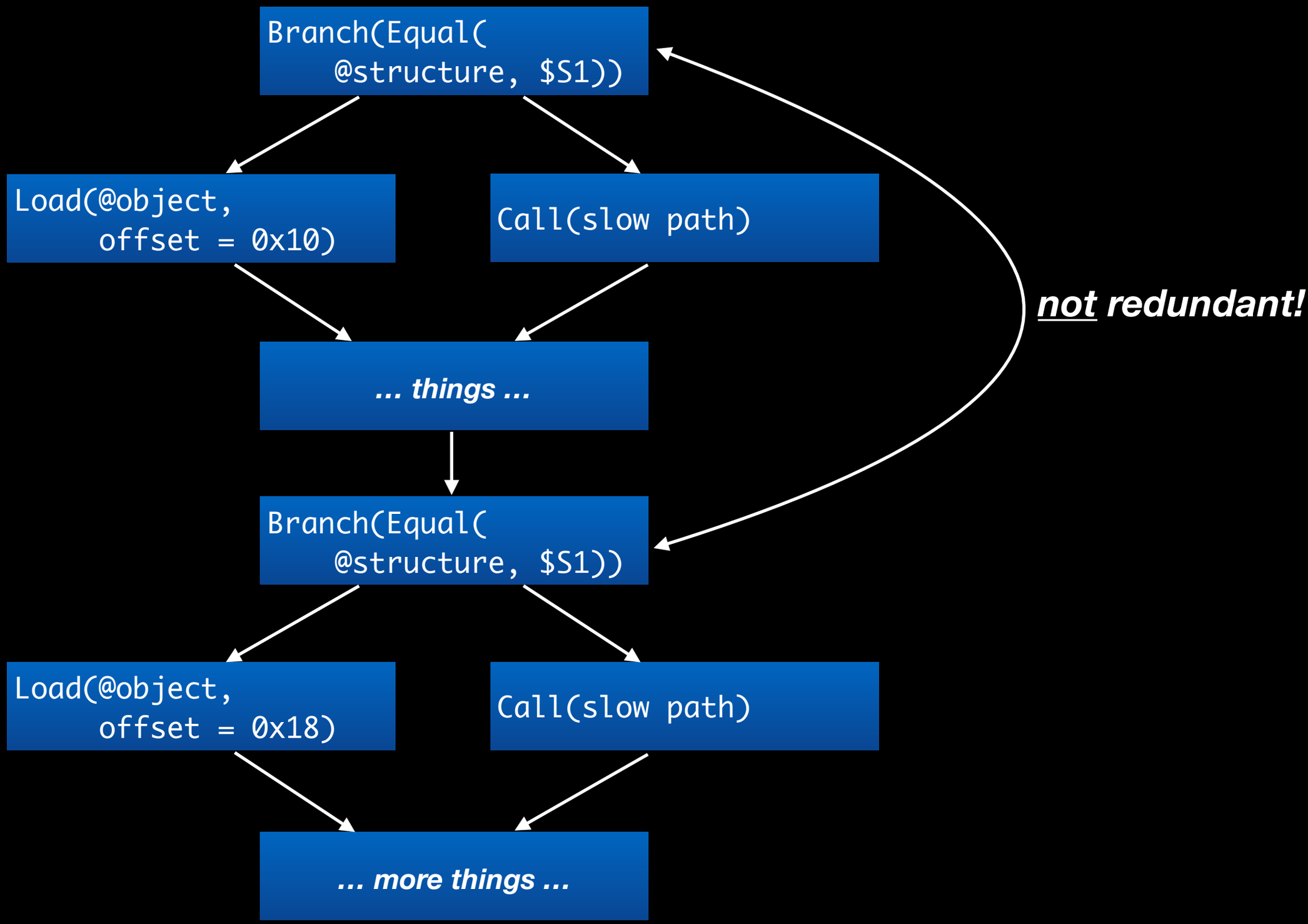
```
CheckStructure(@o, S1)
GetByOffset(@o, “f”, 0)
```

```
GetByOffset(@o, “g”, 1)
```

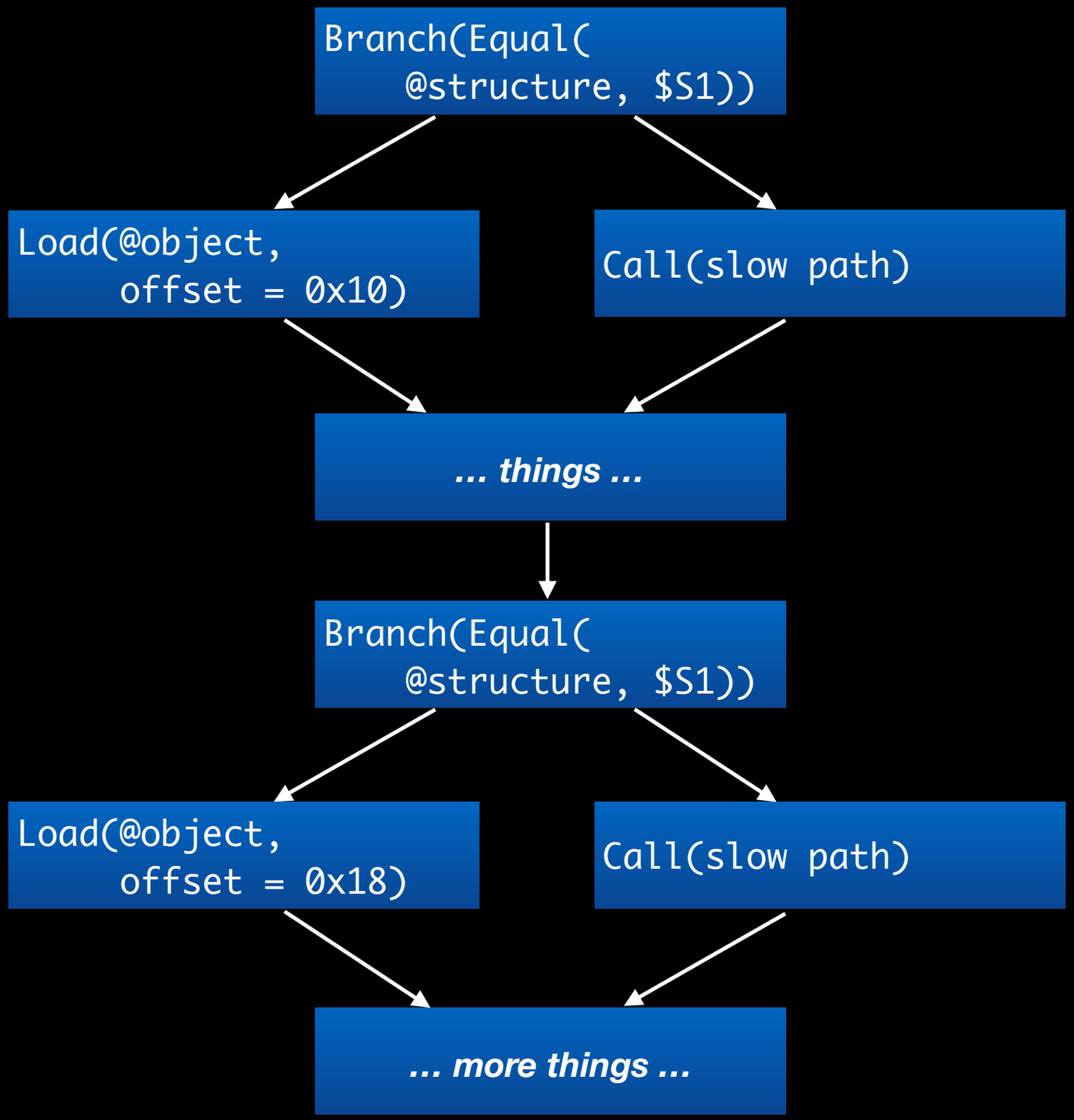
Inline Cache Control Flow



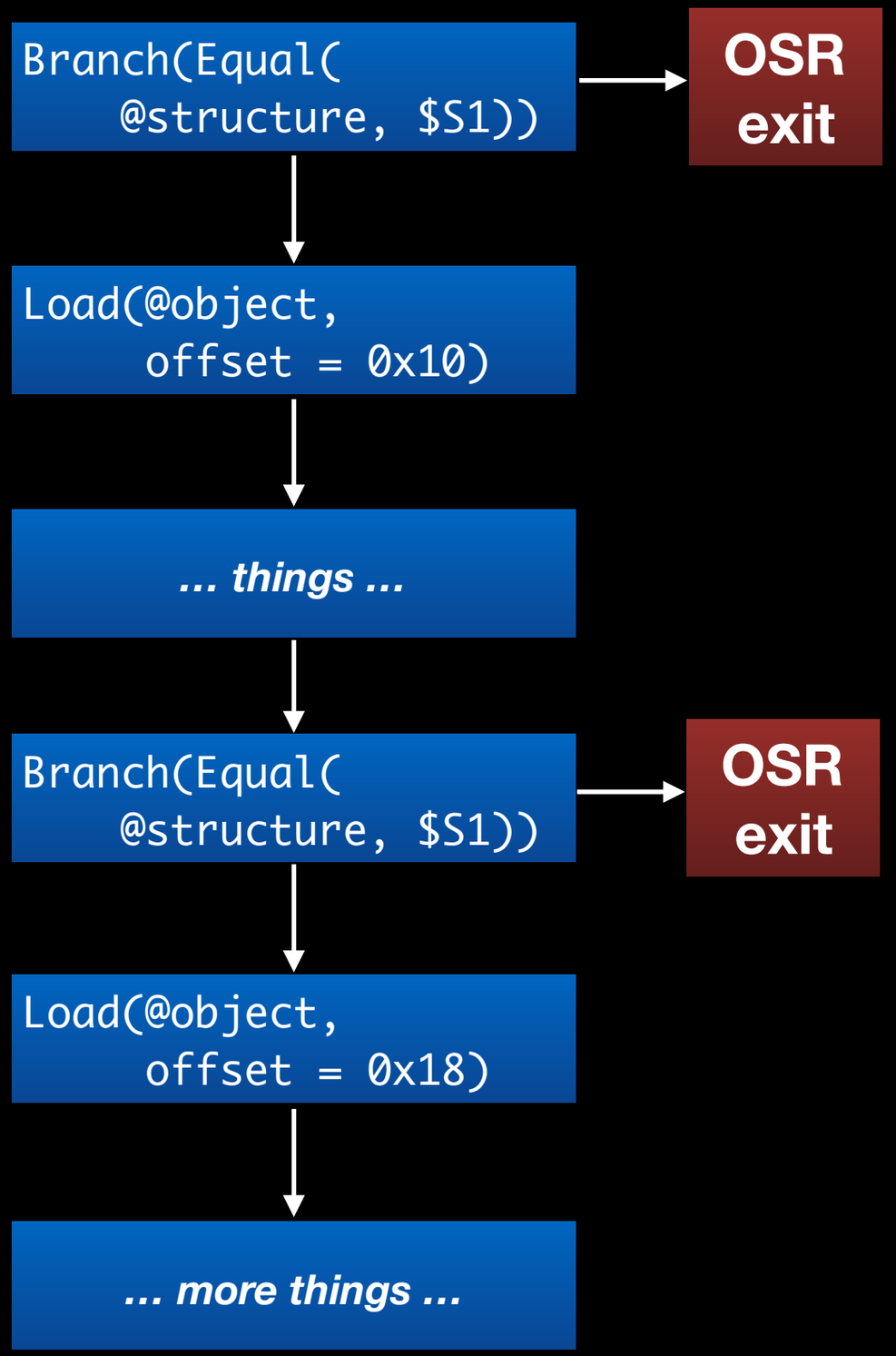
Inline Cache Control Flow



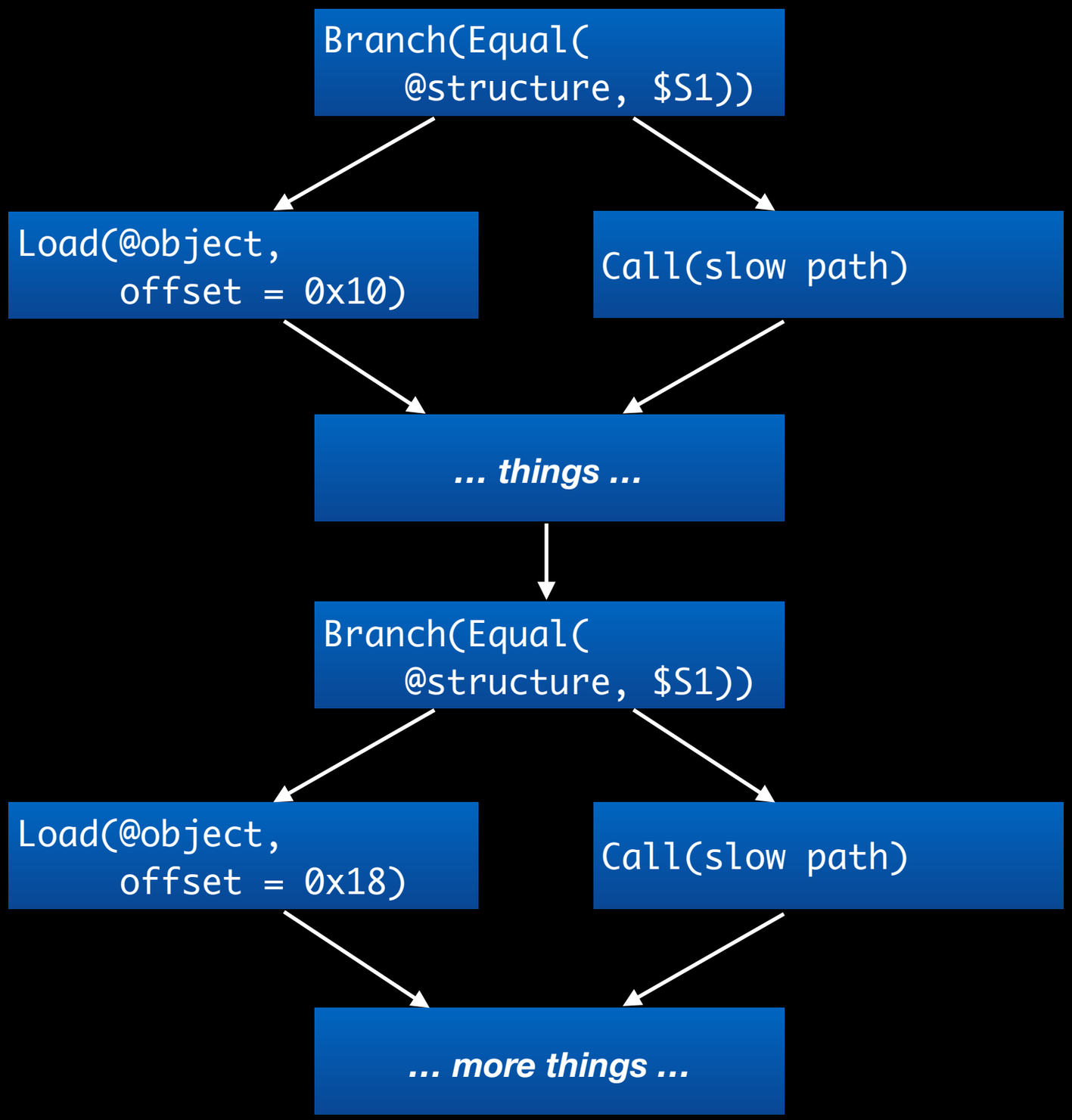
Inline Cache Control Flow



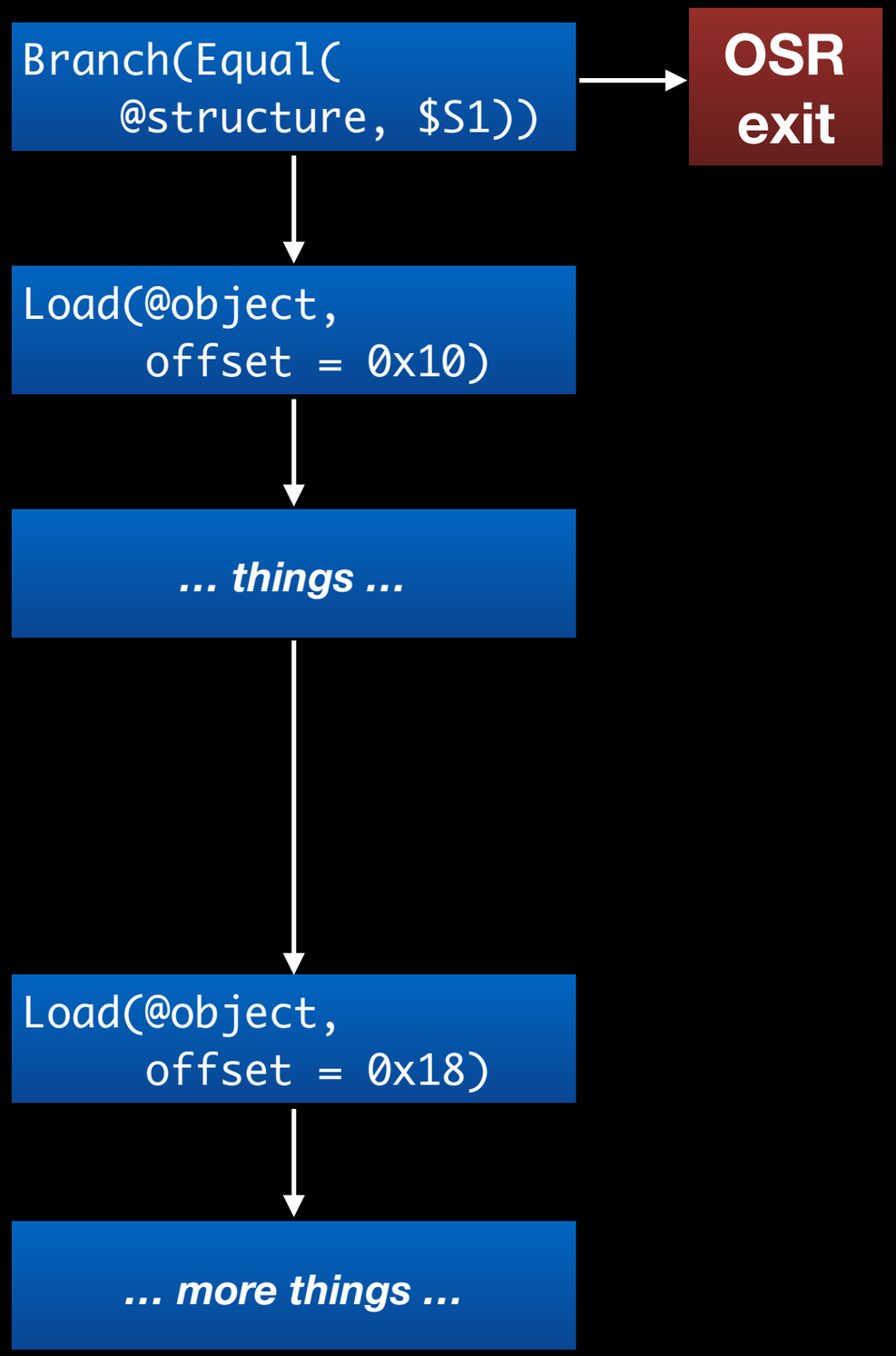
Inlined with OSR exits



Inline Cache Control Flow



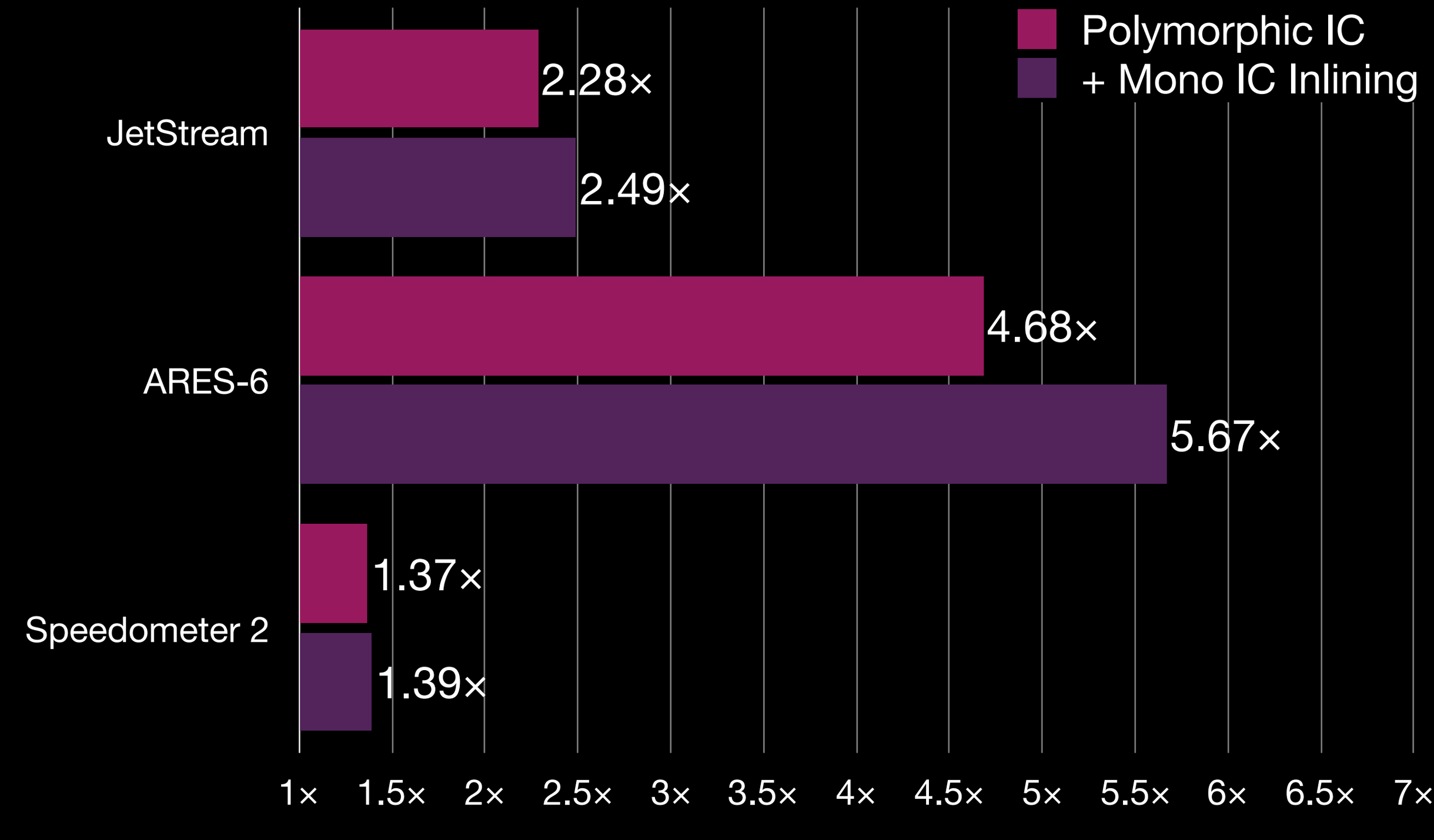
Inlined with OSR exits



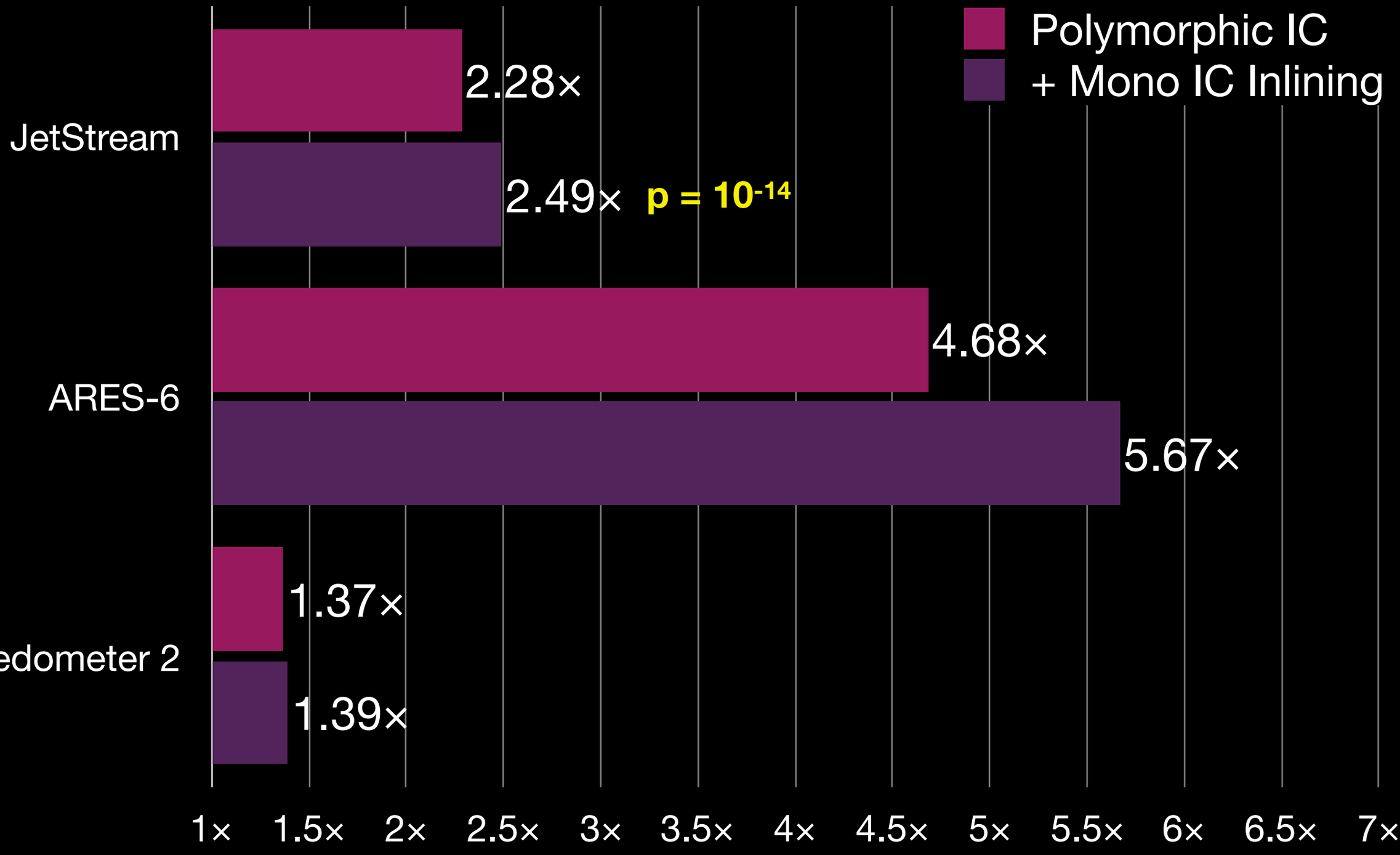
Benefits of Inlining

- Common subexpression elimination
- Abstract interpreter models the structure
- Object allocation sinking
- Many other optimizations

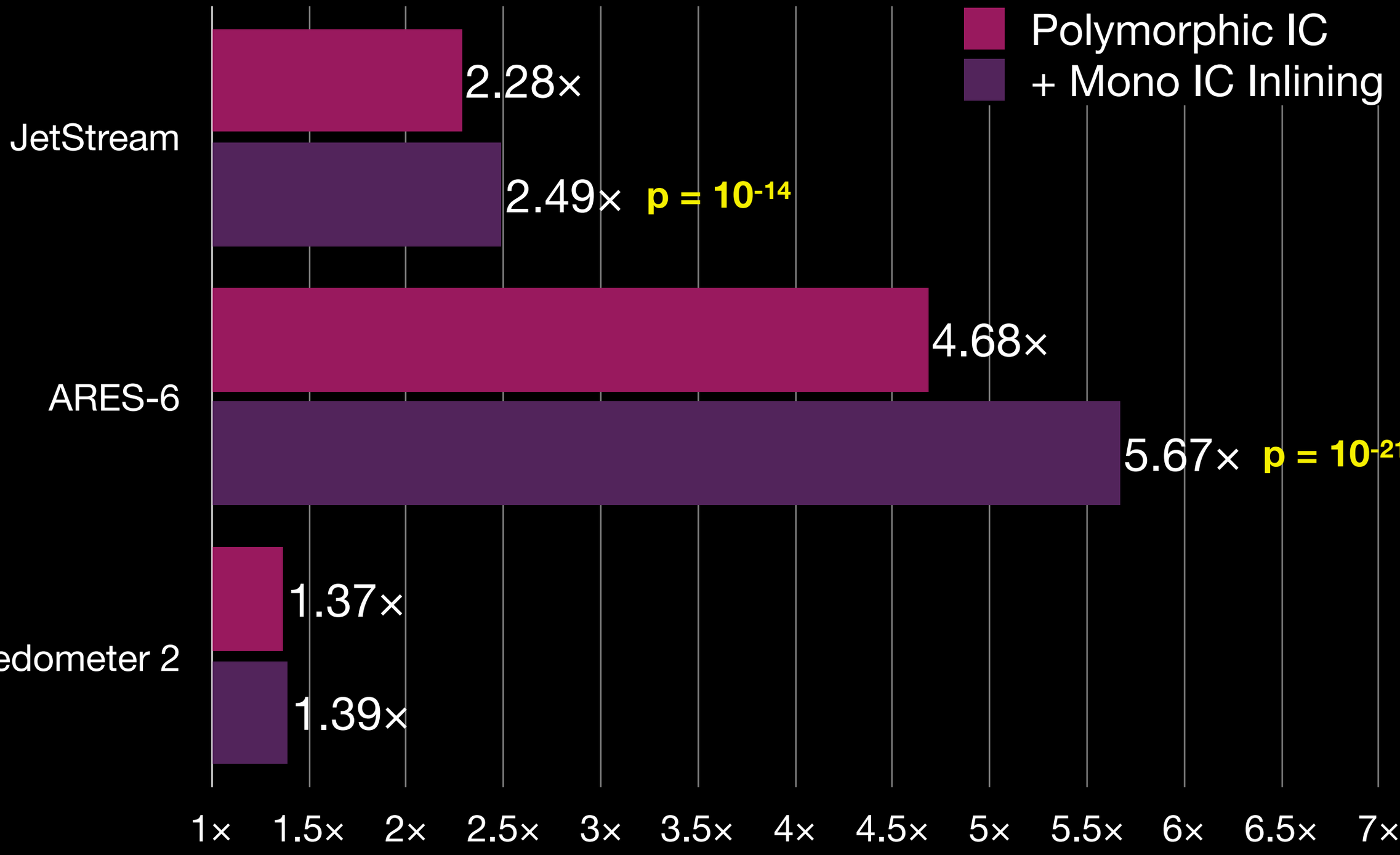
IC Monomorphic Inlining Speed-up



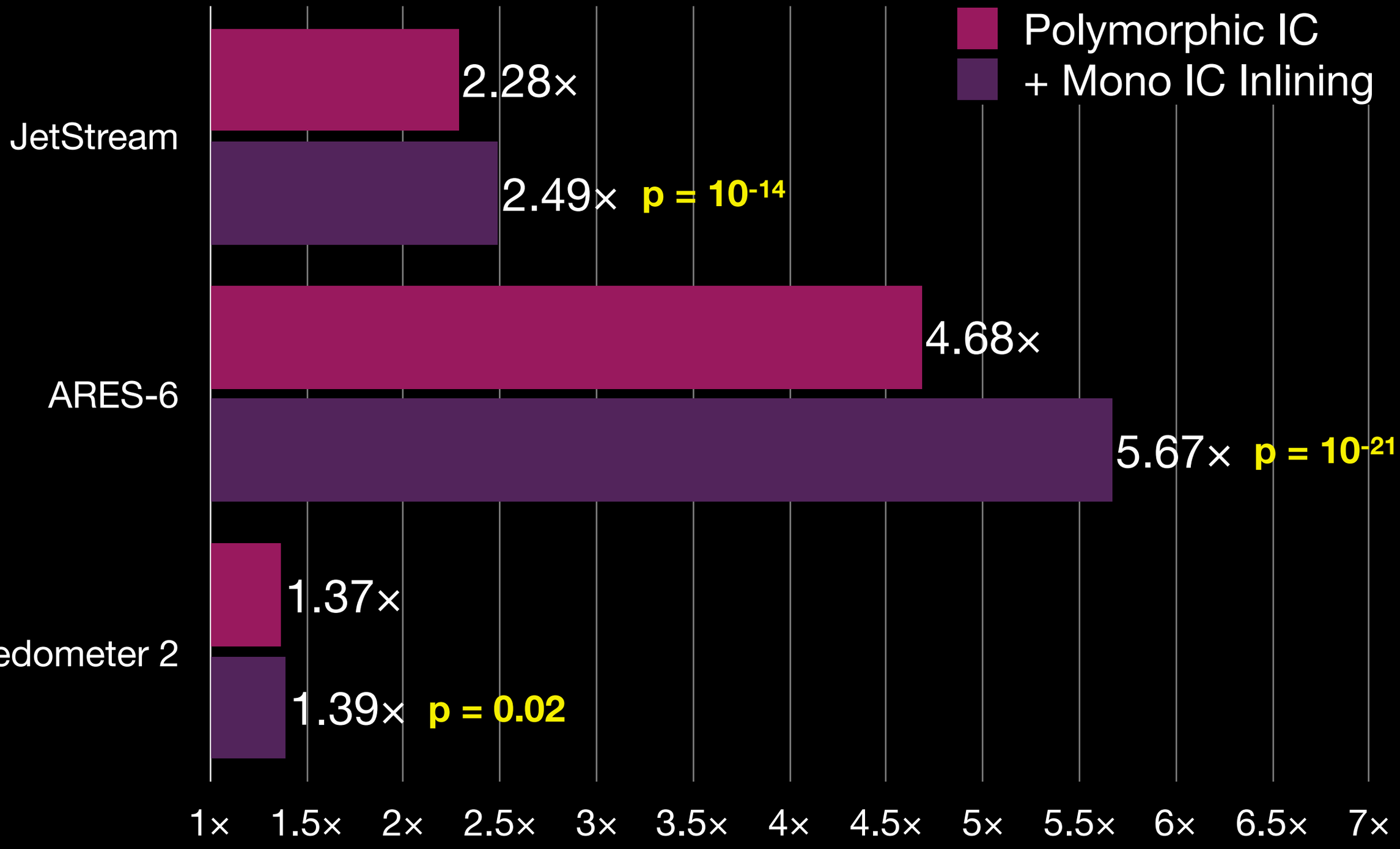
IC Monomorphic Inlining Speed-up



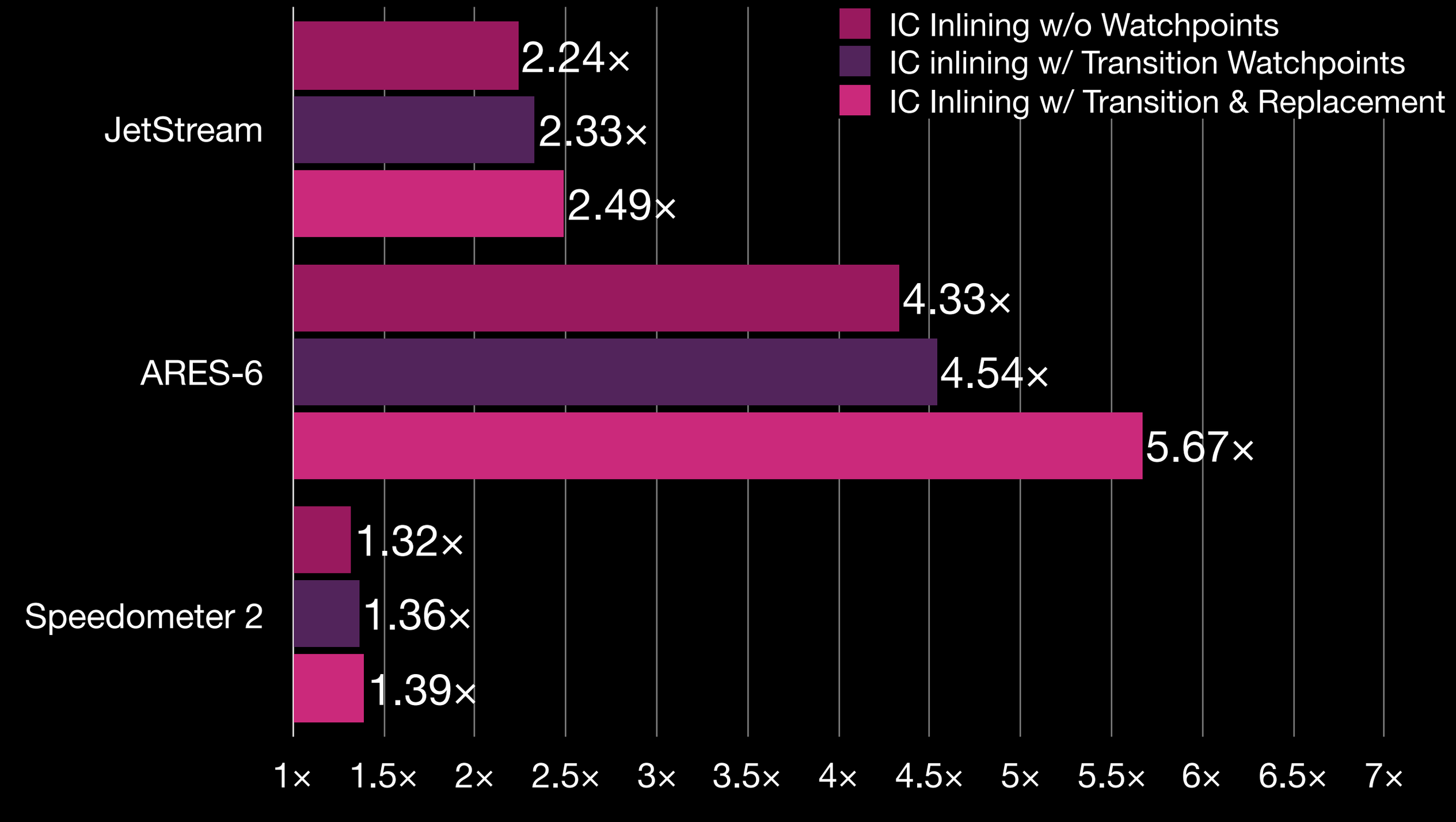
IC Monomorphic Inlining Speed-up



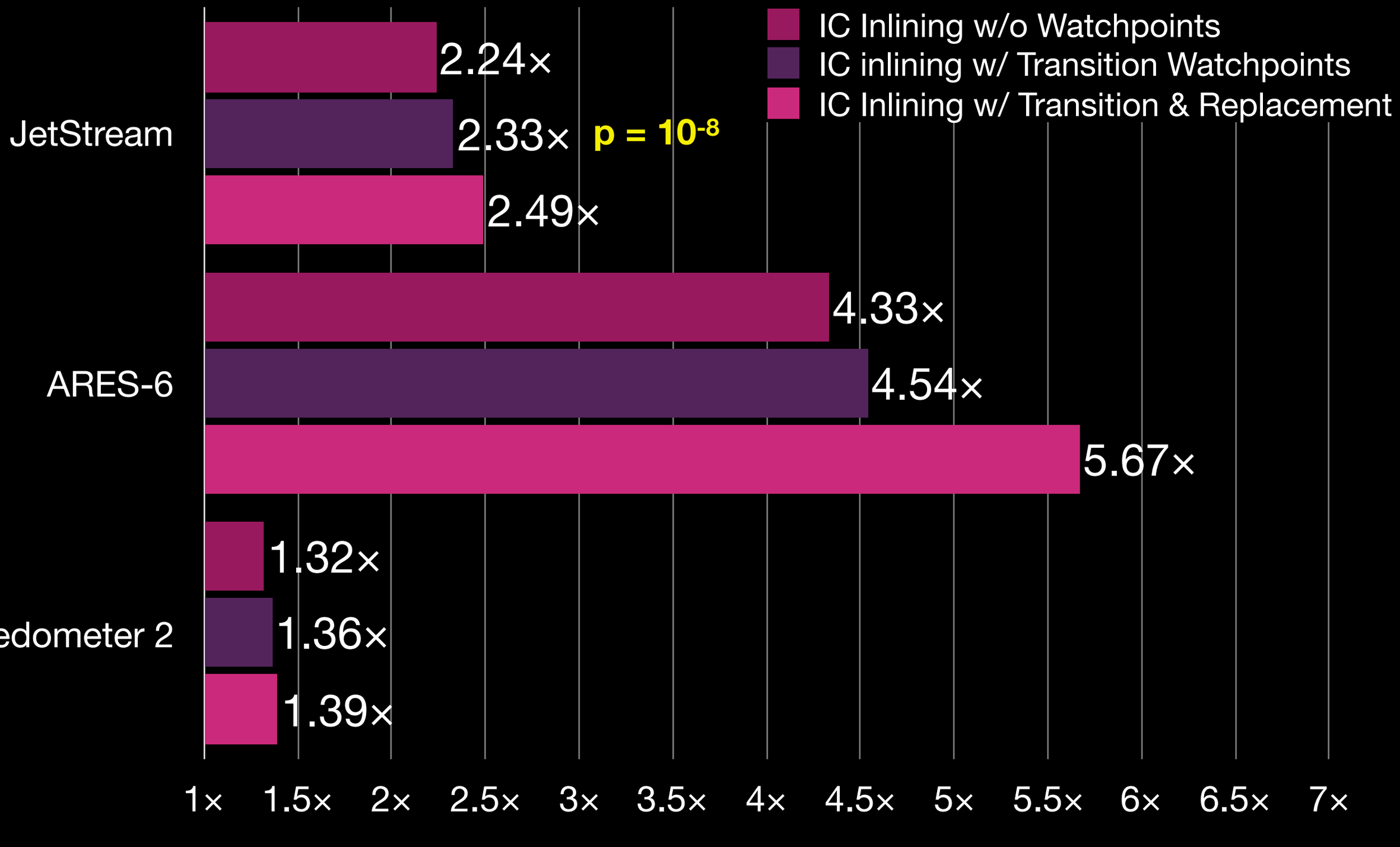
IC Monomorphic Inlining Speed-up



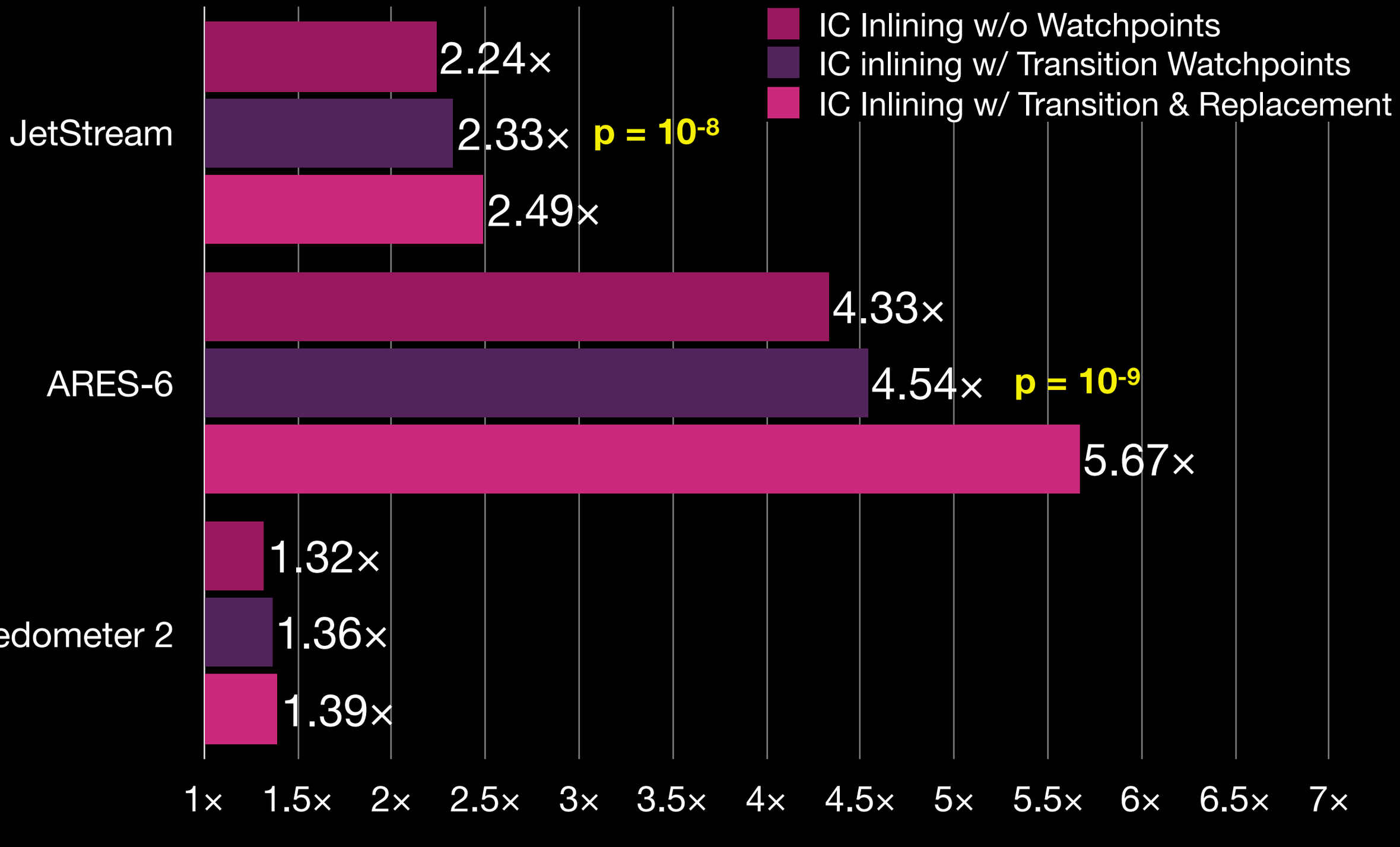
IC Inlining and Watchpoints



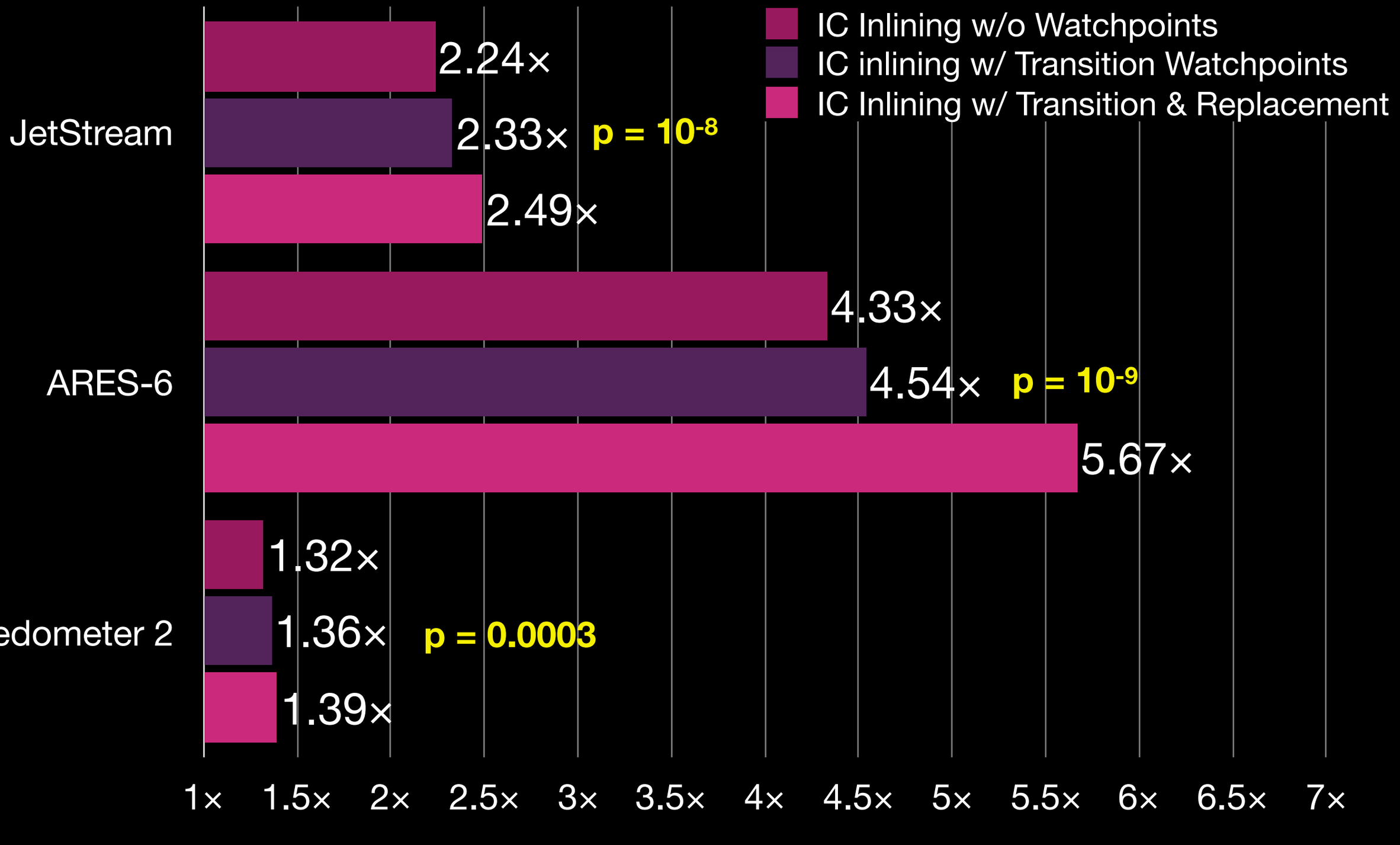
IC Inlining and Watchpoints



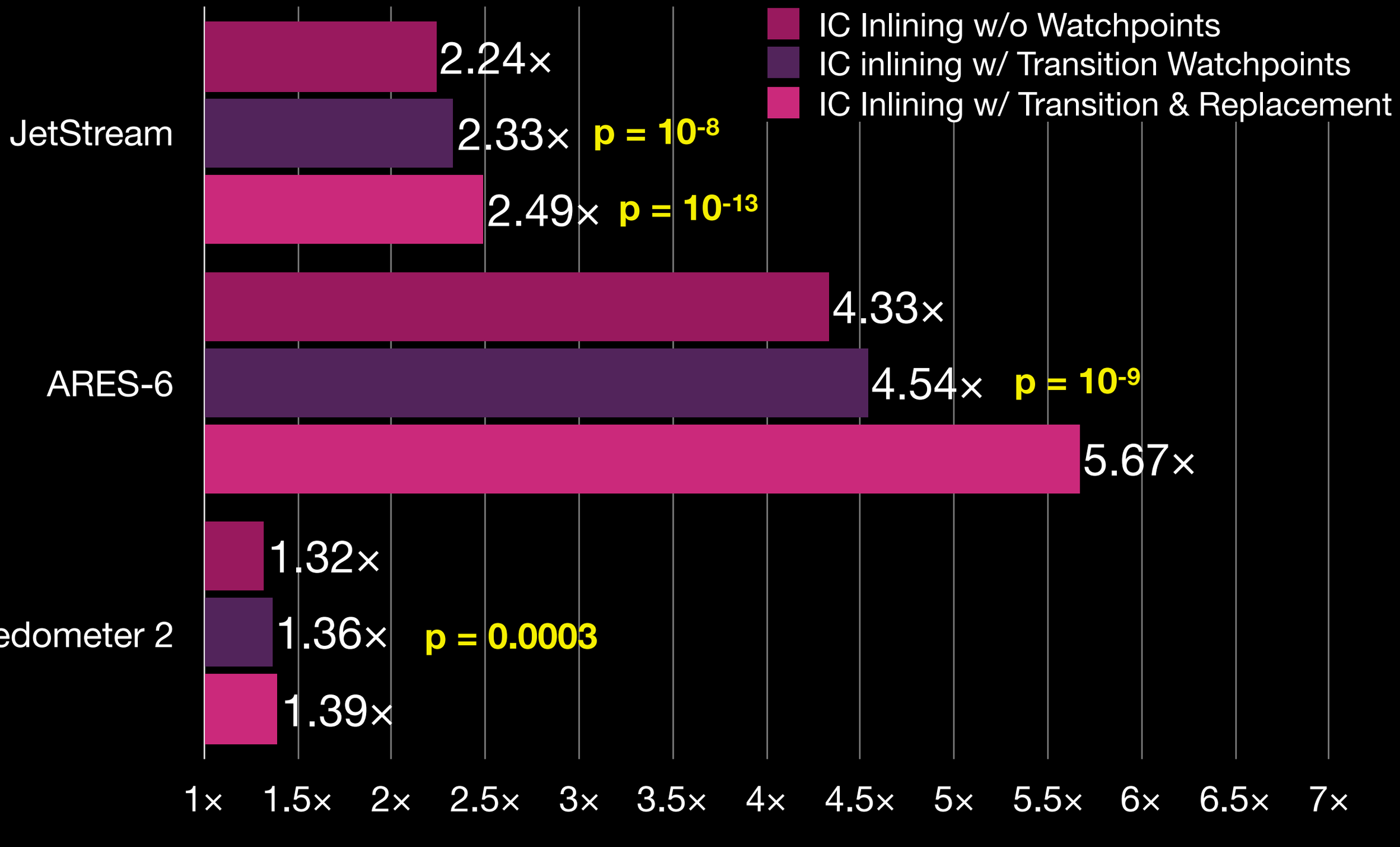
IC Inlining and Watchpoints



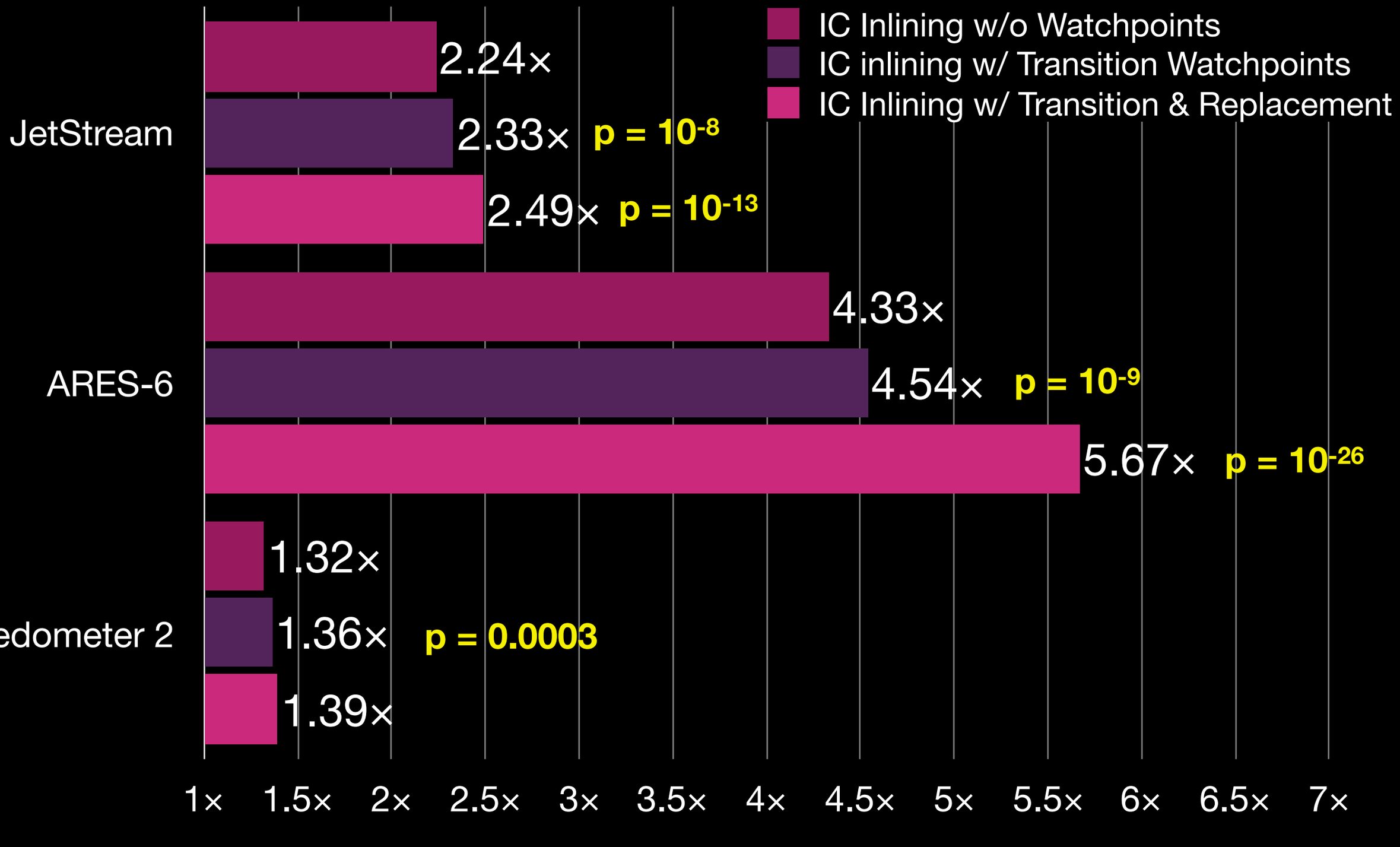
IC Inlining and Watchpoints



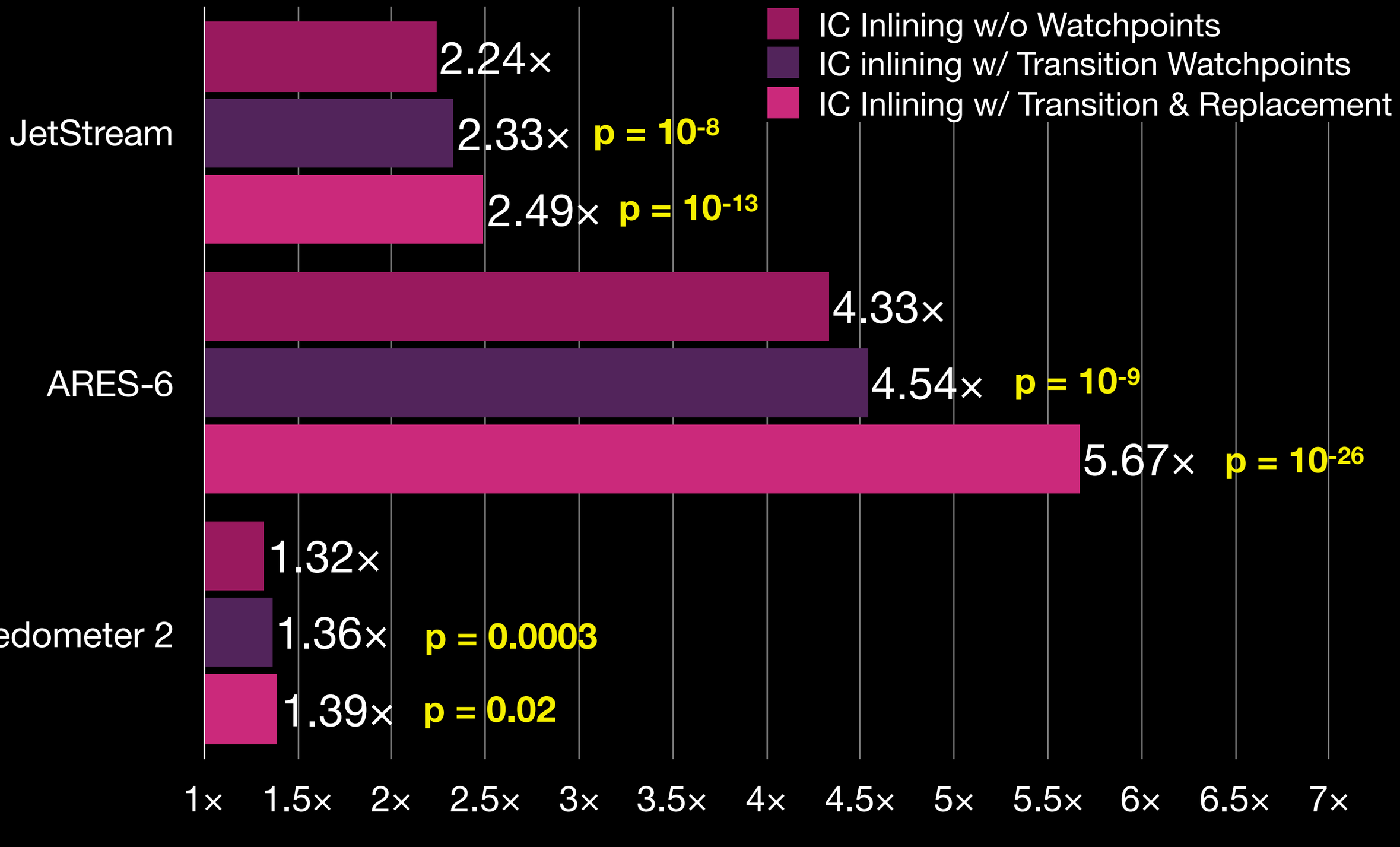
IC Inlining and Watchpoints



IC Inlining and Watchpoints



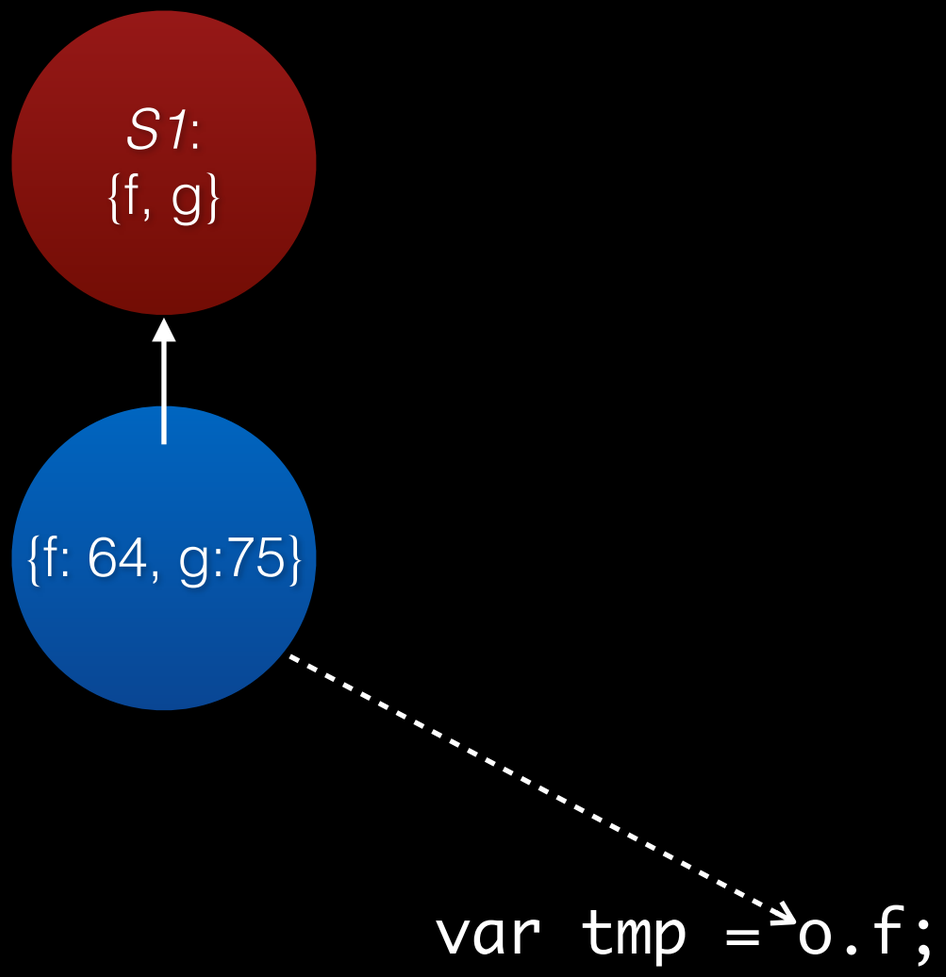
IC Inlining and Watchpoints



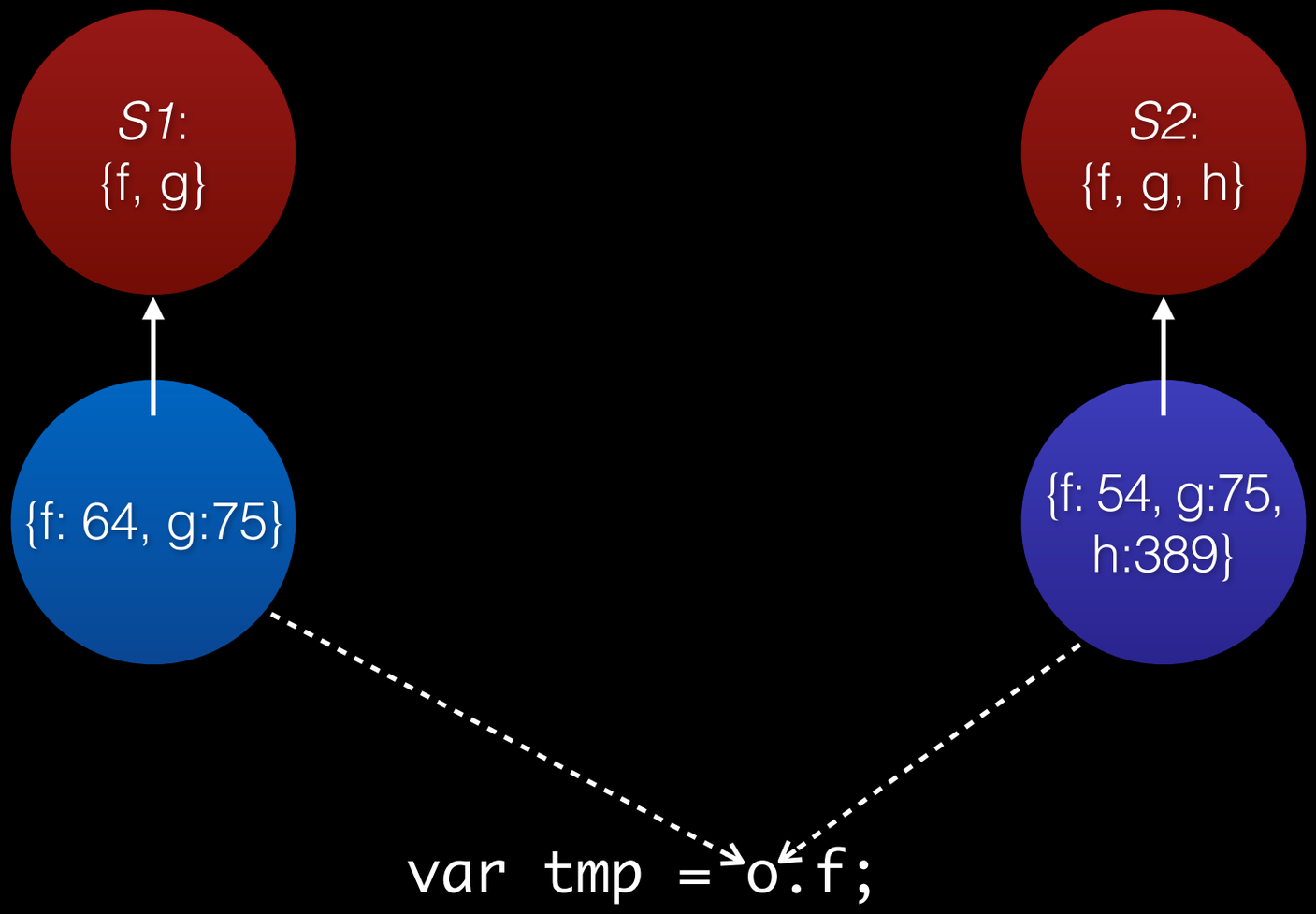
Minimorphic IC Inlining

```
var tmp = o.f;
```

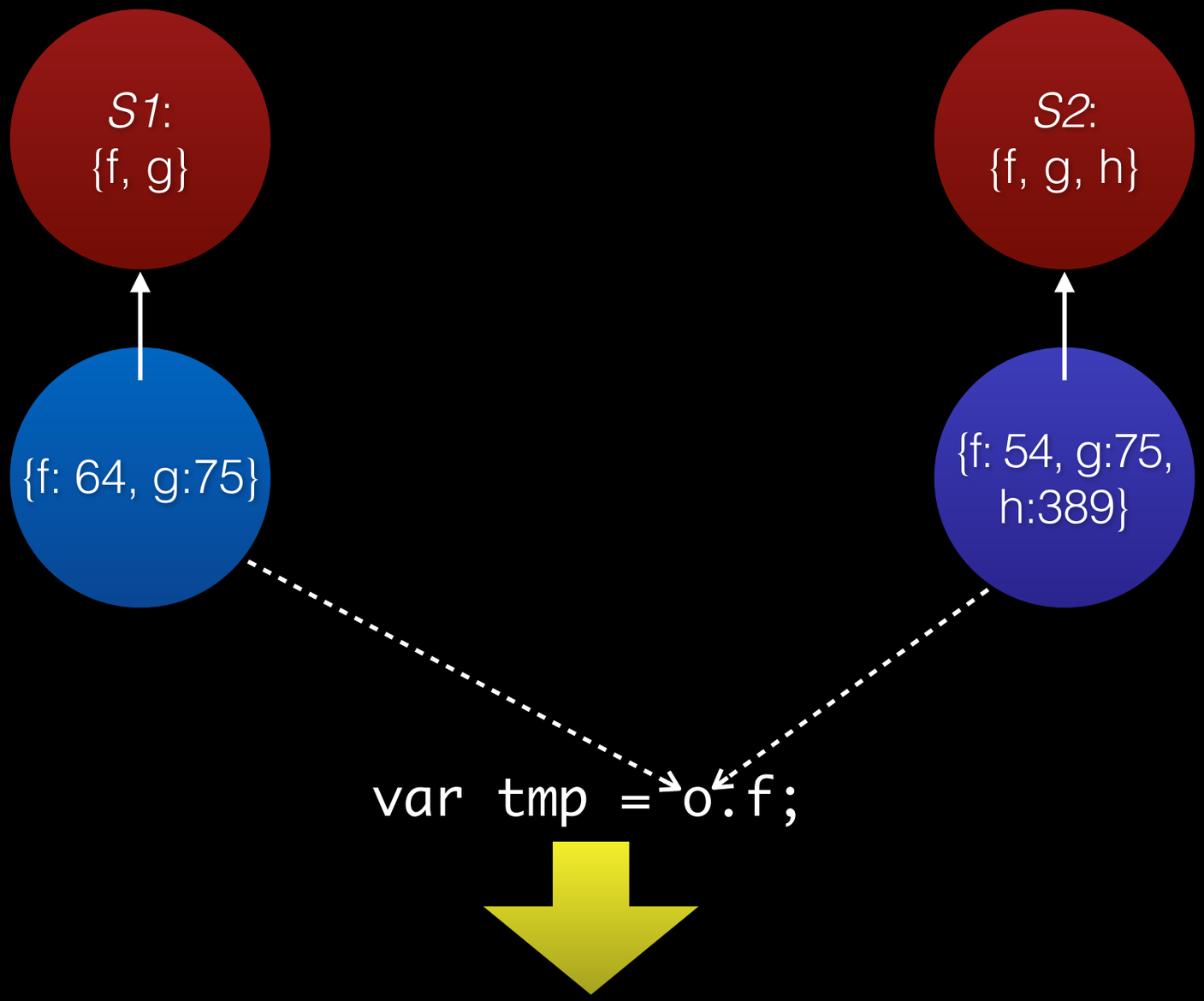
Minimorphic IC Inlining



Minimorphic IC Inlining



Minimorphic IC Inlining

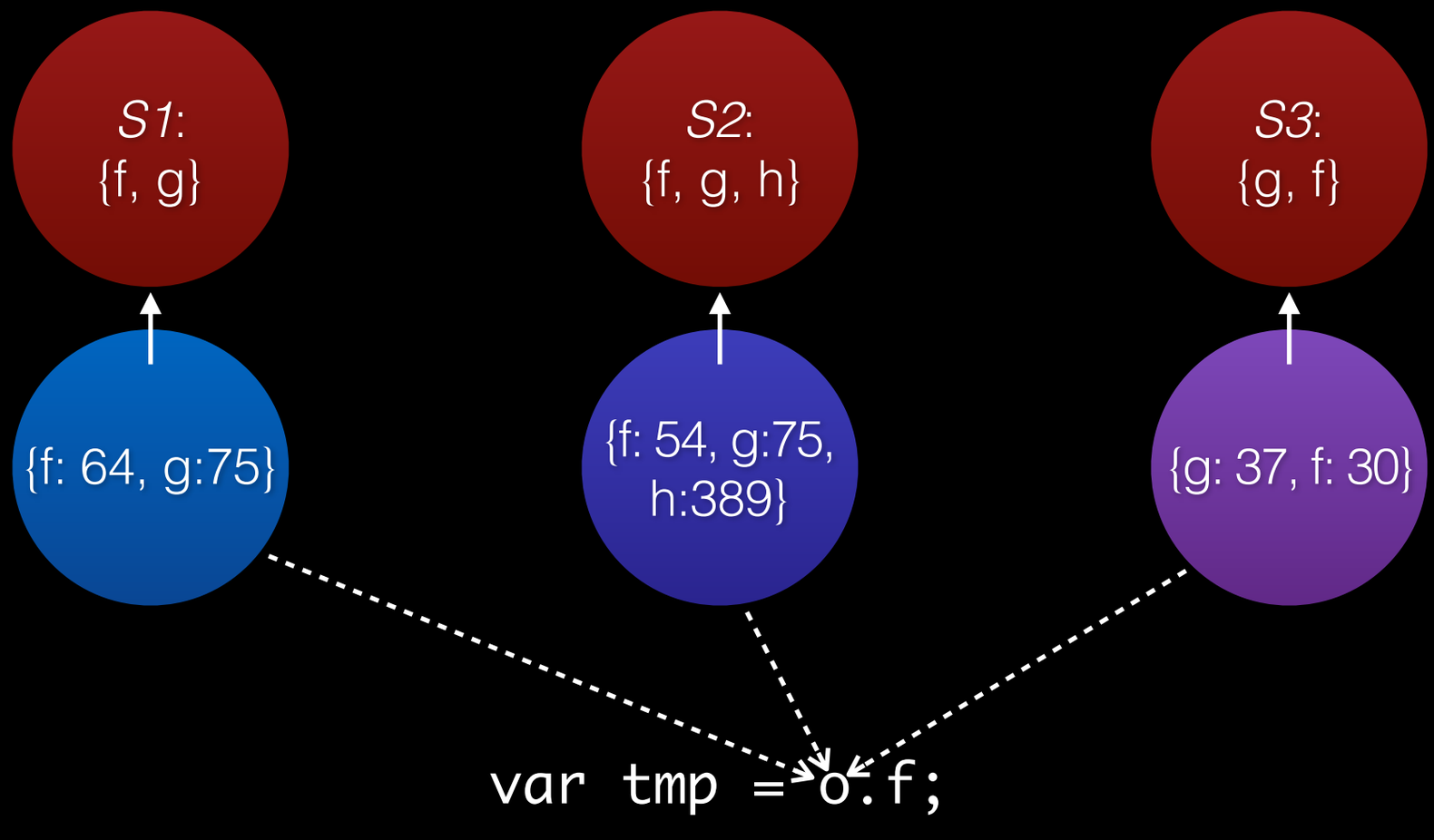



```
CheckStructure(@o, [S1, S2])
GetByOffset(@o, “f”, 0)
```

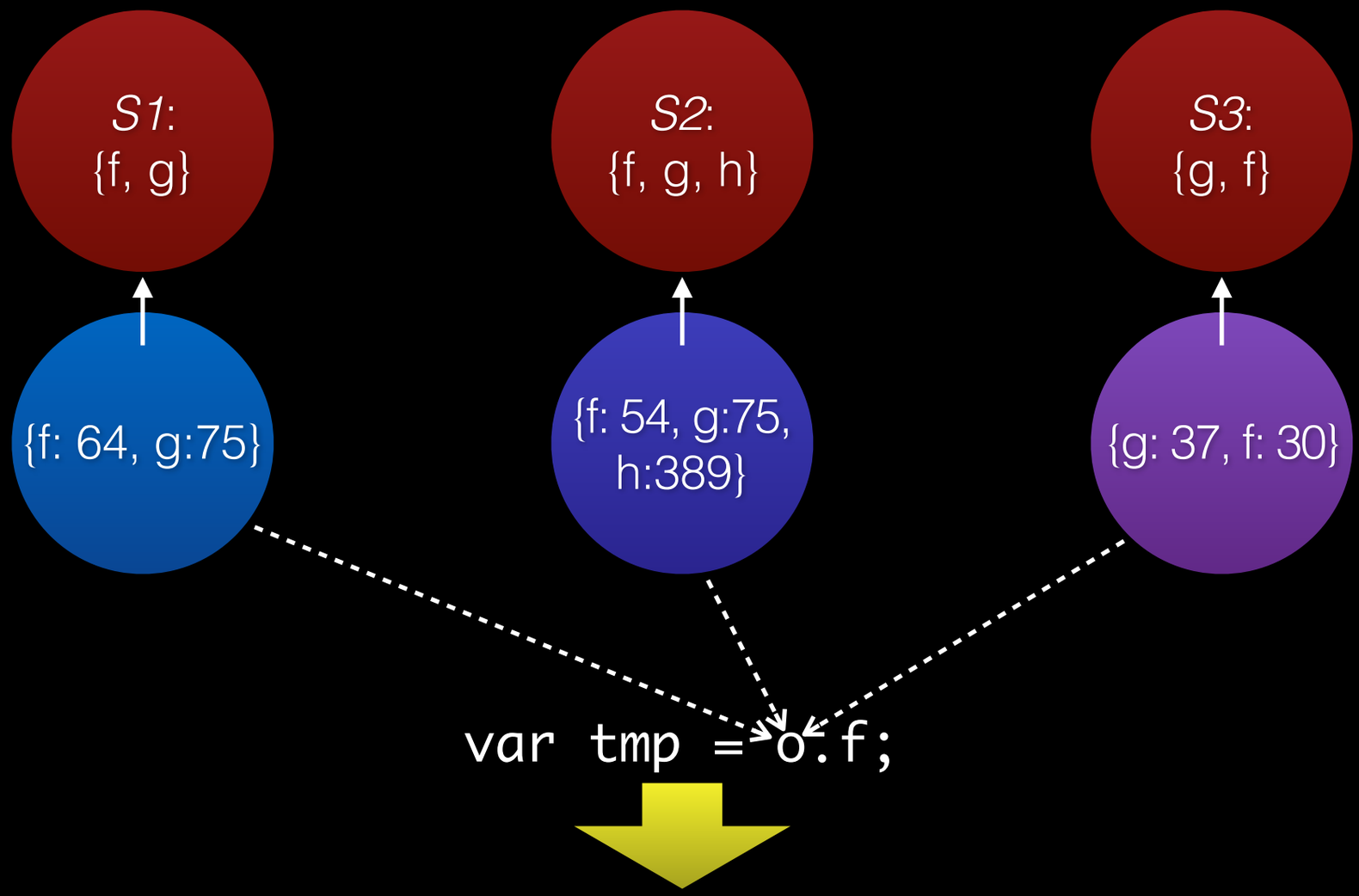
Polymorphic IC Inlining

```
var tmp = o.f;
```

Polymorphic IC Inlining

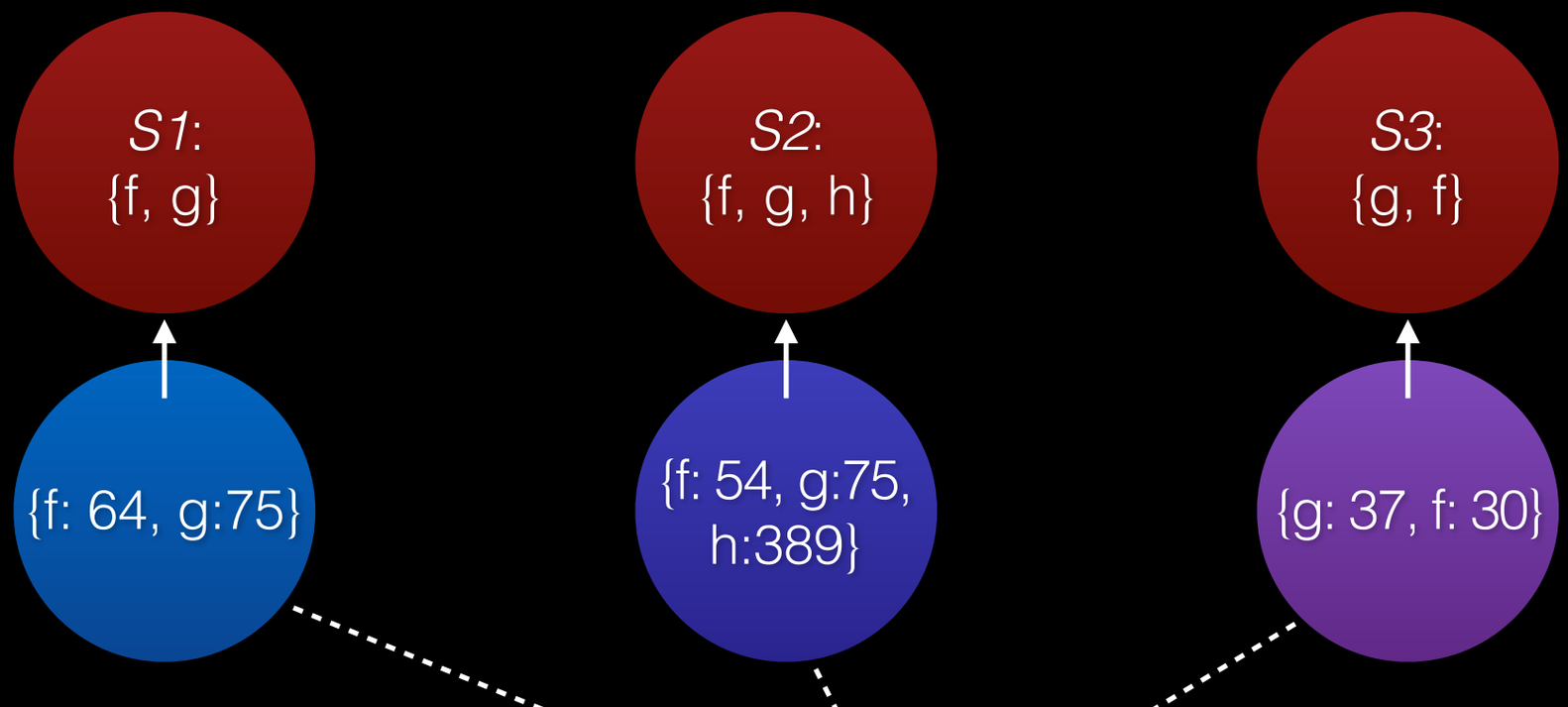


Polymorphic IC Inlining



DFG IR: `MultiGetByOffset(@o, “f”, [S1, S2] => 0, [S3] => 1)`

Polymorphic IC Inlining



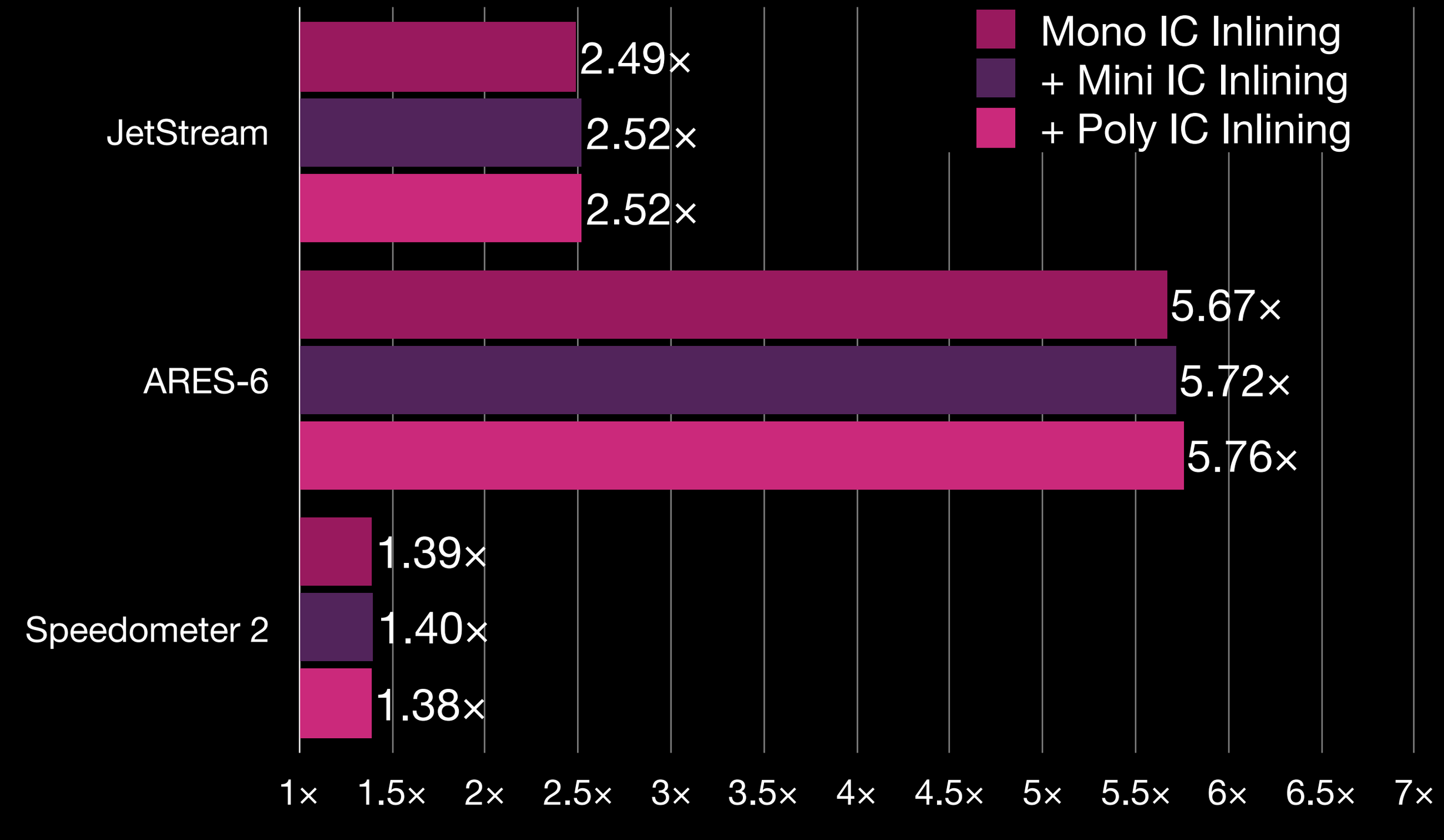

```
var tmp = o.f;
```

DFG IR: `MultiGetByOffset(@o, “f”, [S1, S2] => 0, [S3] => 1)`

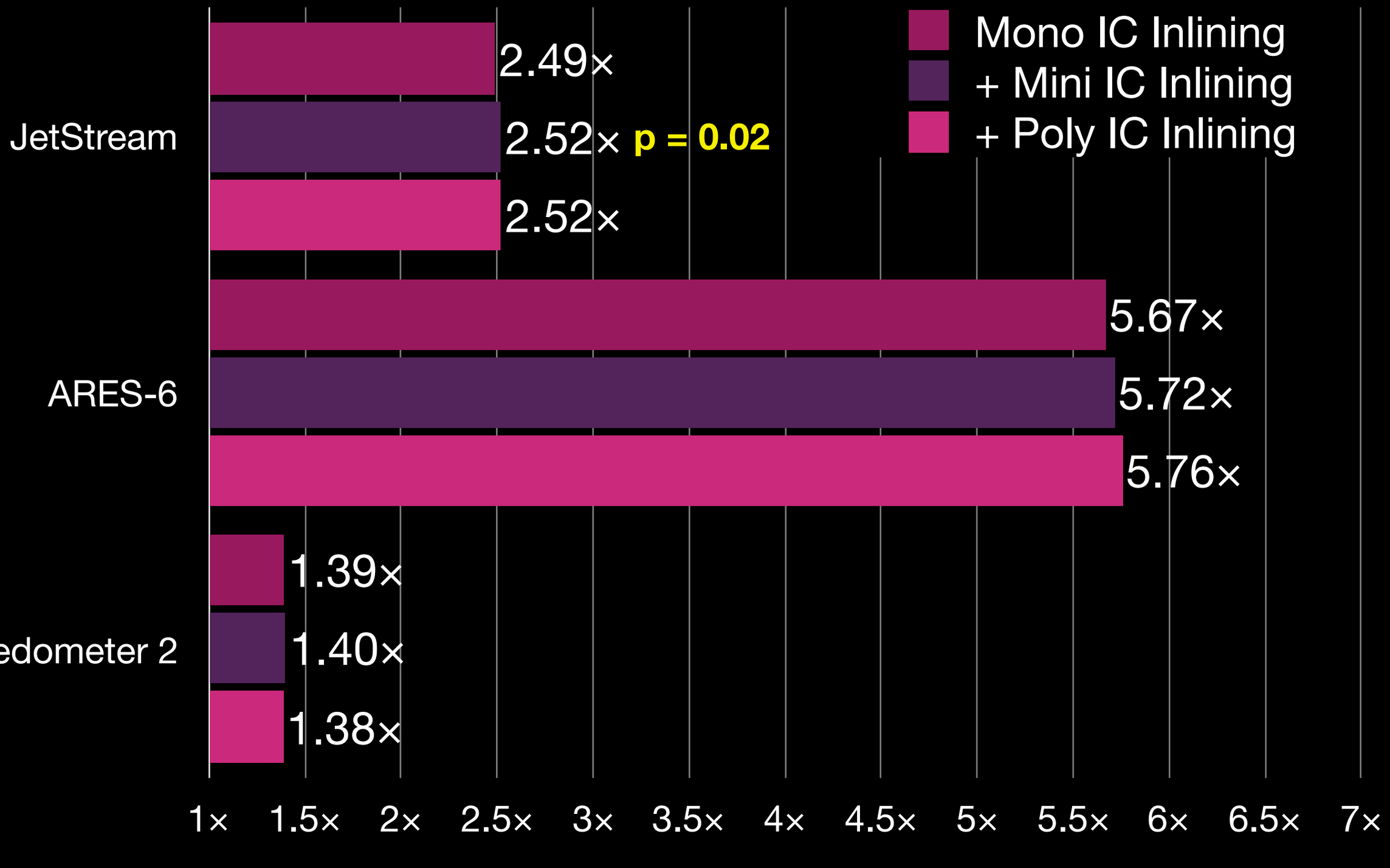
B3 IR:

```
if (o->structureID == S1 || o->structureID == S2)
    result = o->inlineStorage[0]
else
    result = o->inlineStorage[1]
```

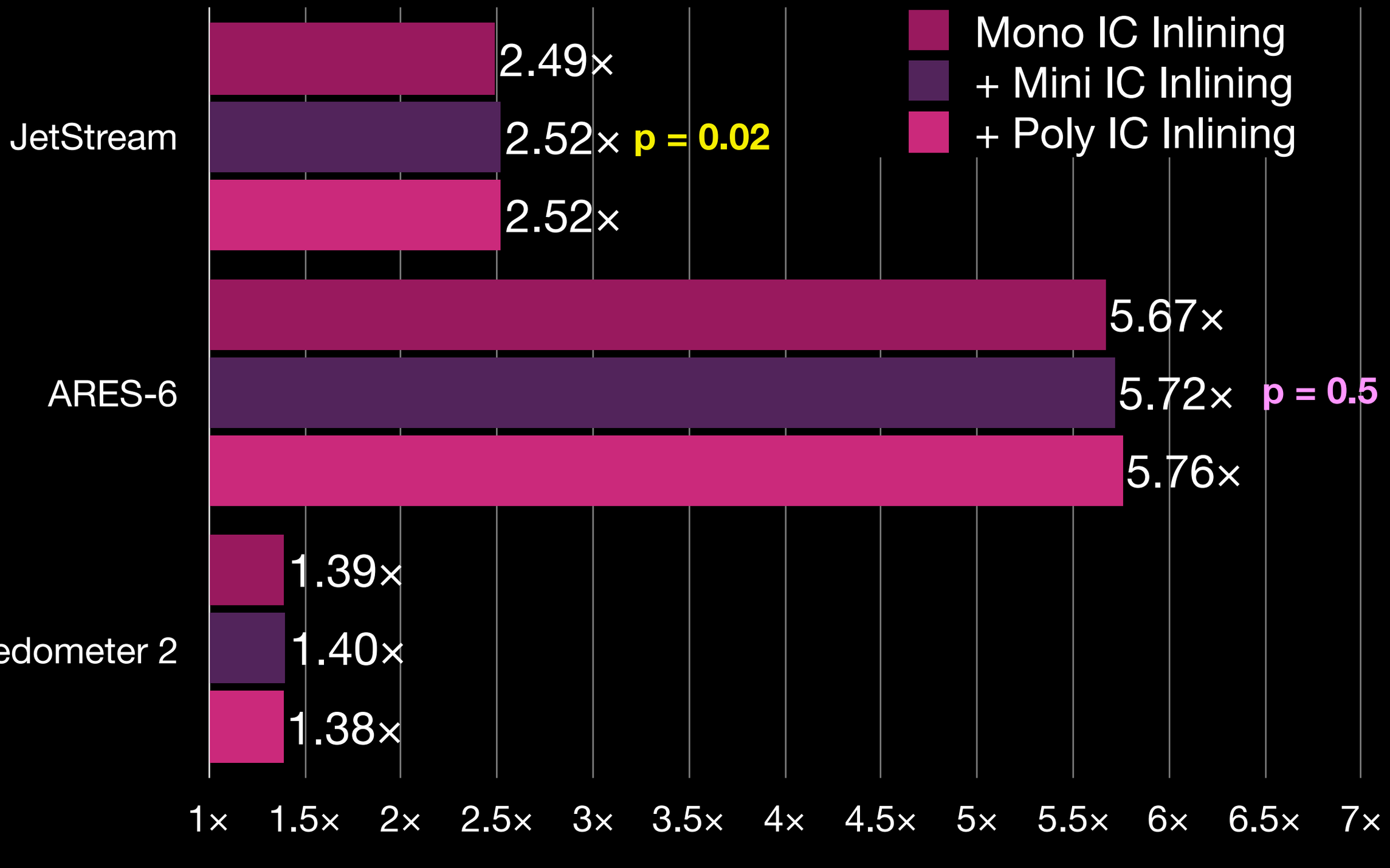
IC Polymorphic Inlining Speed-up



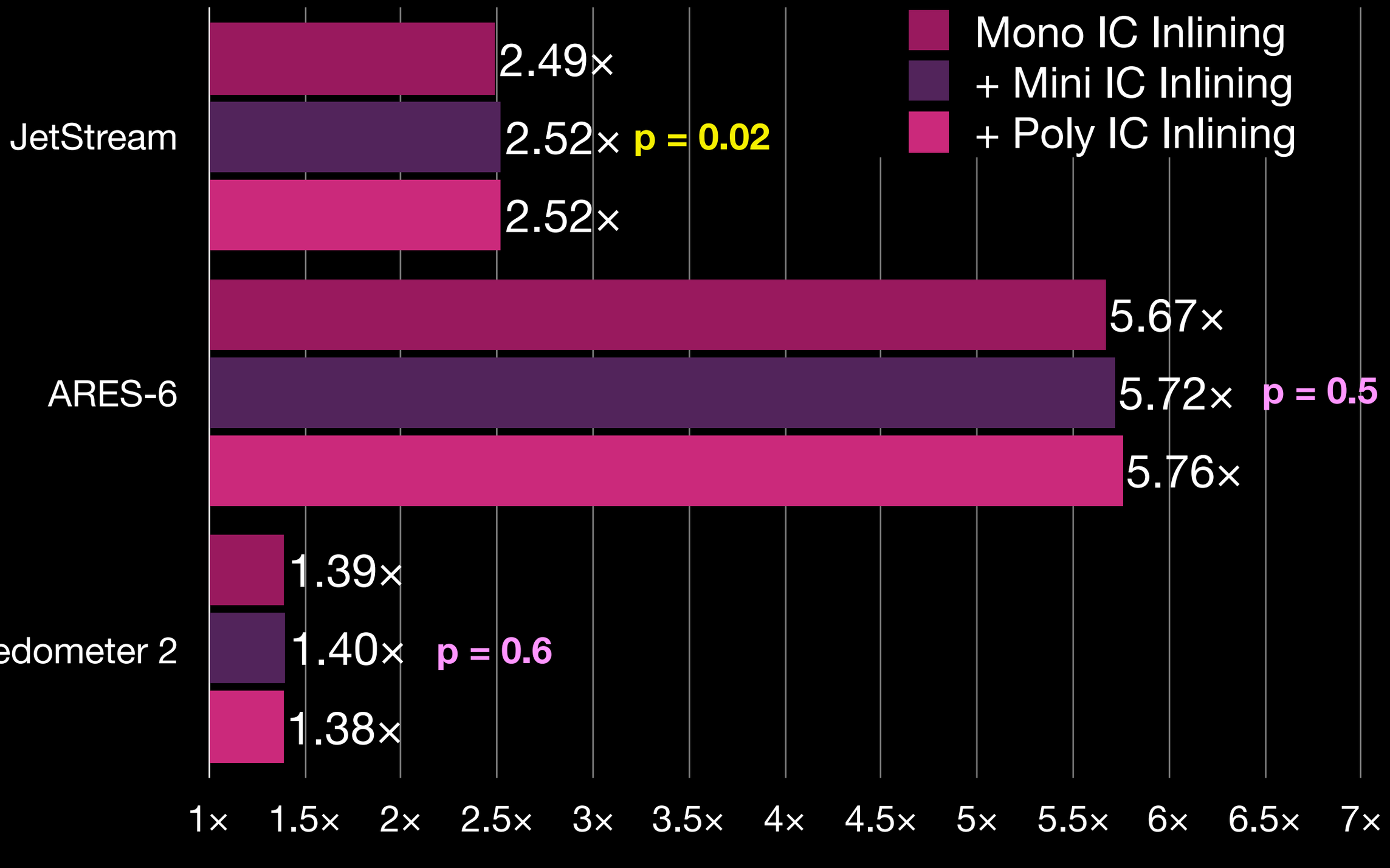
IC Polymorphic Inlining Speed-up



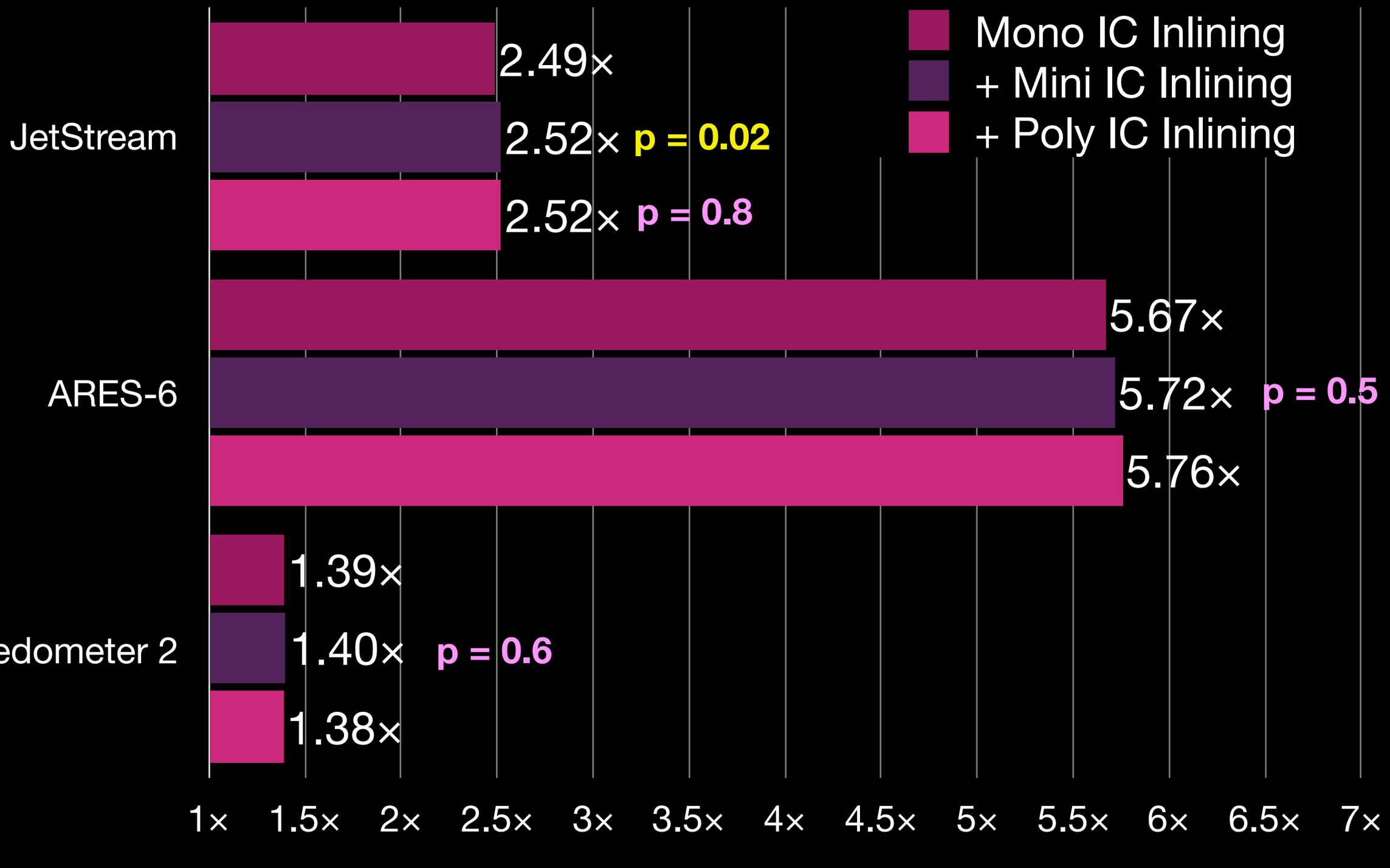
IC Polymorphic Inlining Speed-up



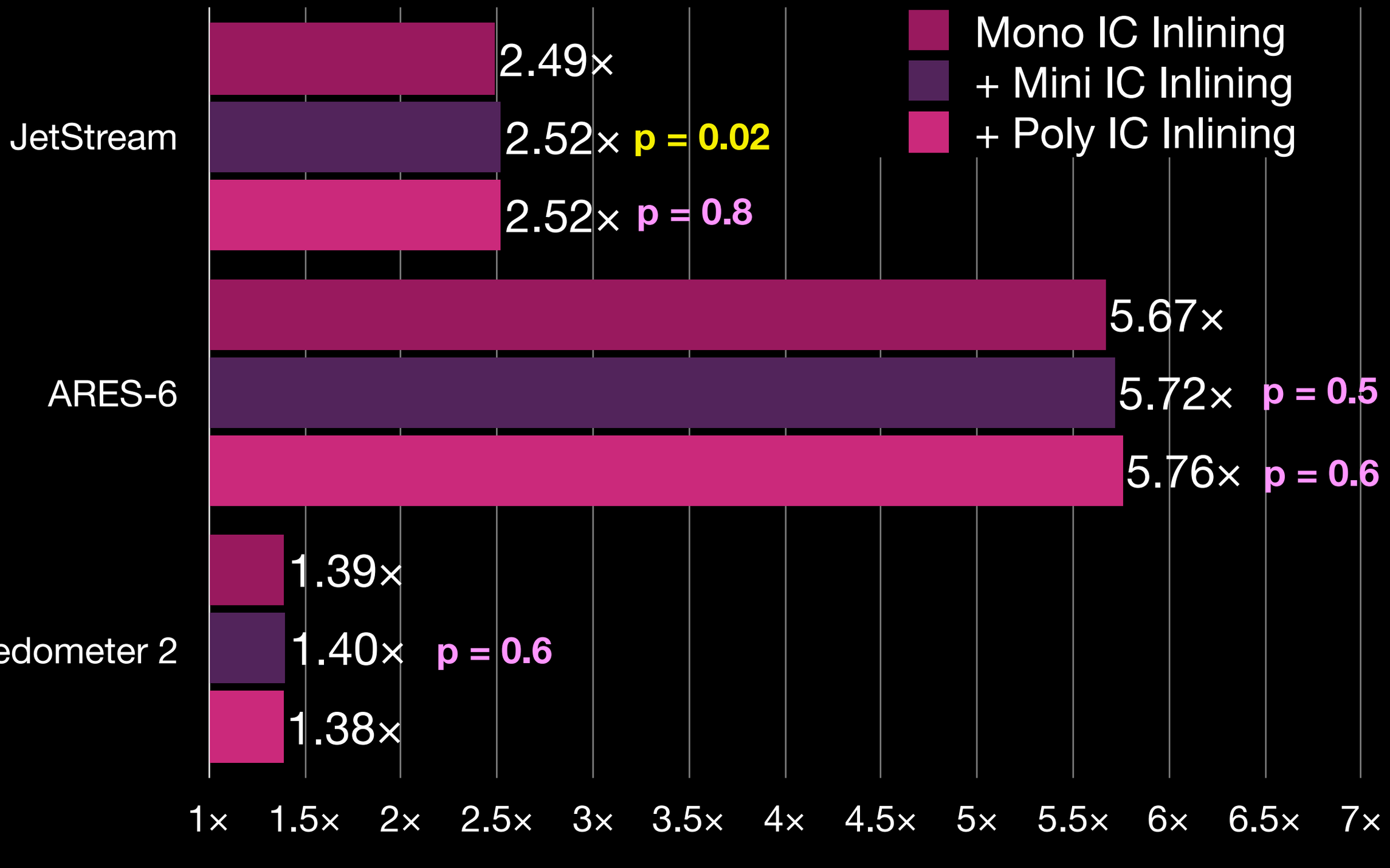
IC Polymorphic Inlining Speed-up



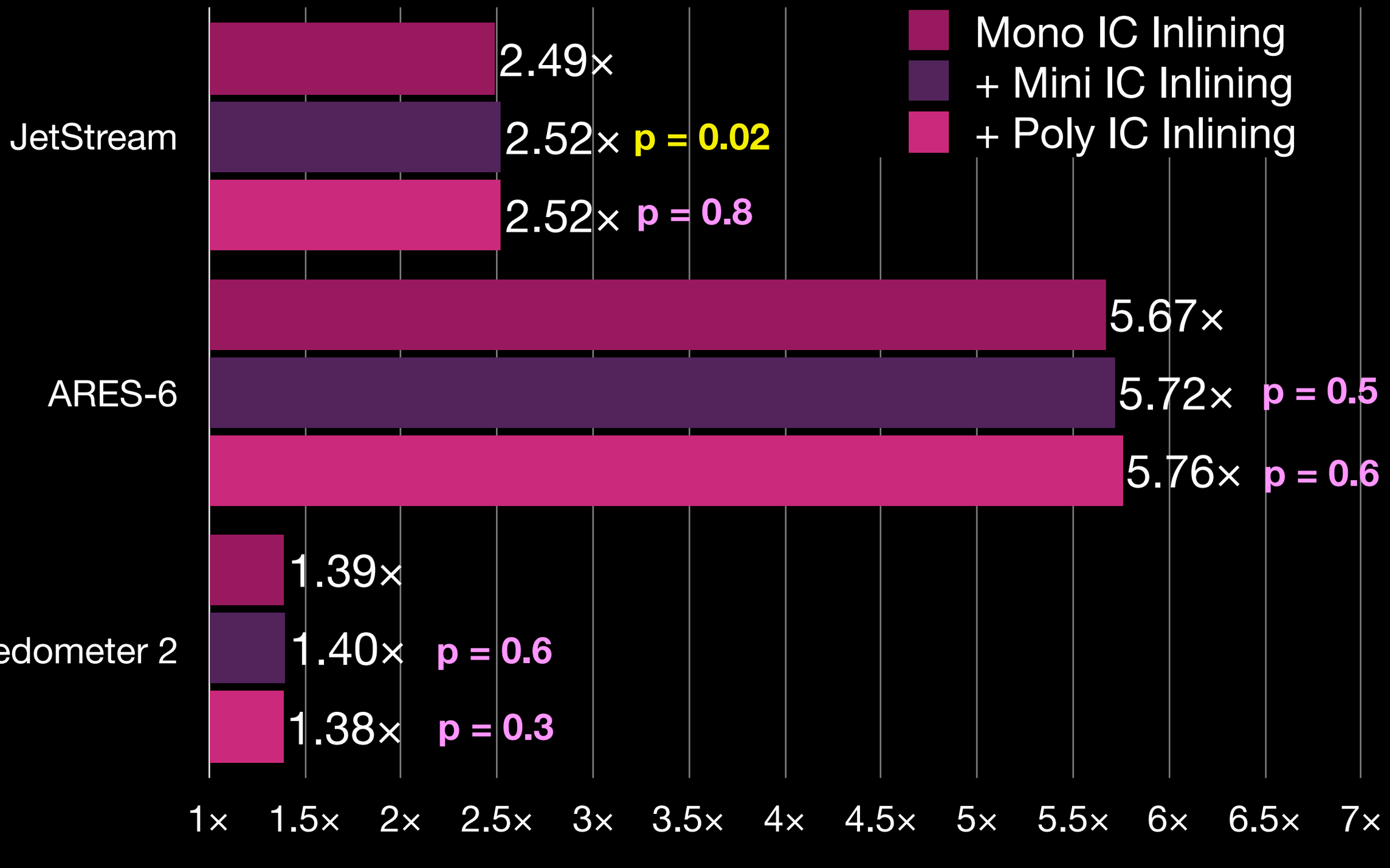
IC Polymorphic Inlining Speed-up



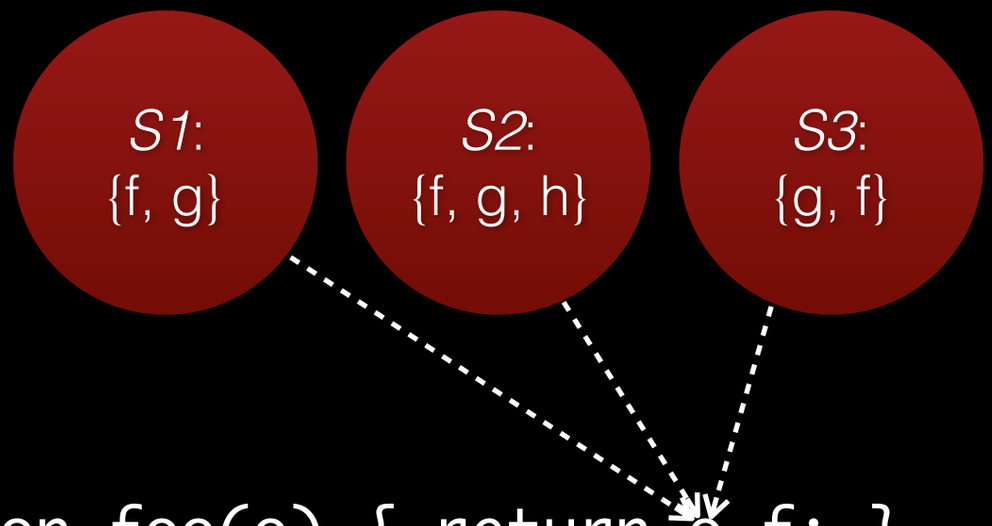
IC Polymorphic Inlining Speed-up

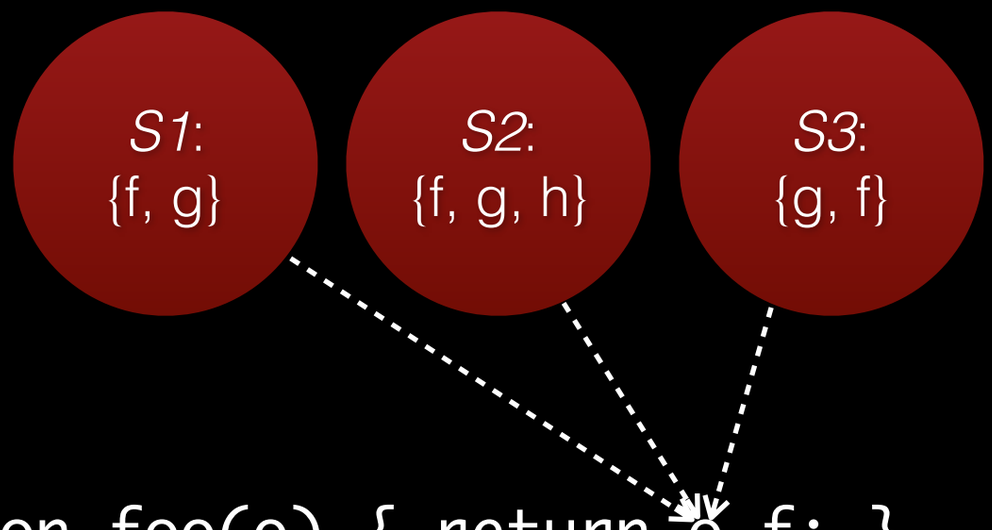


IC Polymorphic Inlining Speed-up



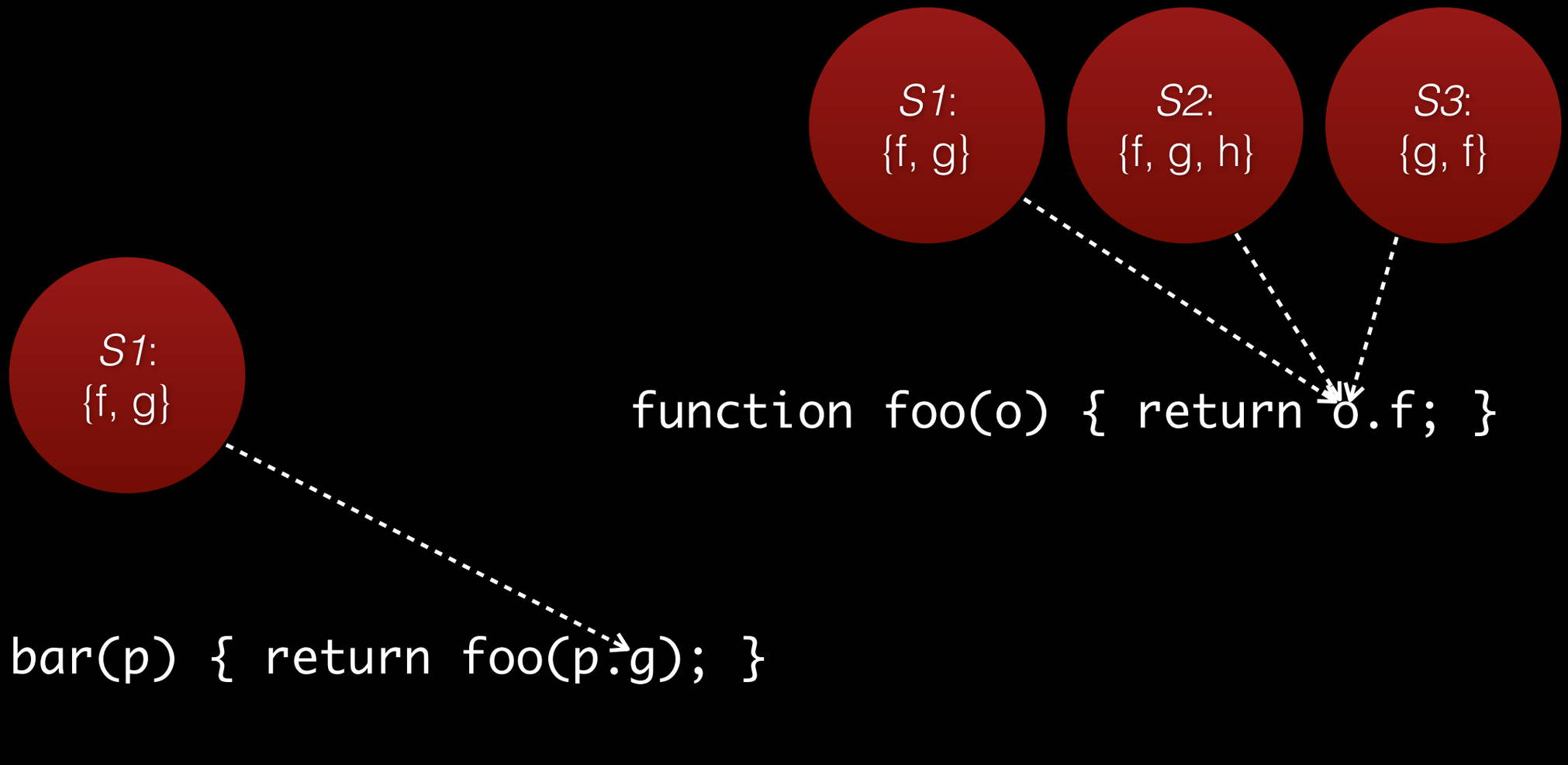
```
function foo(o) { return o.f; }
```


S1:
{f, g}
S2:
{f, g, h}
S3:
{g, f}
function foo(o) { return o.f; }




```
function foo(o) { return o.f; }
```

```
function bar(p) { return foo(p.g); }
```


S1:
{f, g}
S2:
{f, g, h}
S3:
{g, f}
S1:
{f, g}
function foo(o) { return o.f; }
function bar(p) { return foo(p.g); }

S1:
{f, g}

S2:
{f, g, h}

S3:
{g, f}

S1:
{f, g}

```
function foo(o) { return o.f; }
```

```
function bar(p) { return foo(p.g); }
```

DFG
(less optimizing JIT)

S1:
{f, g}

S2:
{f, g, h}

S3:
{g, f}

S1:
{f, g}

```
function foo(o) { return o.f; }
```

```
function bar(p) { return foo(p.g); }
```

DFG
(less optimizing JIT)

```
…
-> foo
```

```
jmp Lslow
```

```
<- foo
```

S1:
{f, g}

S2:
{f, g, h}

S3:
{g, f}

S1:
{f, g}

```
function foo(o) { return o.f; }
```

```
function bar(p) { return foo(p.g); }
```

DFG

(less optimizing JIT)

```
…
-> foo
```

```
cmp S1,
    (%rax)
jnz Lslow
mov 10(%rax),
    %rax
```

```
<- foo
```

S1:
{f, g}

S2:
{f, g, h}

S3:
{g, f}

```
function foo(o) { return o.f; }
```

S1:
{f, g}

```
function bar(p) { return foo(p.g); }
```

DFG
(less optimizing JIT)

```
…
-> foo
```

```
cmp S1,
    (%rax)
jnz Lslow
mov 10(%rax),
    %rax
```

```
<- foo
```

FTL
(optimizing JIT)

S1:
{f, g}

S2:
{f, g, h}

S3:
{g, f}

S1:
{f, g}

```
function foo(o) { return o.f; }
```

```
function bar(p) { return foo(p.g); }
```

DFG
(less optimizing JIT)

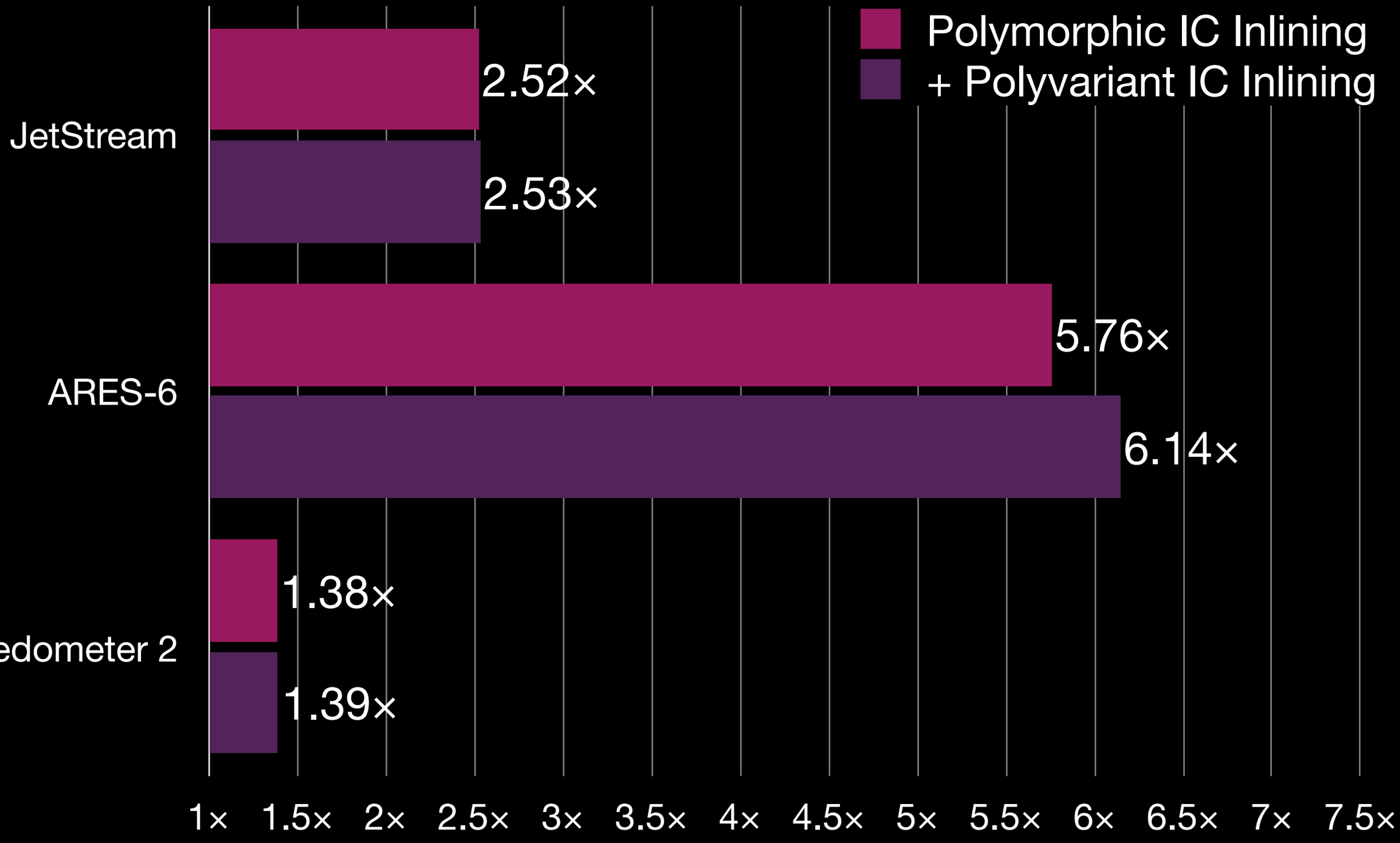
```
…
-> foo
<- foo
```

```
cmp S1,
    (%rax)
jnz Lslow
mov 10(%rax),
    %rax
```

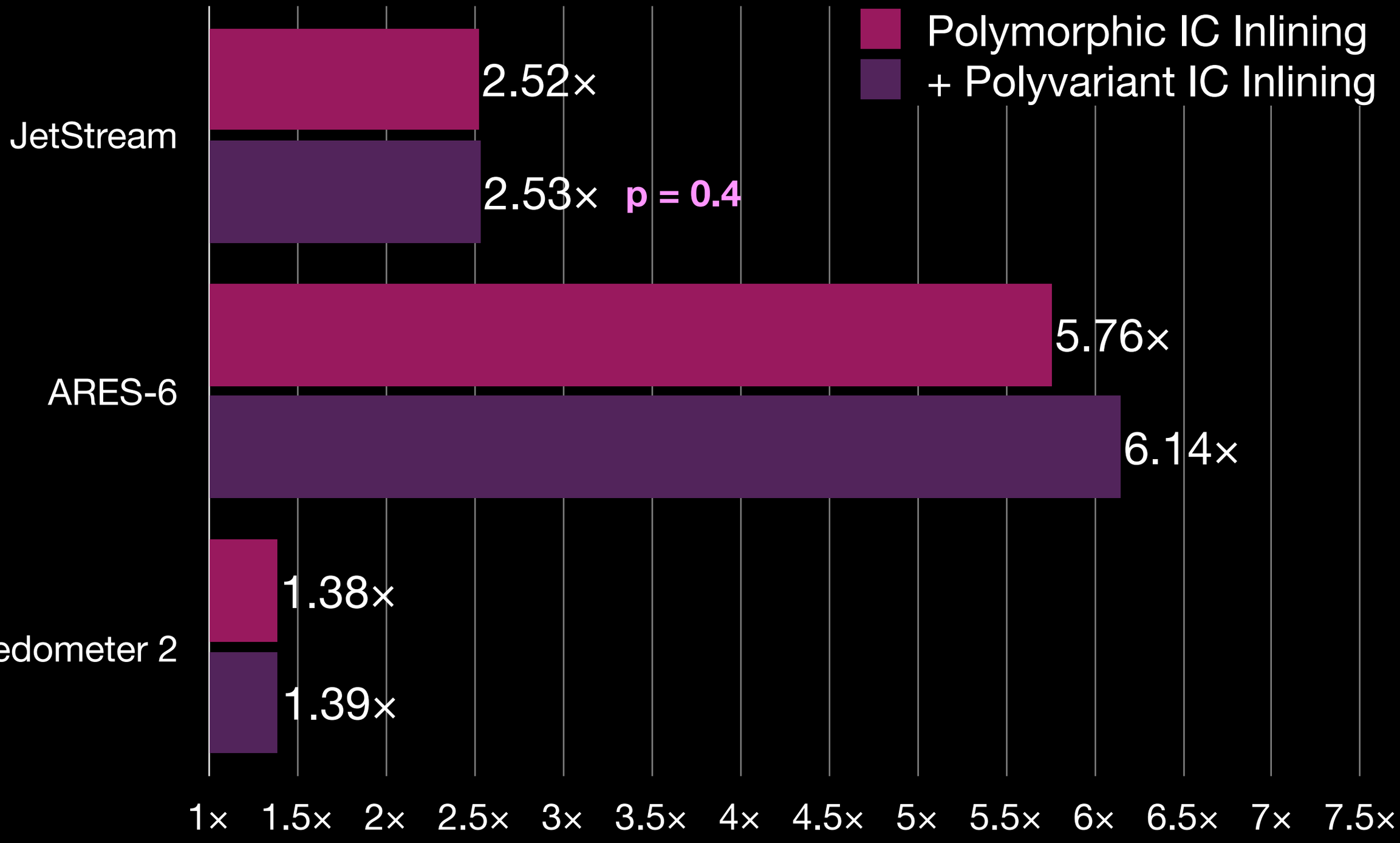
FTL
(optimizing JIT)

```
…
-> foo
    CheckStructure
    GetByOffset
<- foo
```

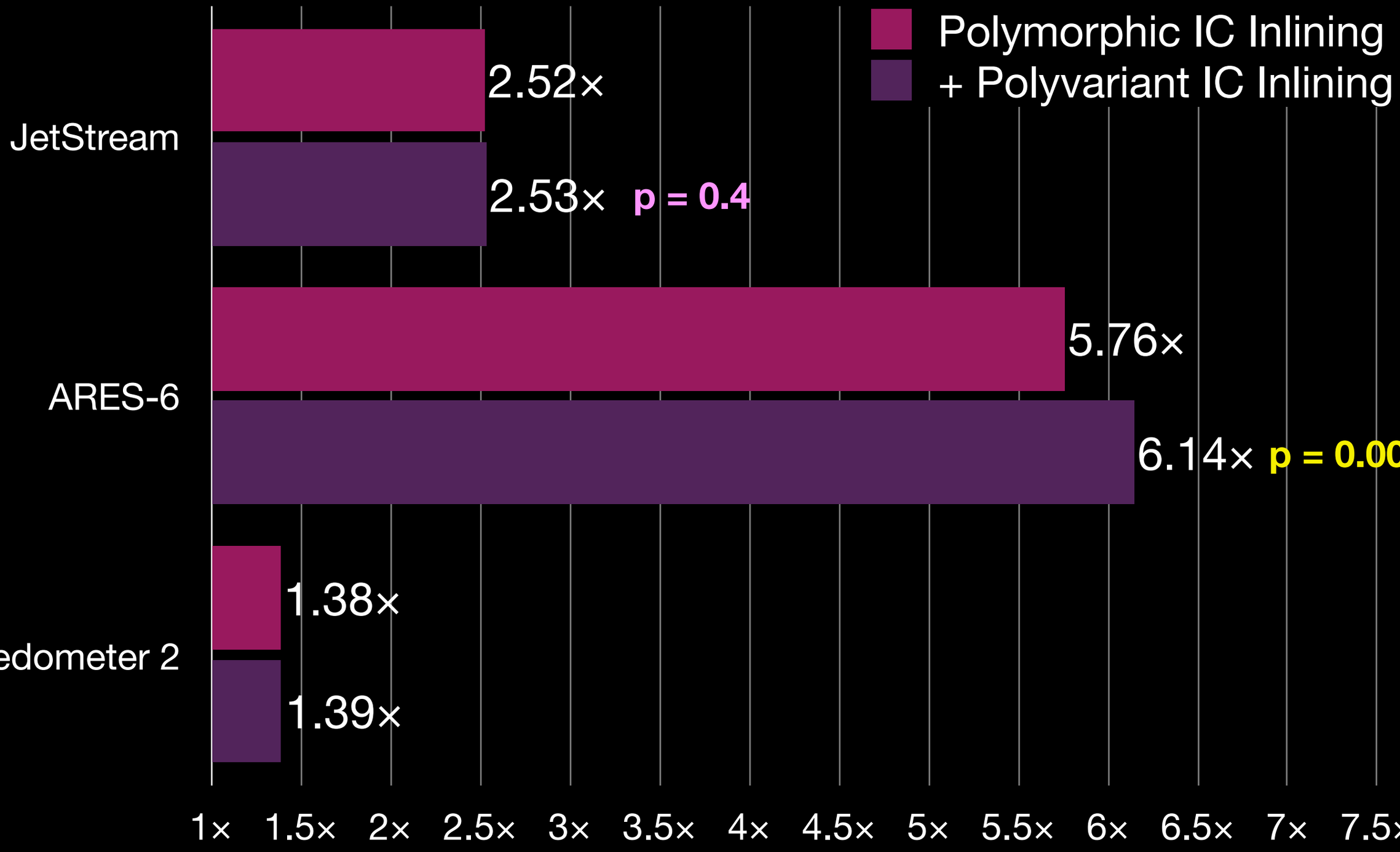
IC Polyvariant Inlining Speed-up



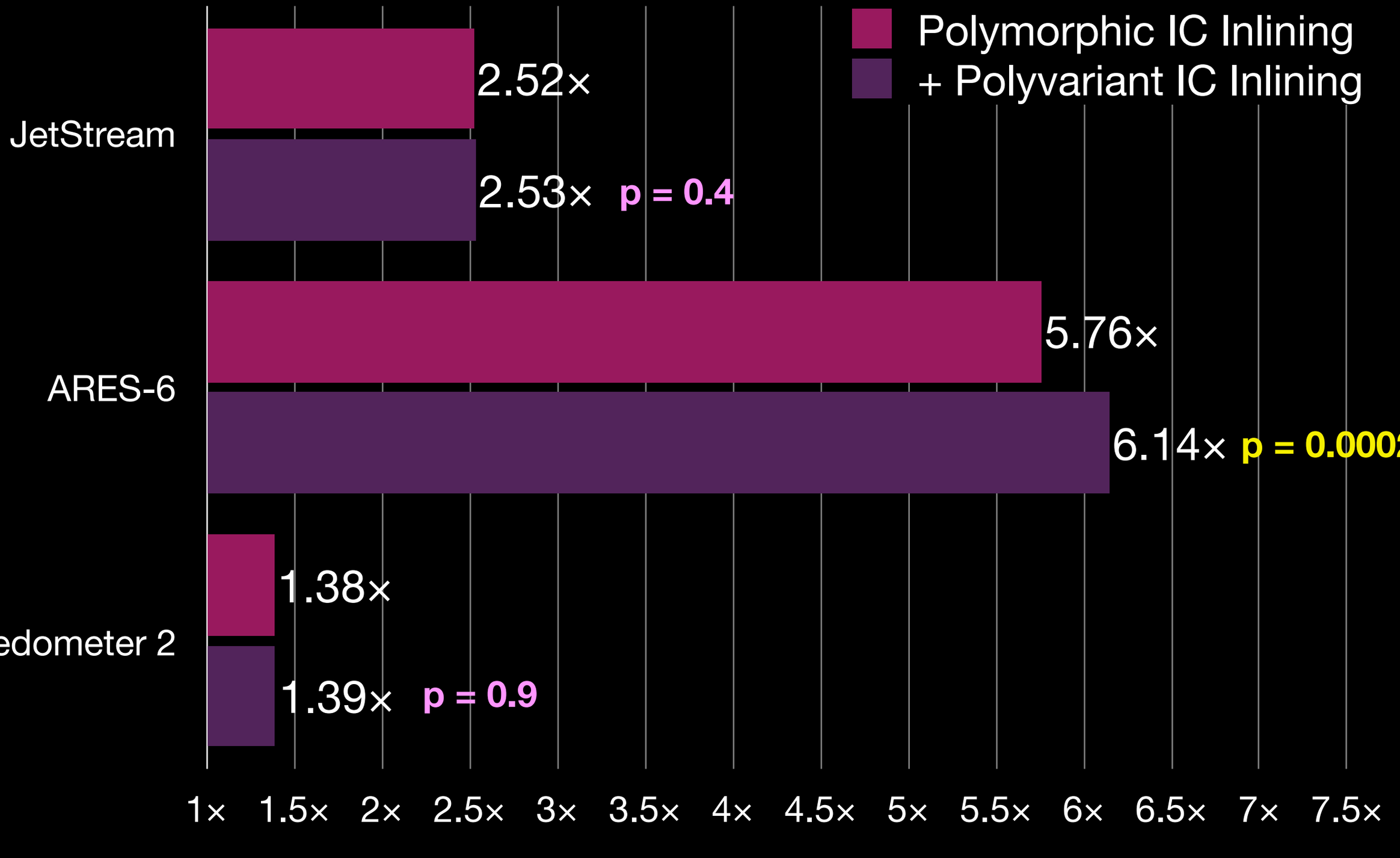
IC Polyvariant Inlining Speed-up



IC Polyvariant Inlining Speed-up



IC Polyvariant Inlining Speed-up



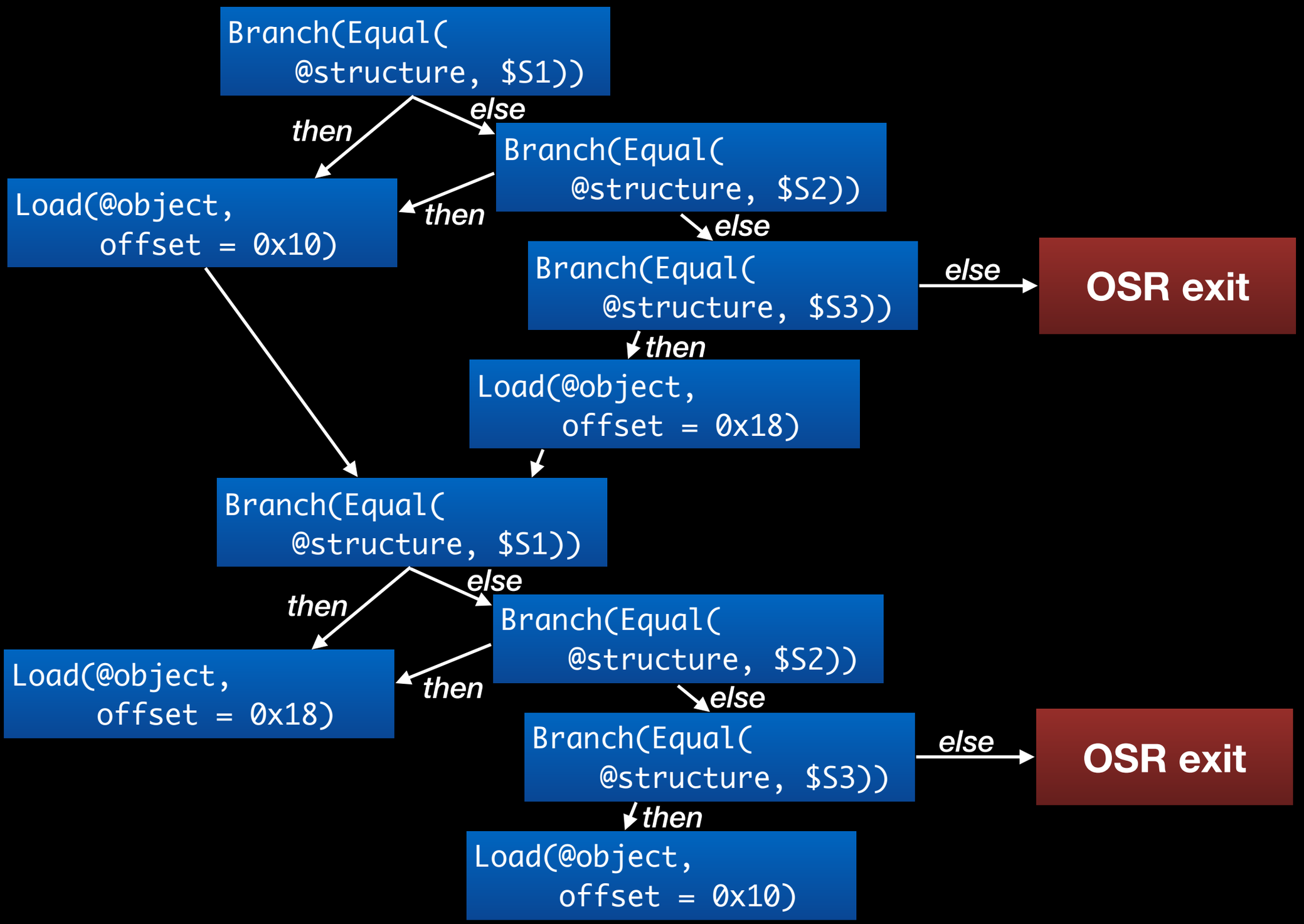
DFG IR:

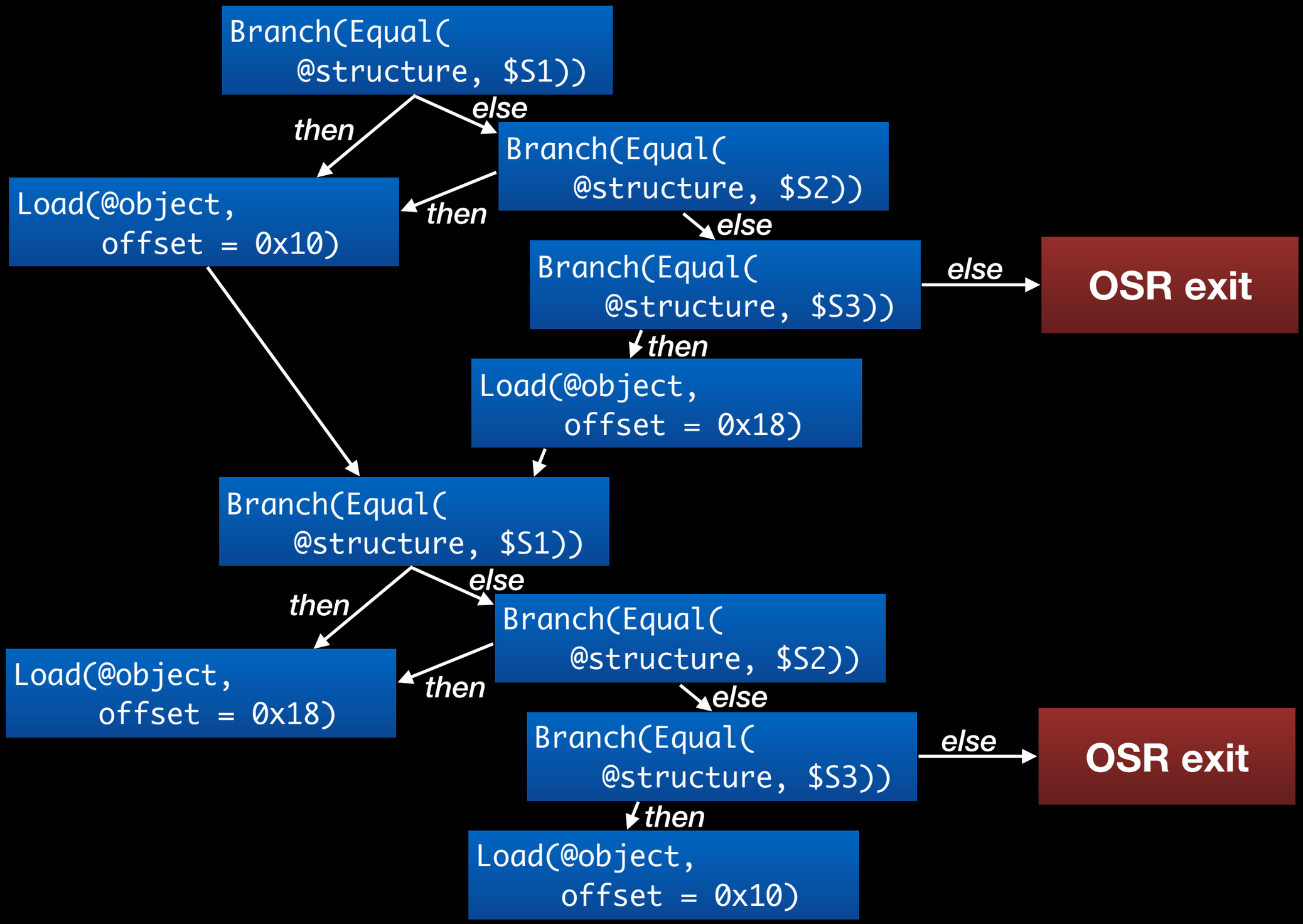
```
MultiGetByOffset(@o, “f”, [S1, S2] => 0, [S3] => 1)
MultiGetByOffset(@o, “g”, [S1, S2] => 1, [S3] => 0)
```

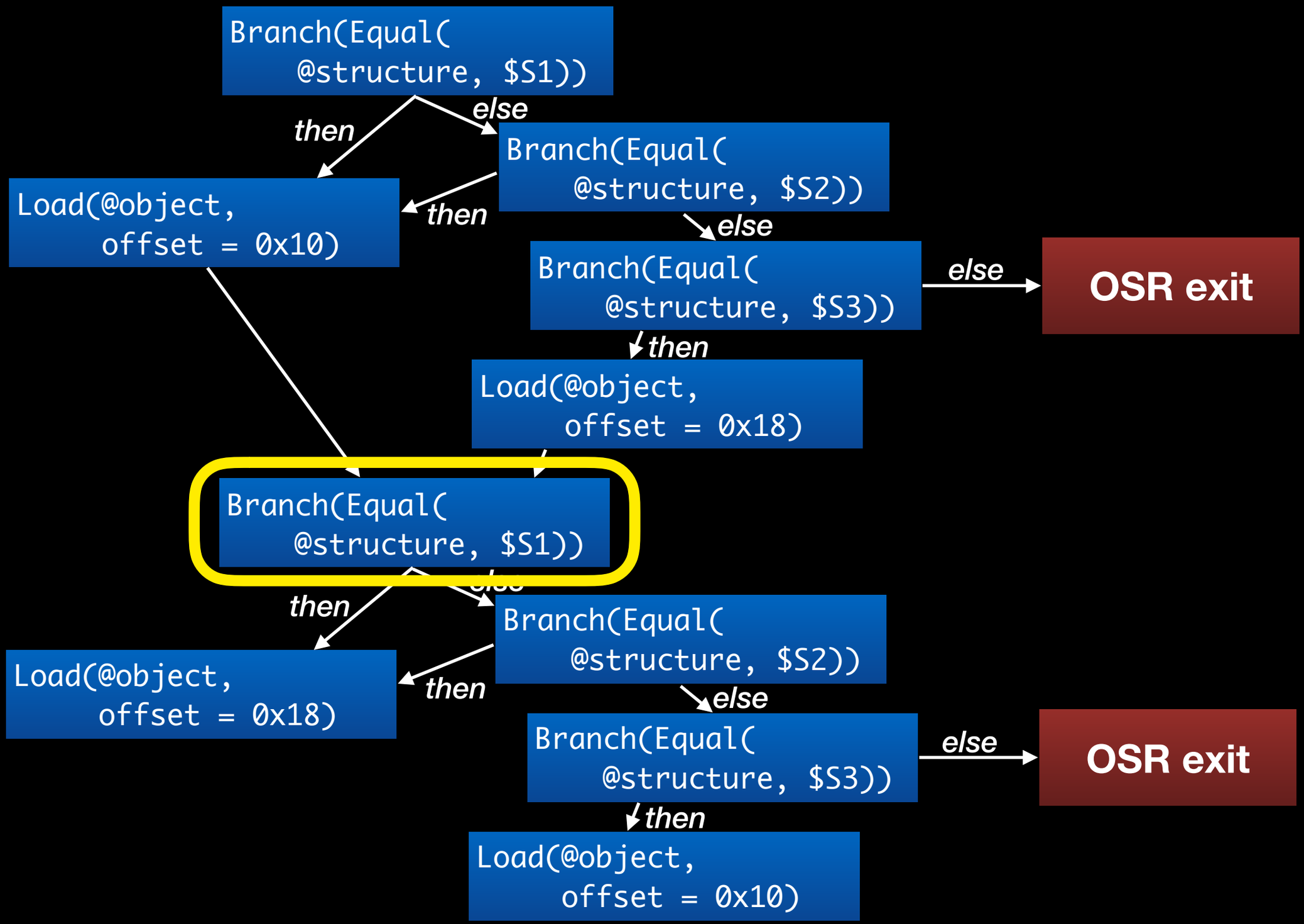
DFG IR:

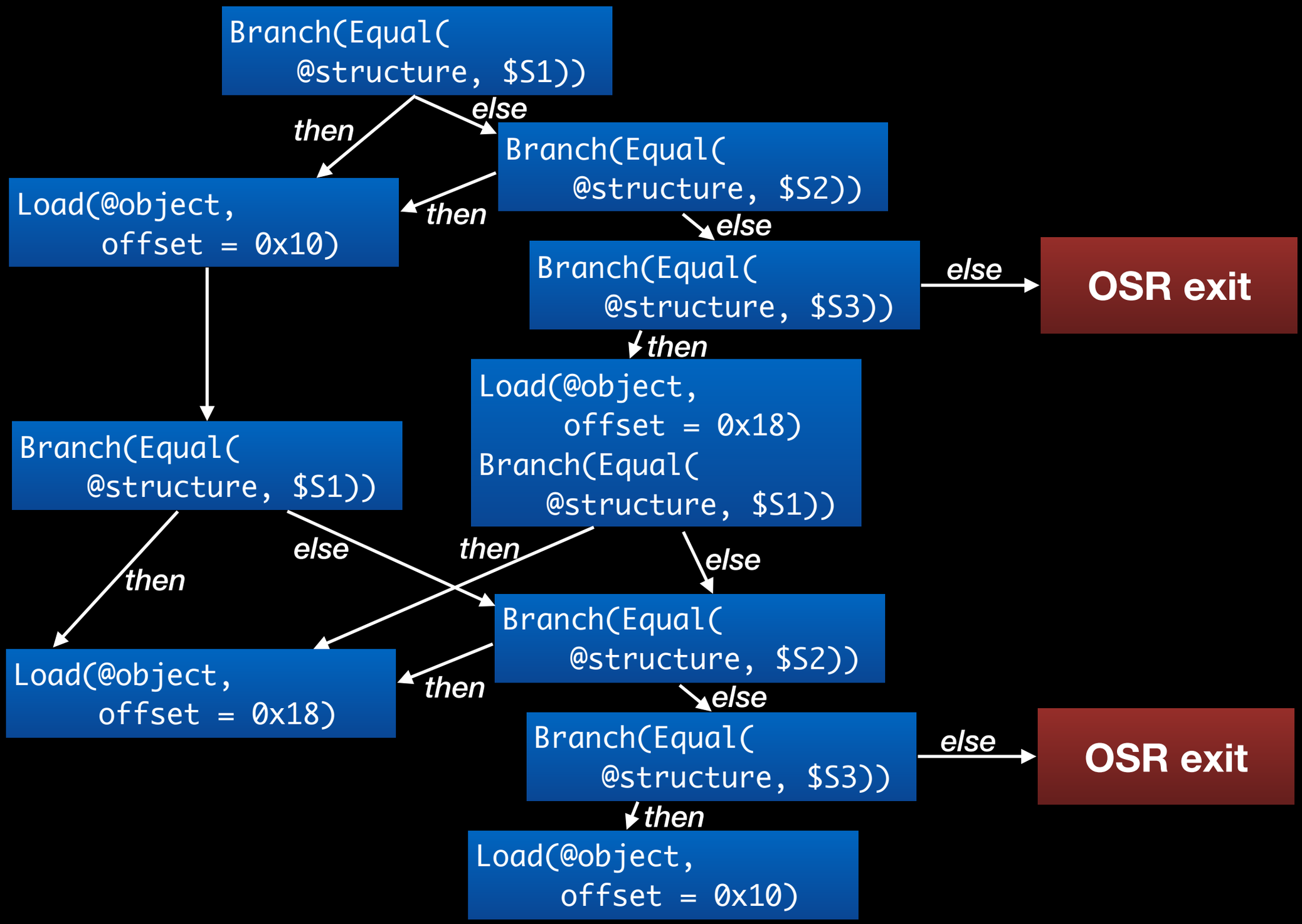
```
MultiGetByOffset(@o, “f”, [S1, S2] => 0, [S3] => 1)
MultiGetByOffset(@o, “g”, [S1, S2] => 1, [S3] => 0)
```

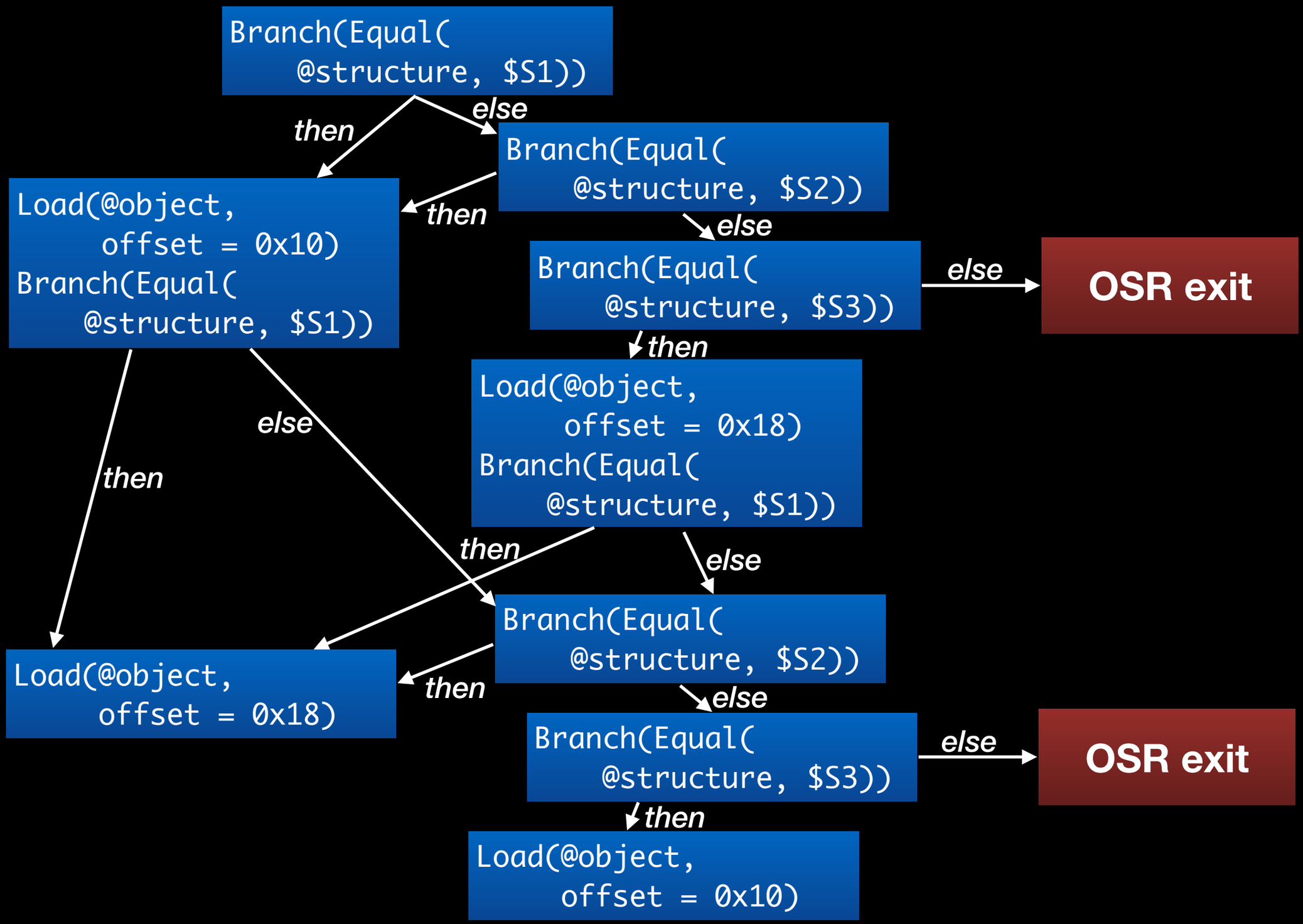
B3 IR:

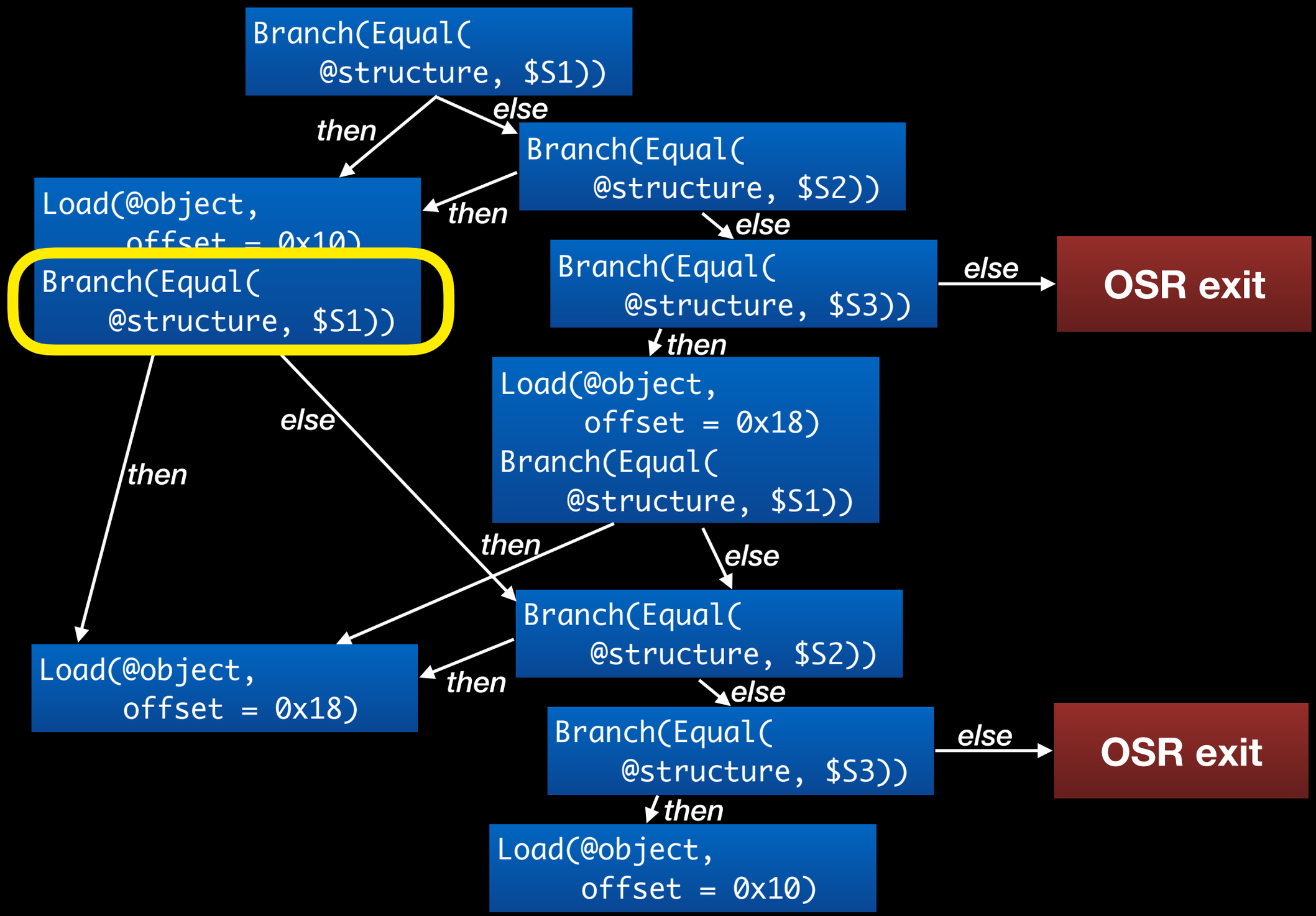


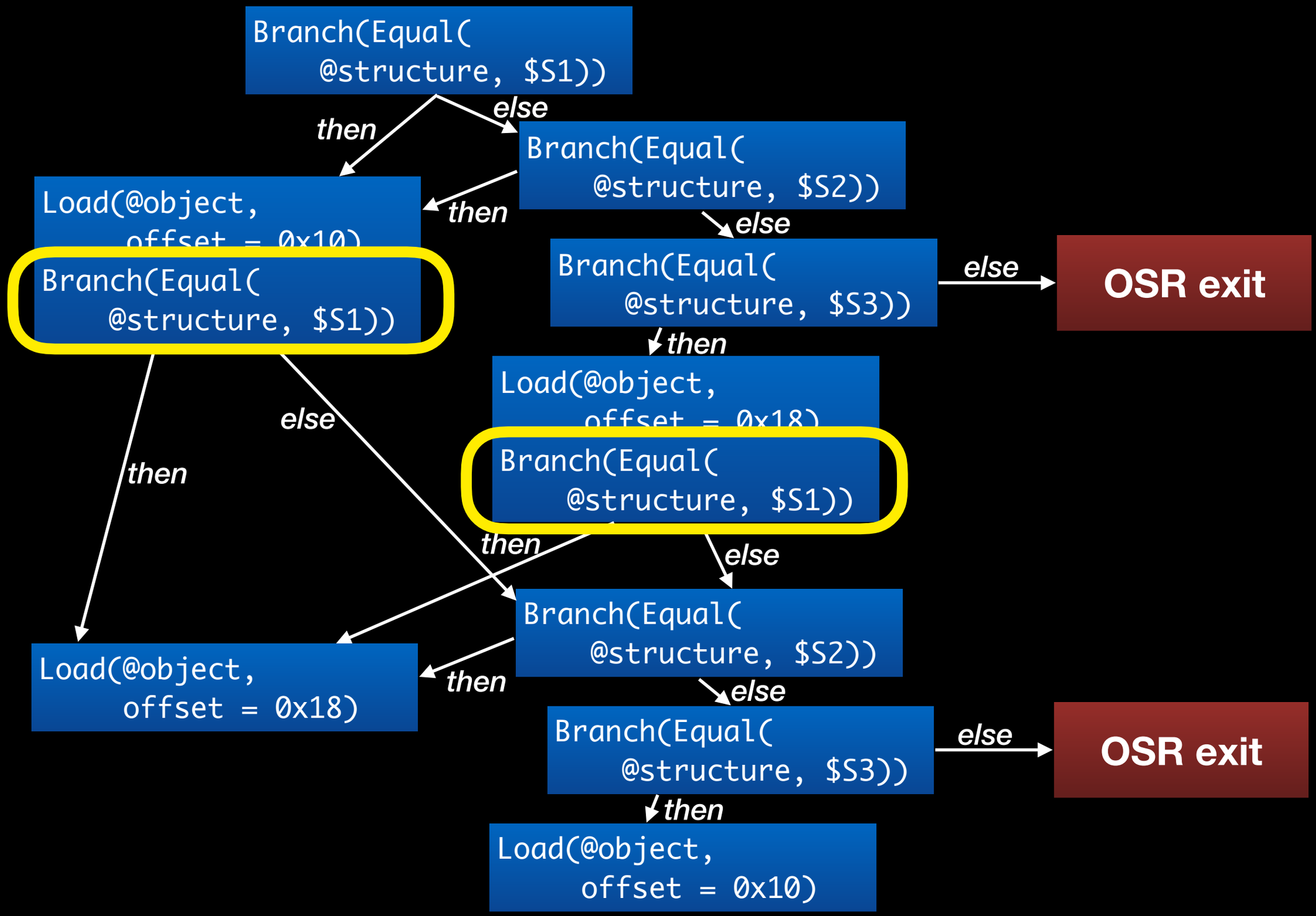

Branch(Equal(
@structure, $S1))
then
else
Branch(Equal(
@structure, $S2))
then
else
Branch(Equal(
@structure, $S3))
else
OSR exit
then
Load(@object,
offset = 0x18)
Load(@object,
offset = 0x10)
Branch(Equal(
@structure, $S1))
then
else
Branch(Equal(
@structure, $S2))
then
else
Branch(Equal(
@structure, $S3))
else
OSR exit
then
Load(@object,
offset = 0x18)
Load(@object,
offset = 0x10)

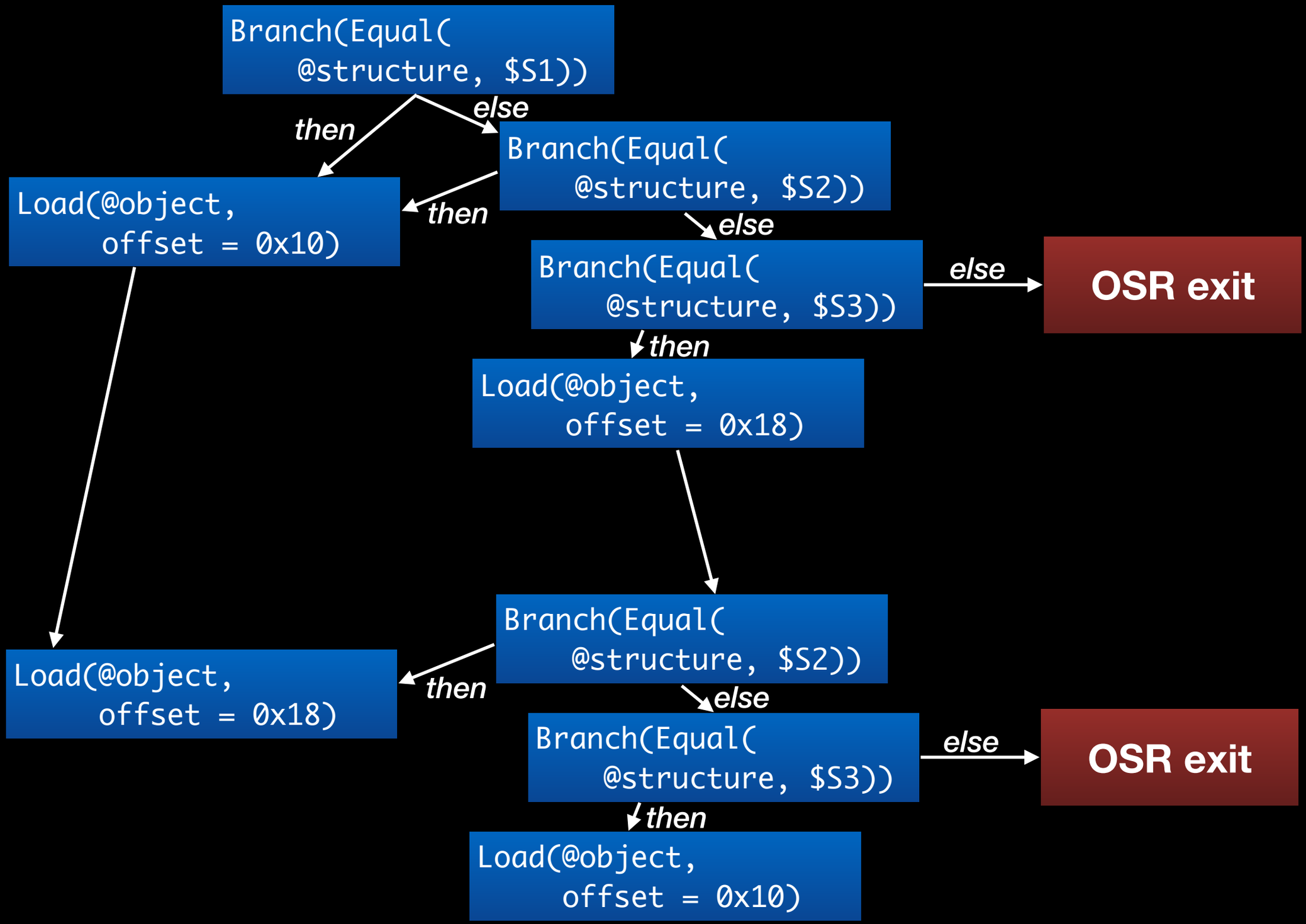

Branch(Equal(
@structure, $S1))
then
else
Load(@object,
offset = 0x10)
Branch(Equal(
@structure, $S2))
then
else
Branch(Equal(
@structure, $S3))
else
OSR exit
then
Load(@object,
offset = 0x18)
Branch(Equal(
@structure, $S1))
then
else
Load(@object,
offset = 0x18)
Branch(Equal(
@structure, $S2))
then
else
Branch(Equal(
@structure, $S3))
else
OSR exit
then
Load(@object,
offset = 0x10)

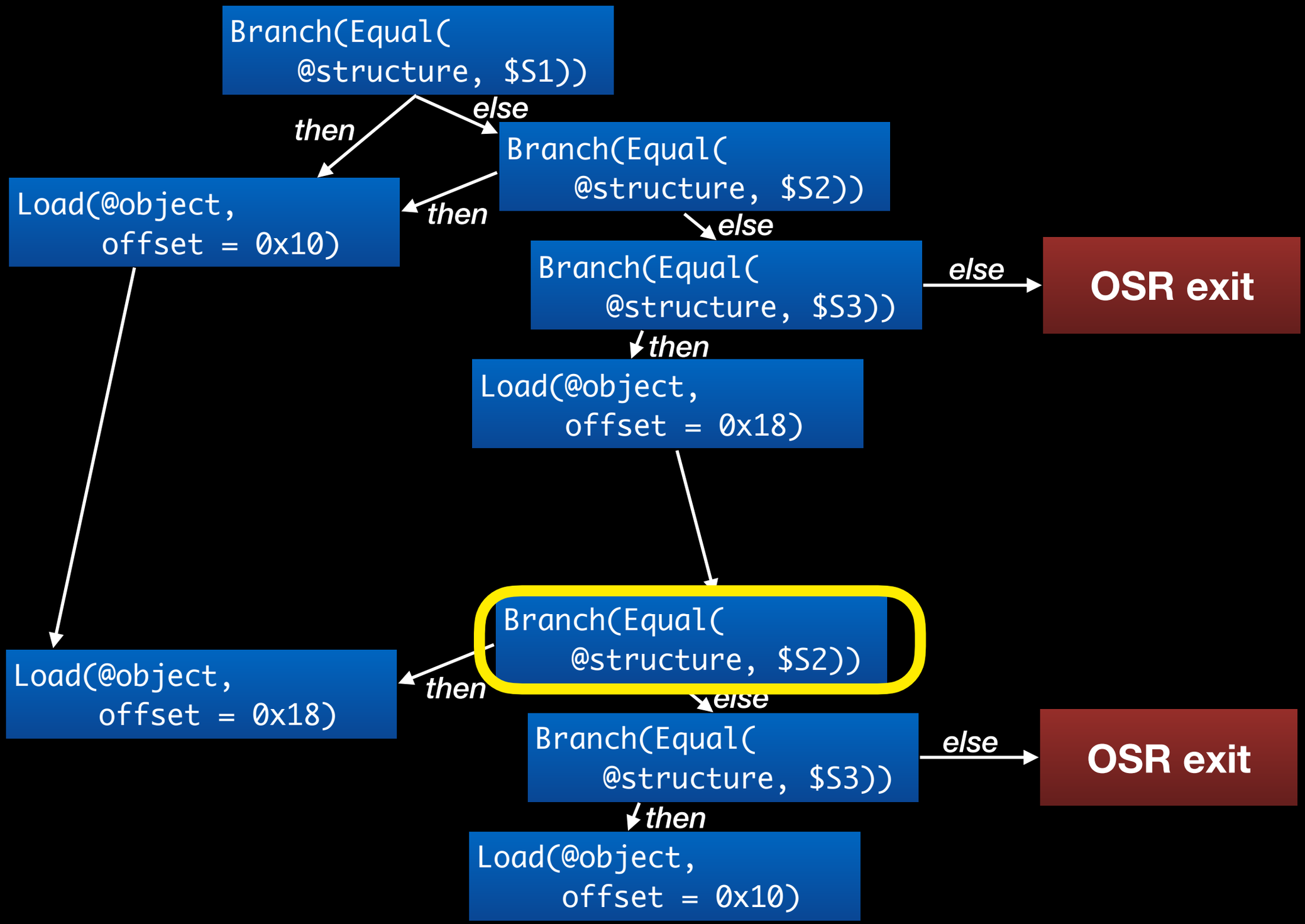

Branch(Equal(
@structure, $S1))
then
else
Load(@object,
offset = 0x10)
Branch(Equal(
@structure, $S2))
then
else
Branch(Equal(
@structure, $S3))
else
OSR exit
then
Load(@object,
offset = 0x18)
Branch(Equal(
@structure, $S1))
Branch(Equal(
@structure, $S1))
then
else
then
else
Load(@object,
offset = 0x18)
Branch(Equal(
@structure, $S2))
then
else
Branch(Equal(
@structure, $S3))
else
OSR exit
then
Load(@object,
offset = 0x10)

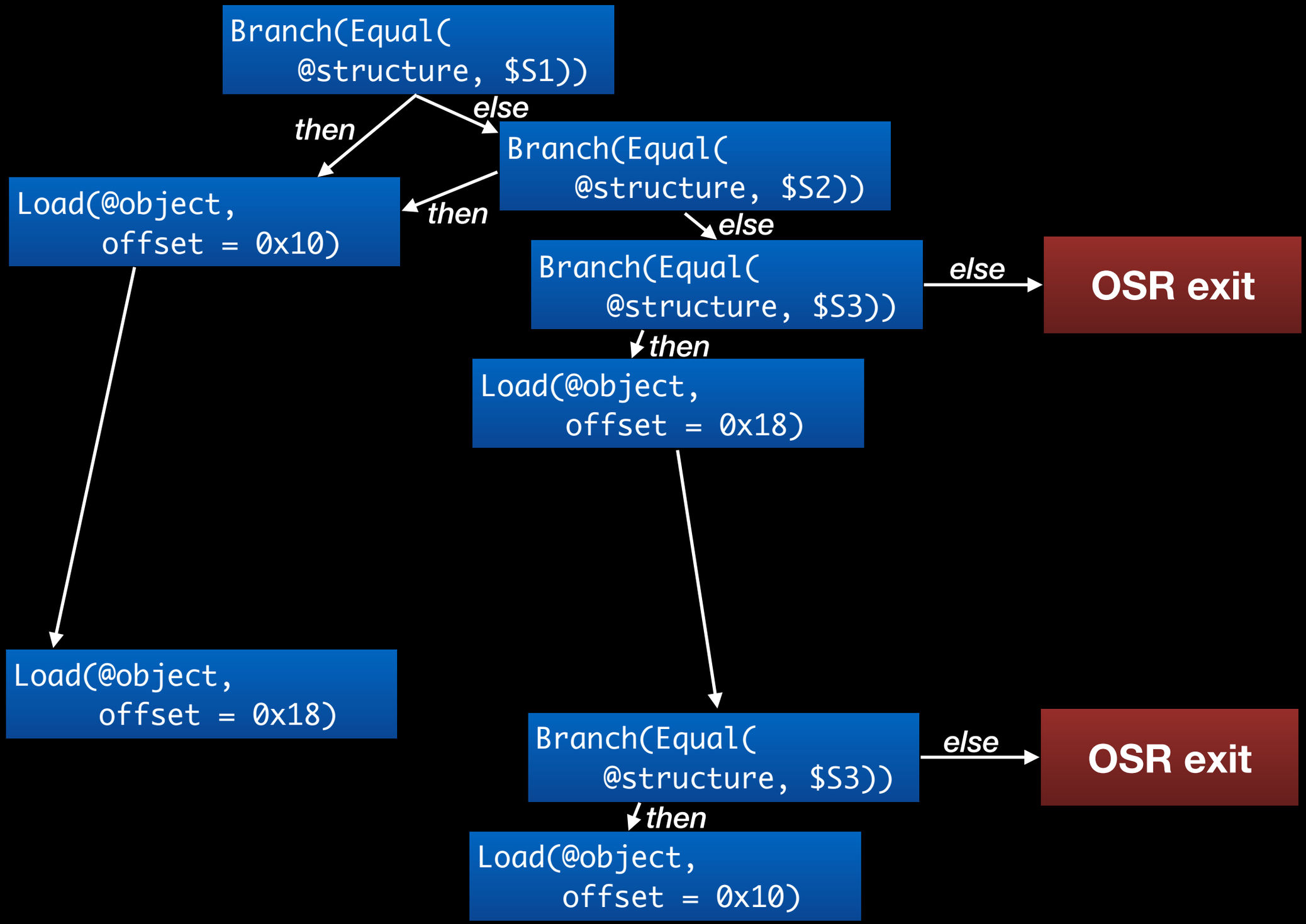

Branch(Equal(
@structure, $S1))
then
else
Branch(Equal(
@structure, $S2))
then
else
Branch(Equal(
@structure, $S3))
else
OSR exit
then
Load(@object,
offset = 0x10)
Branch(Equal(
@structure, $S1))
then
else
Load(@object,
offset = 0x18)
Load(@object,
offset = 0x18)
Branch(Equal(
@structure, $S1))
then
else
Branch(Equal(
@structure, $S2))
then
else
Branch(Equal(
@structure, $S3))
else
OSR exit
then
Load(@object,
offset = 0x10)

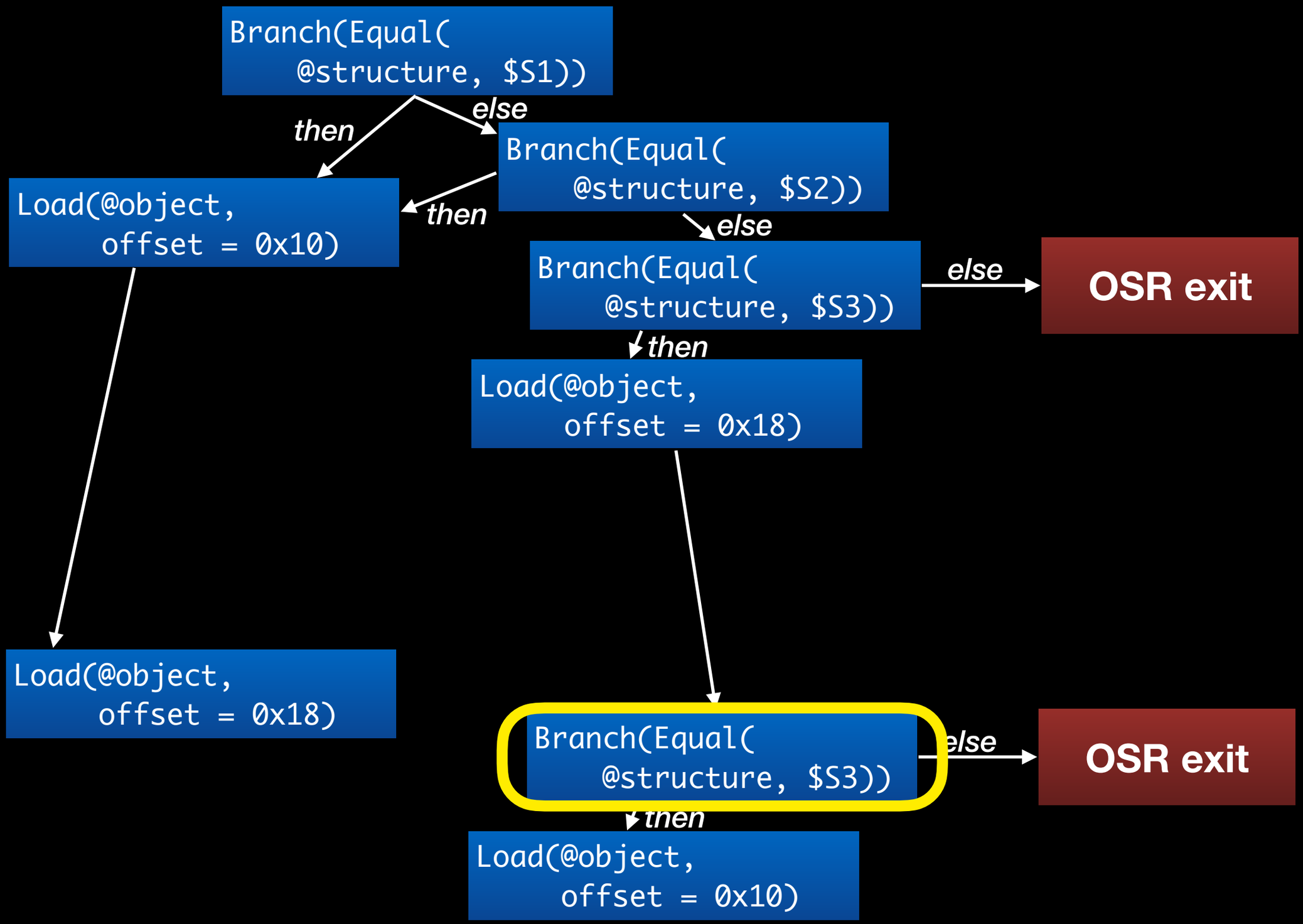

Branch(Equal(
@structure, $S1))
then
else
Branch(Equal(
@structure, $S2))
then
else
Load(@object,
offset = 0x10)
Branch(Equal(
@structure, $S1))
Branch(Equal(
@structure, $S3))
else
OSR exit
then
else
Load(@object,
offset = 0x18)
Branch(Equal(
@structure, $S1))
then
else
Load(@object,
offset = 0x18)
Branch(Equal(
@structure, $S2))
then
else
Branch(Equal(
@structure, $S3))
else
OSR exit
then
Load(@object,
offset = 0x10)

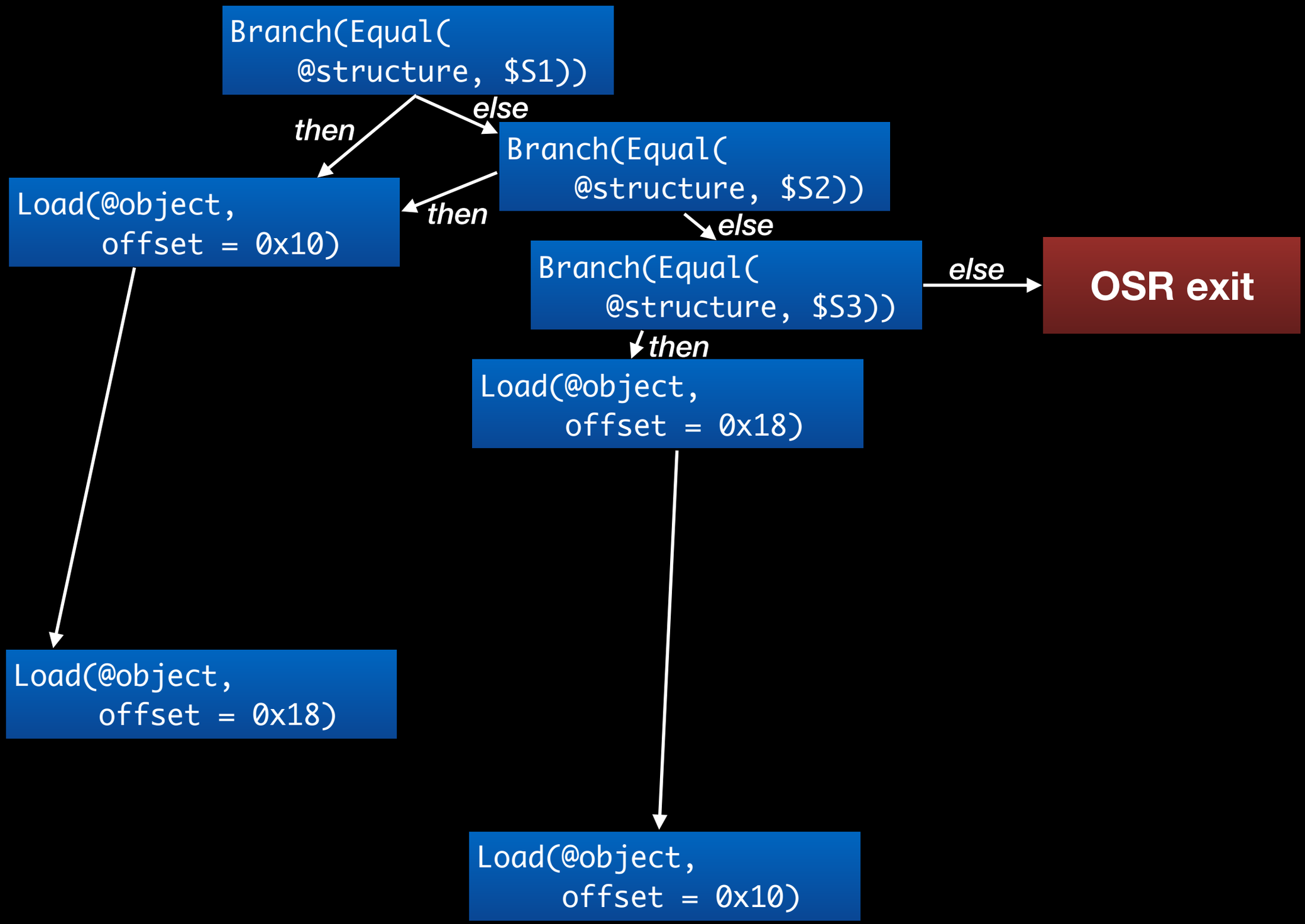

Branch(Equal(
@structure, $S1))
then
else
Load(@object,
offset = 0x10)
Branch(Equal(
@structure, $S2))
then
else
Branch(Equal(
@structure, $S1))
Branch(Equal(
@structure, $S3))
else
OSR exit
then
else
Load(@object,
offset = 0x18)
Branch(Equal(
@structure, $S1))
then
else
Load(@object,
offset = 0x18)
Branch(Equal(
@structure, $S2))
then
else
Branch(Equal(
@structure, $S3))
else
OSR exit
then
Load(@object,
offset = 0x10)

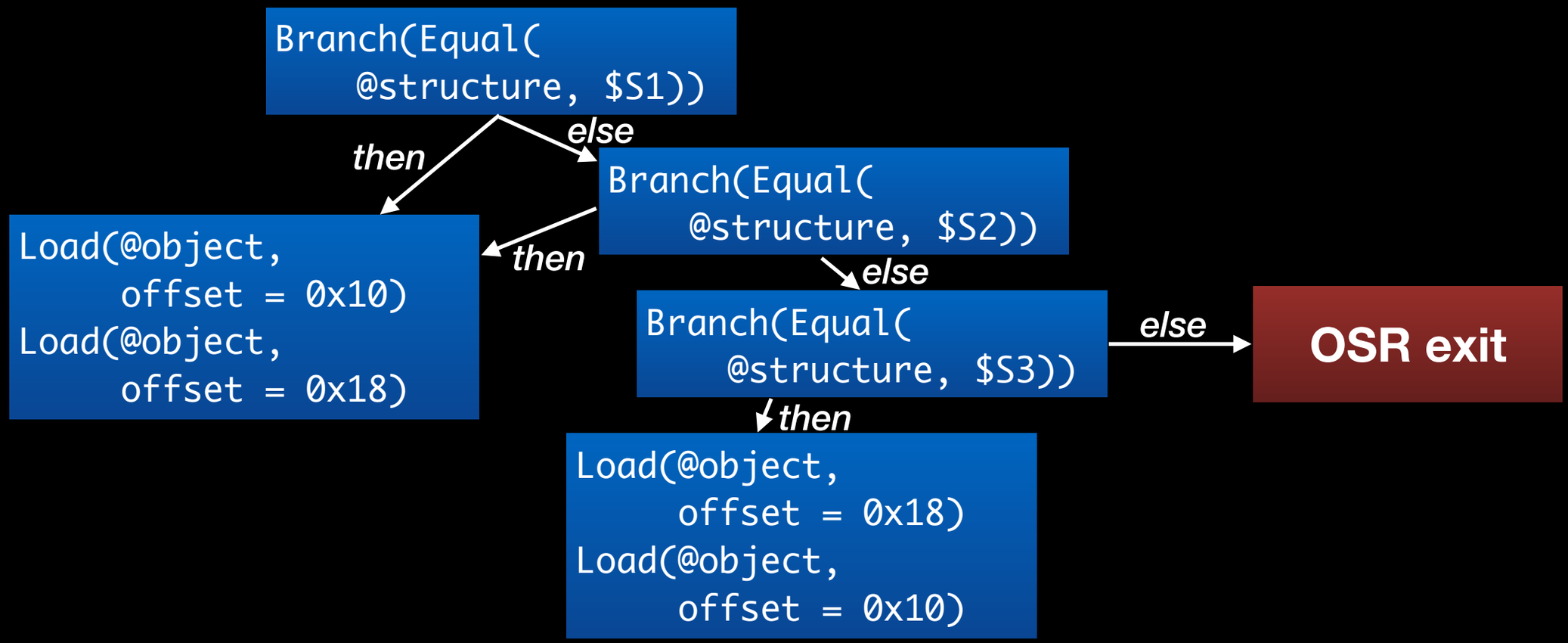

Branch(Equal(
@structure, $S1))
then
else
Branch(Equal(
@structure, $S2))
then
else
Branch(Equal(
@structure, $S3))
else
OSR exit
then
Load(@object,
offset = 0x18)
Load(@object,
offset = 0x10)
Branch(Equal(
@structure, $S2))
then
else
Branch(Equal(
@structure, $S3))
else
OSR exit
then
Load(@object,
offset = 0x18)
Load(@object,
offset = 0x10)


Branch(Equal(
@structure, $S1))
then
else
Branch(Equal(
@structure, $S2))
then
else
Branch(Equal(
@structure, $S3))
else
OSR exit
then
Load(@object,
offset = 0x10)
Load(@object,
offset = 0x18)
Branch(Equal(
@structure, $S2))
then
else
Branch(Equal(
@structure, $S3))
else
OSR exit
then
Load(@object,
offset = 0x18)
Load(@object,
offset = 0x10)

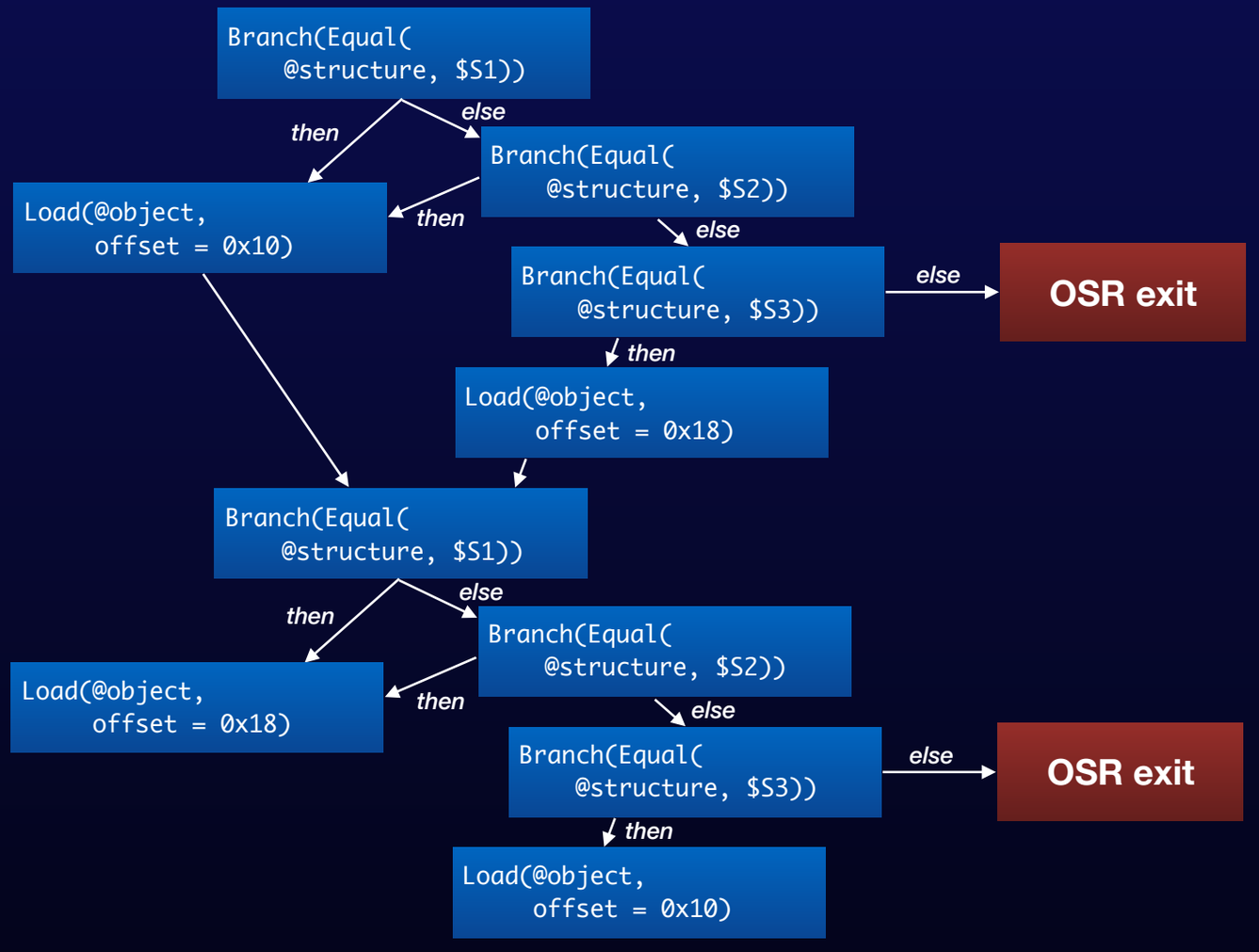

Branch(Equal(
@structure, $S1))
then
else
Load(@object,
offset = 0x10)
Branch(Equal(
@structure, $S2))
then
else
Branch(Equal(
@structure, $S3))
else
OSR exit
then
Load(@object,
offset = 0x18)
Load(@object,
offset = 0x18)
Branch(Equal(
@structure, $S3))
else
OSR exit
then
Load(@object,
offset = 0x10)


Branch(Equal(
@structure, $S1))
then
else
Load(@object,
offset = 0x10)
Branch(Equal(
@structure, $S2))
then
else
Branch(Equal(
@structure, $S3))
else
OSR exit
then
Load(@object,
offset = 0x18)
Load(@object,
offset = 0x18)
Branch(Equal(
@structure, $S3))
else
OSR exit
then
Load(@object,
offset = 0x10)

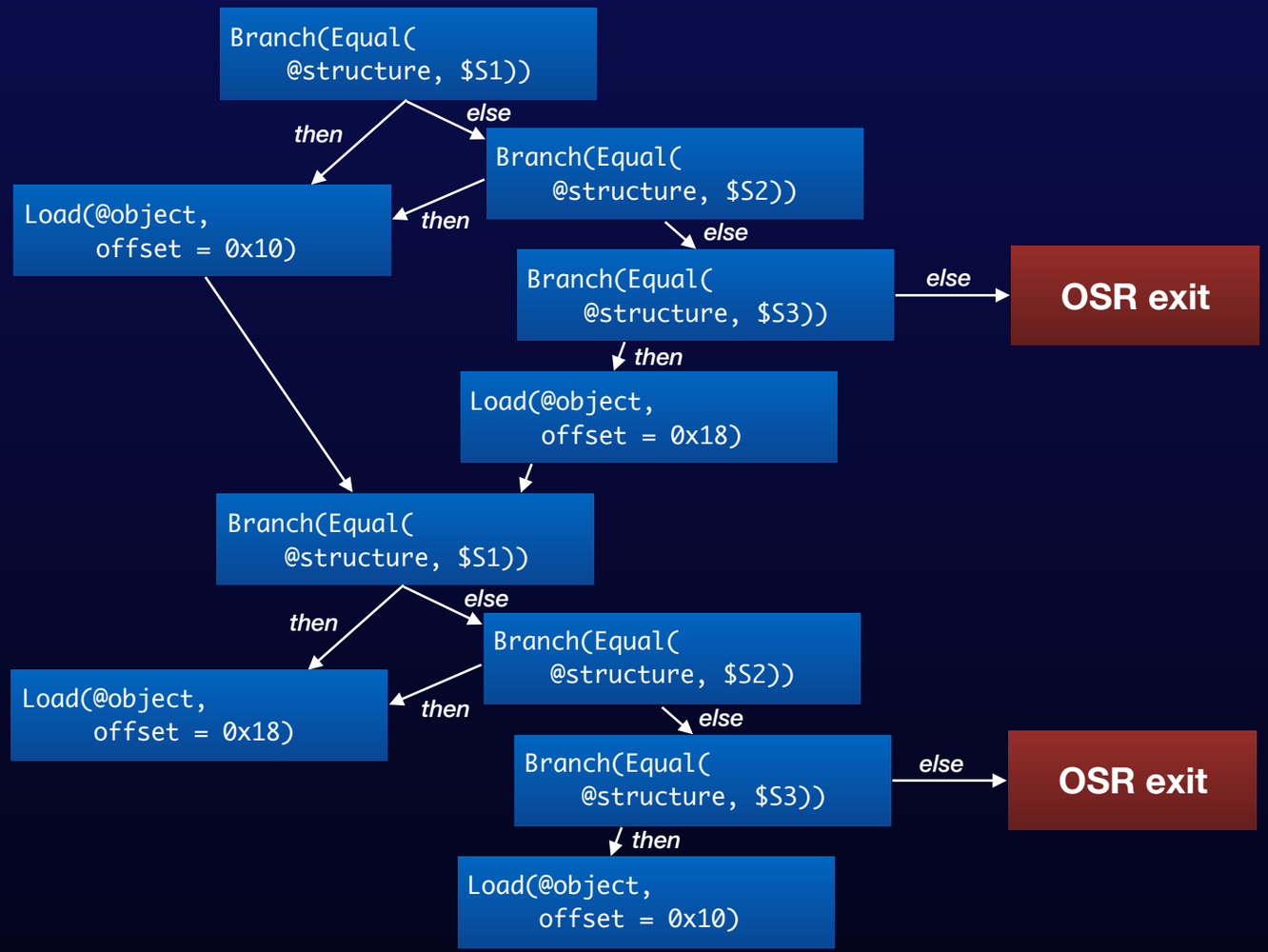

Branch(Equal(
@structure, $S1))
then
else
Branch(Equal(
@structure, $S2))
then
else
Branch(Equal(
@structure, $S3))
else
OSR exit
then
Load(@object,
offset = 0x10)
Load(@object,
offset = 0x18)
Load(@object,
offset = 0x18)
Load(@object,
offset = 0x10)


Branch(Equal(
@structure, $S1))
then
else
Load(@object,
offset = 0x10)
Load(@object,
offset = 0x18)
Branch(Equal(
@structure, $S2))
then
else
Branch(Equal(
@structure, $S3))
else
OSR exit
then
Load(@object,
offset = 0x18)
Load(@object,
offset = 0x10)

B3 IR before:

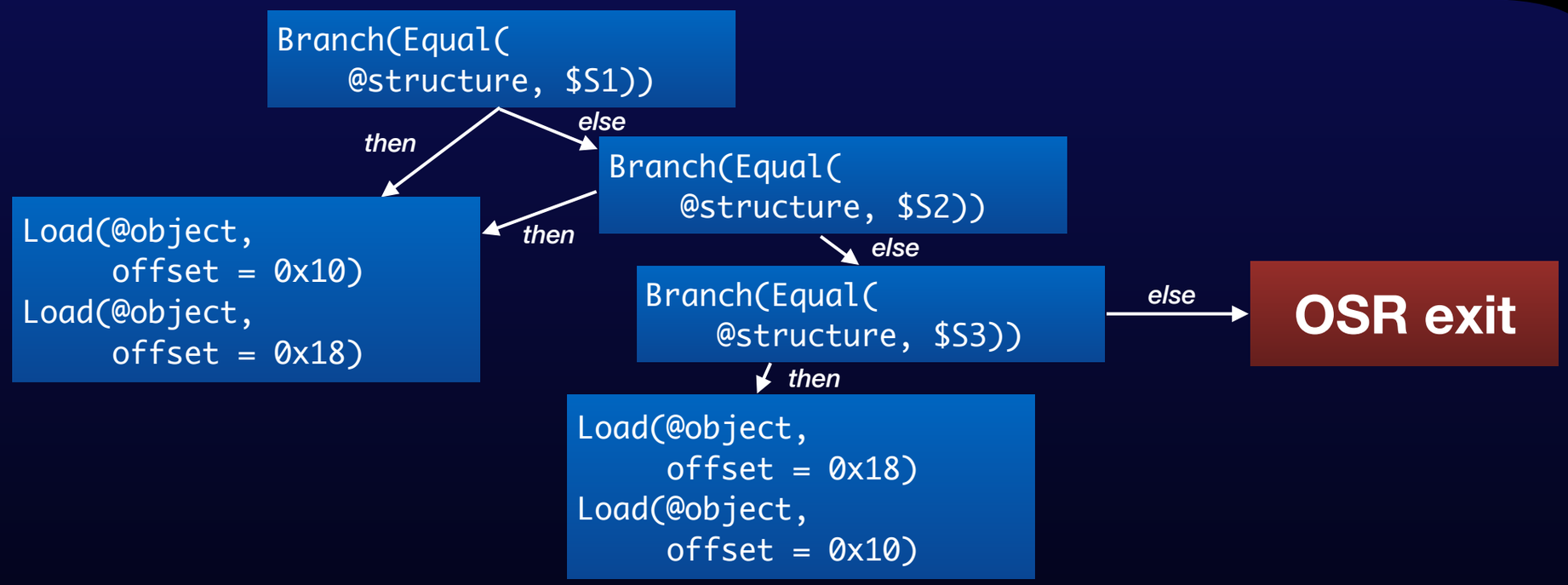

Branch(Equal(
@structure, $S1))
then
else
Branch(Equal(
@structure, $S2))
Load(@object,
offset = 0x10)
then
else
Branch(Equal(
@structure, $S3))
else
OSR exit
then
Load(@object,
offset = 0x18)
Branch(Equal(
@structure, $S1))
then
else
Branch(Equal(
@structure, $S2))
Load(@object,
offset = 0x18)
then
else
Branch(Equal(
@structure, $S3))
else
OSR exit
then
Load(@object,
offset = 0x10)

B3 IR before:

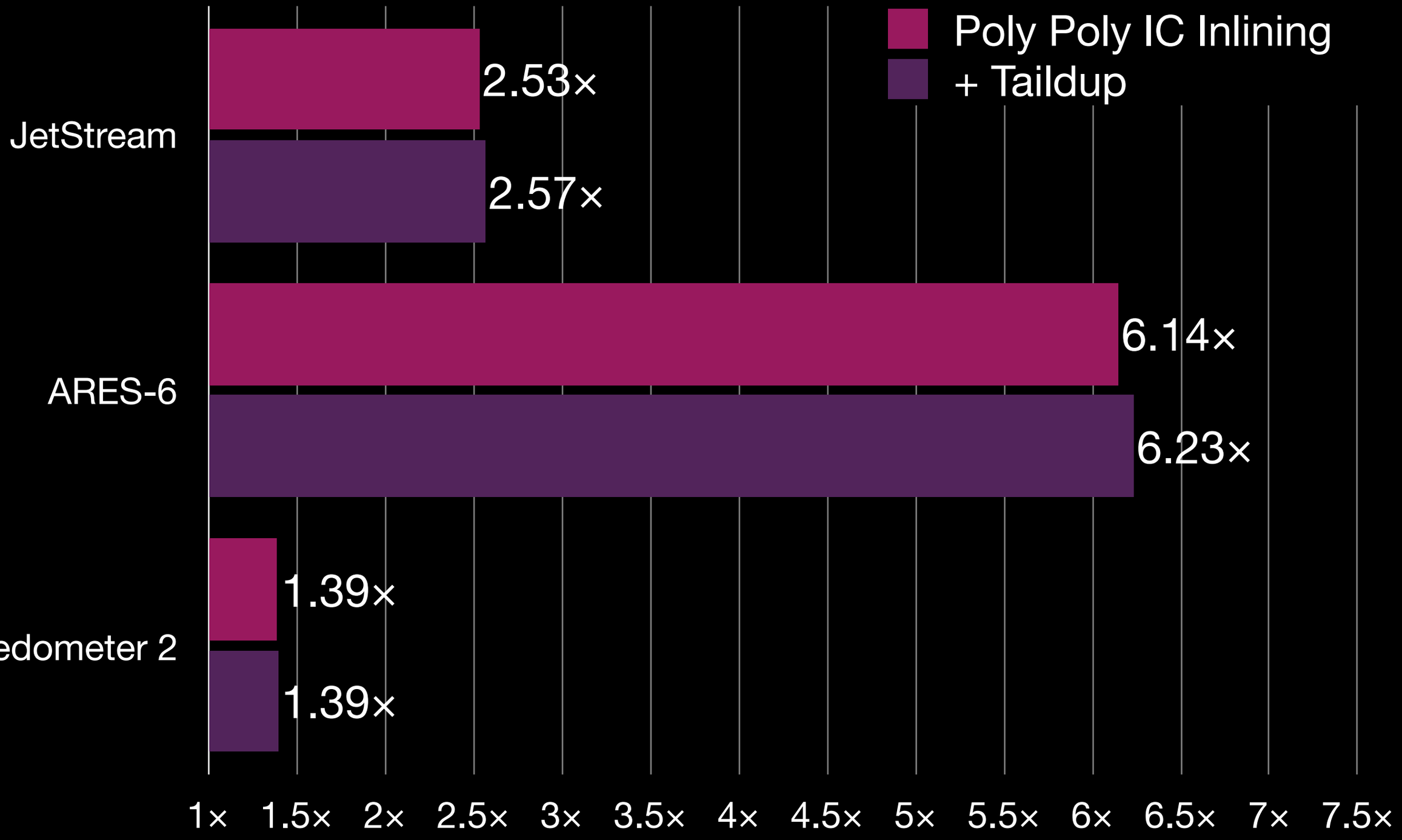


duplicateTails(procedure);
foldPathConstants(procedure);

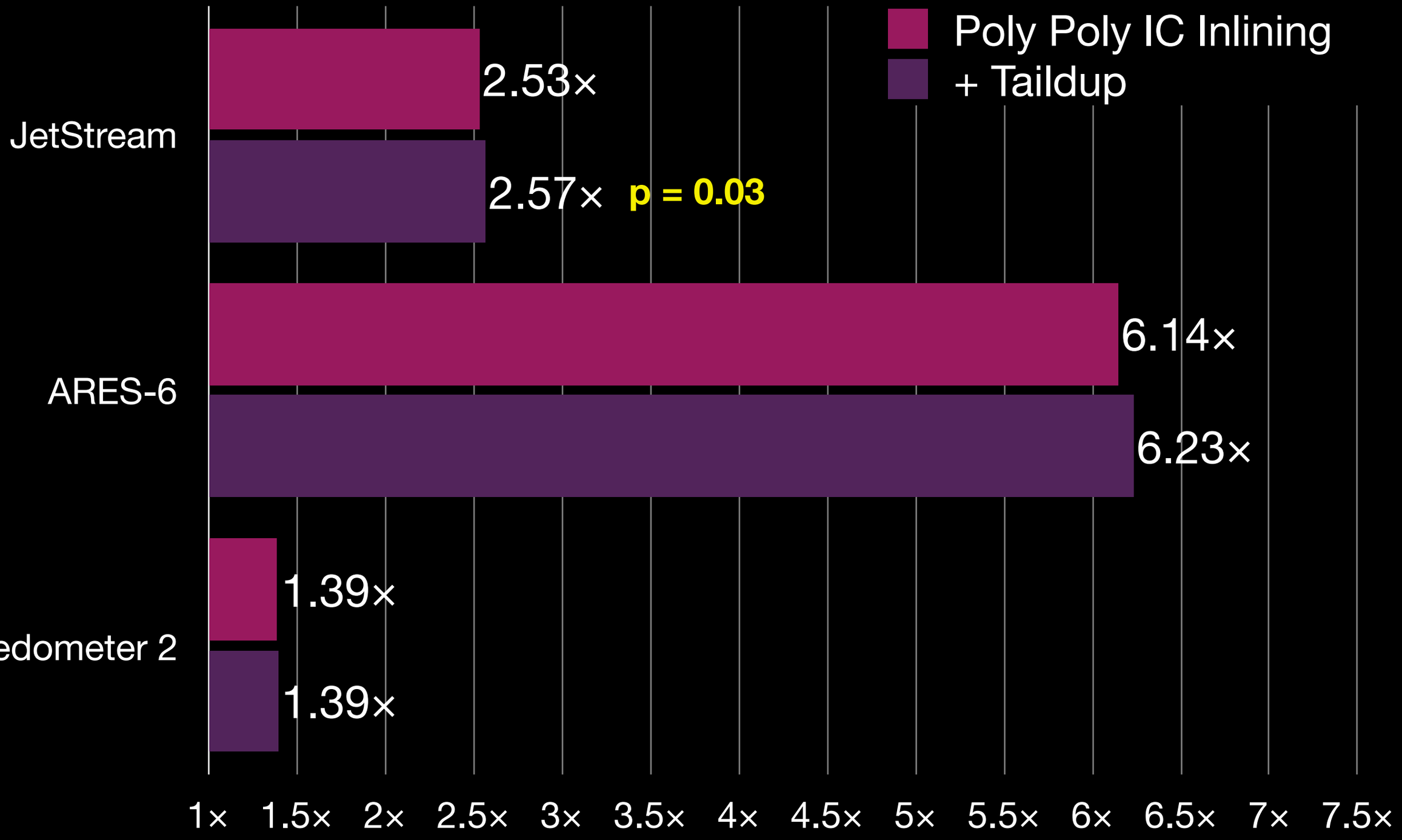
B3 IR after:



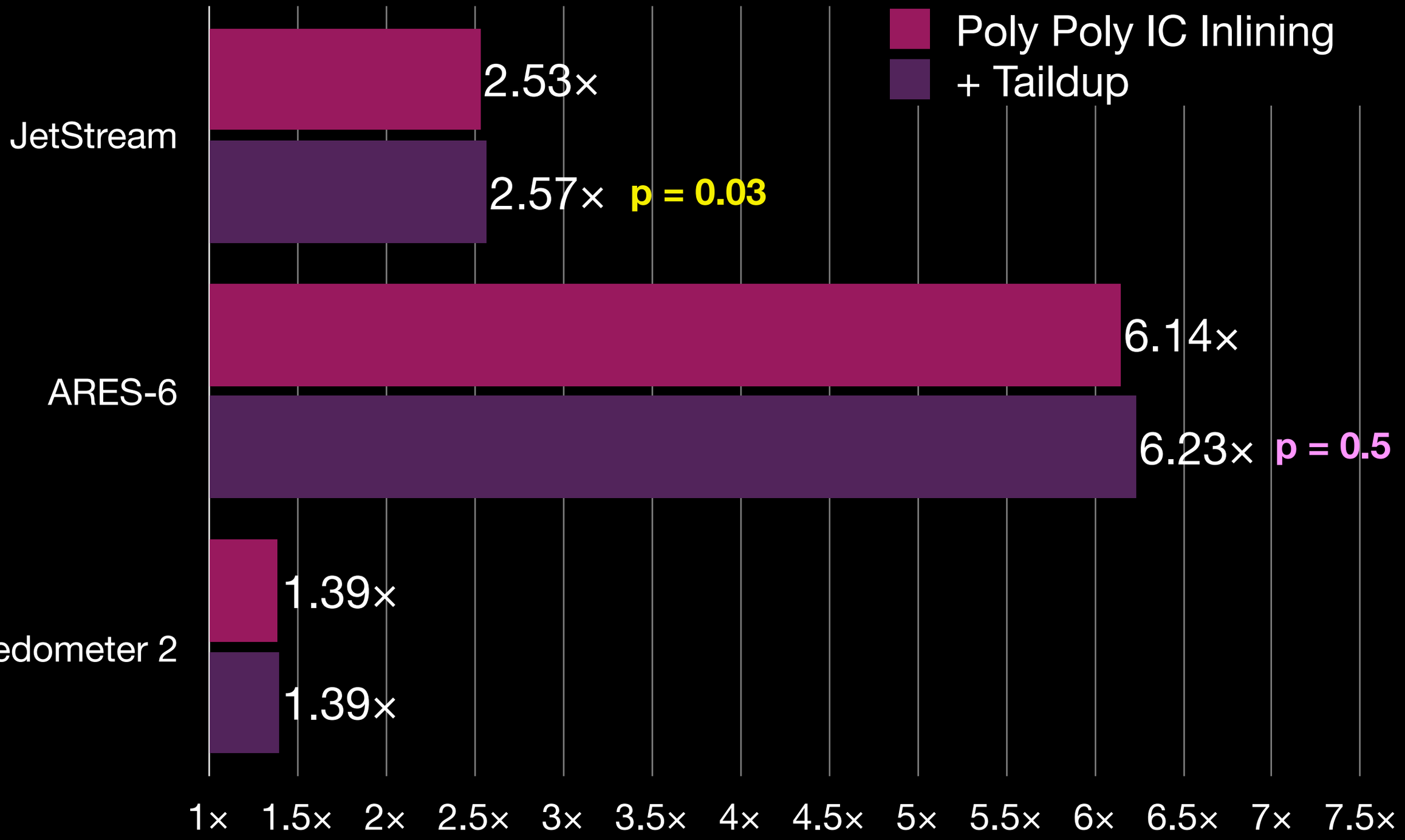
Taildup Speed-up



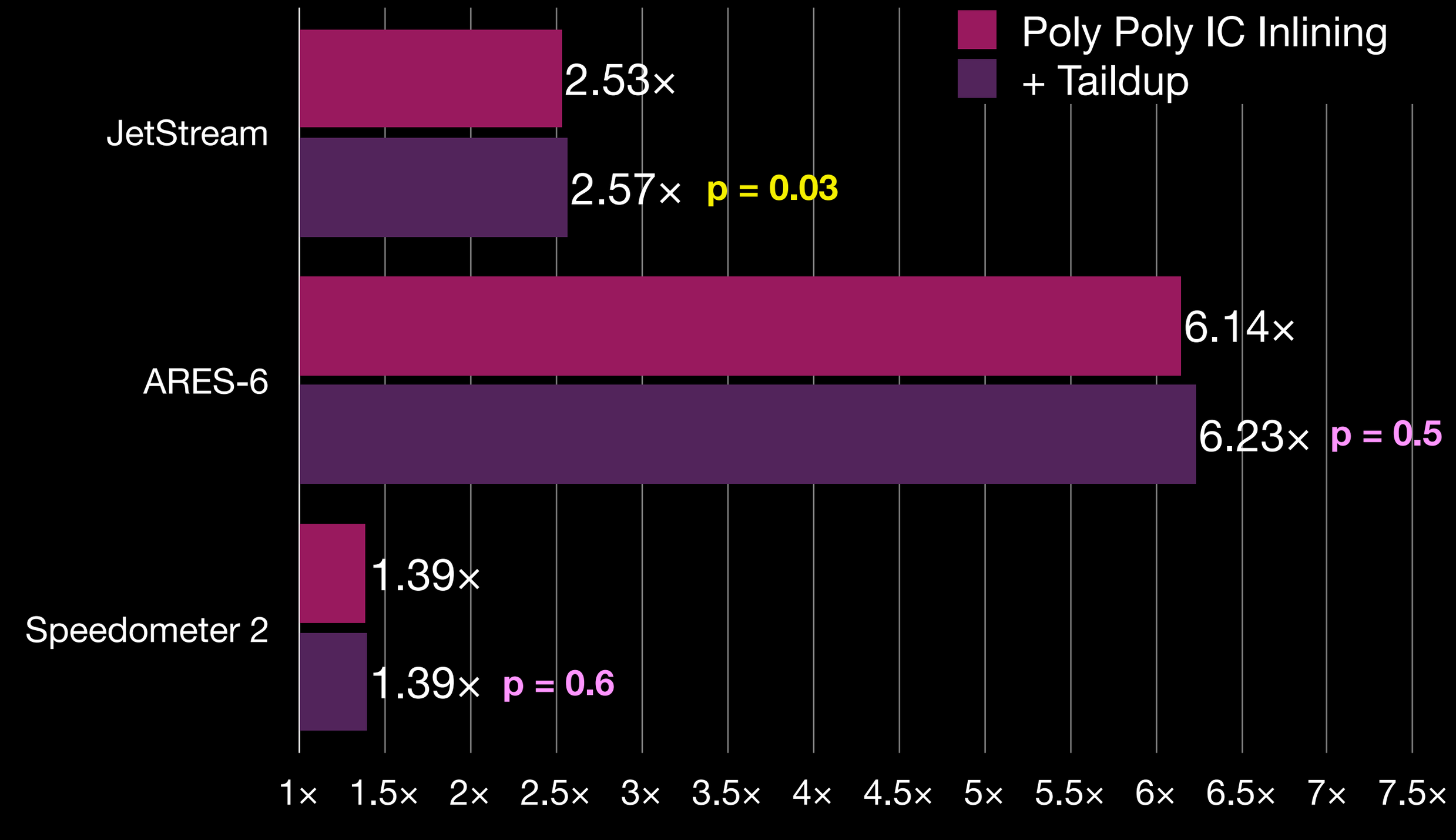
Taildup Speed-up



Taildup Speed-up

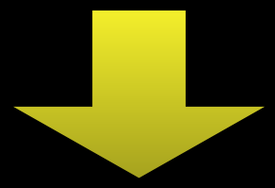


Taildup Speed-up



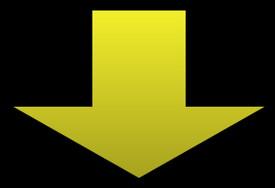
o.f.g

o.f.g



```
a: CheckStructure(@o, S1)
b: GetByOffset(@o)
c: CheckStructure(@b, S2)
d: GetByOffset(@c)
```

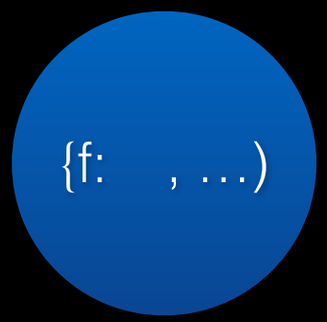
o.f.g



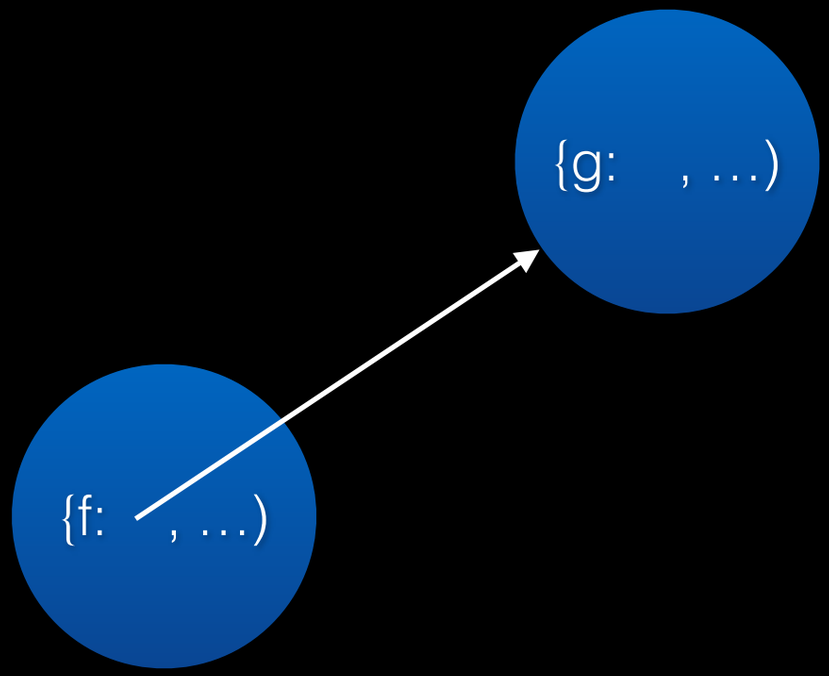
```
a: CheckStructure(@o, S1)
b: GetByOffset(@o)
c: CheckStructure(@b, S2)
d: GetByOffset(@c)
```

Property Type Inference

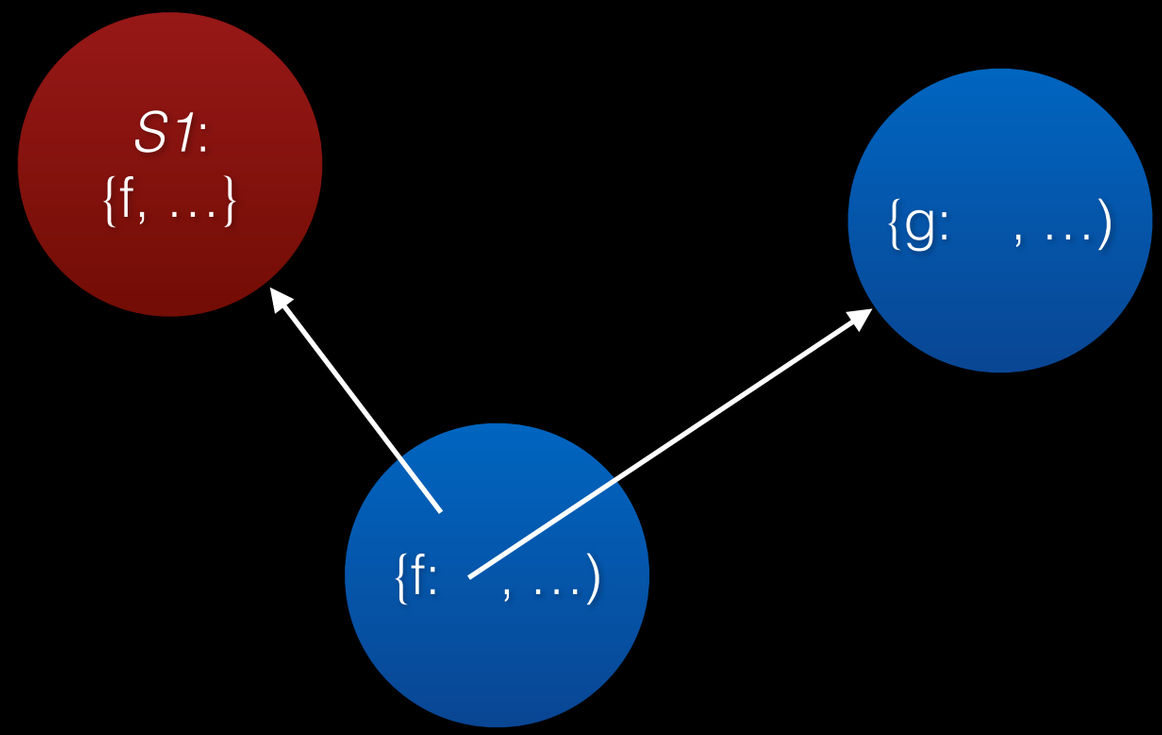
Property Type Inference



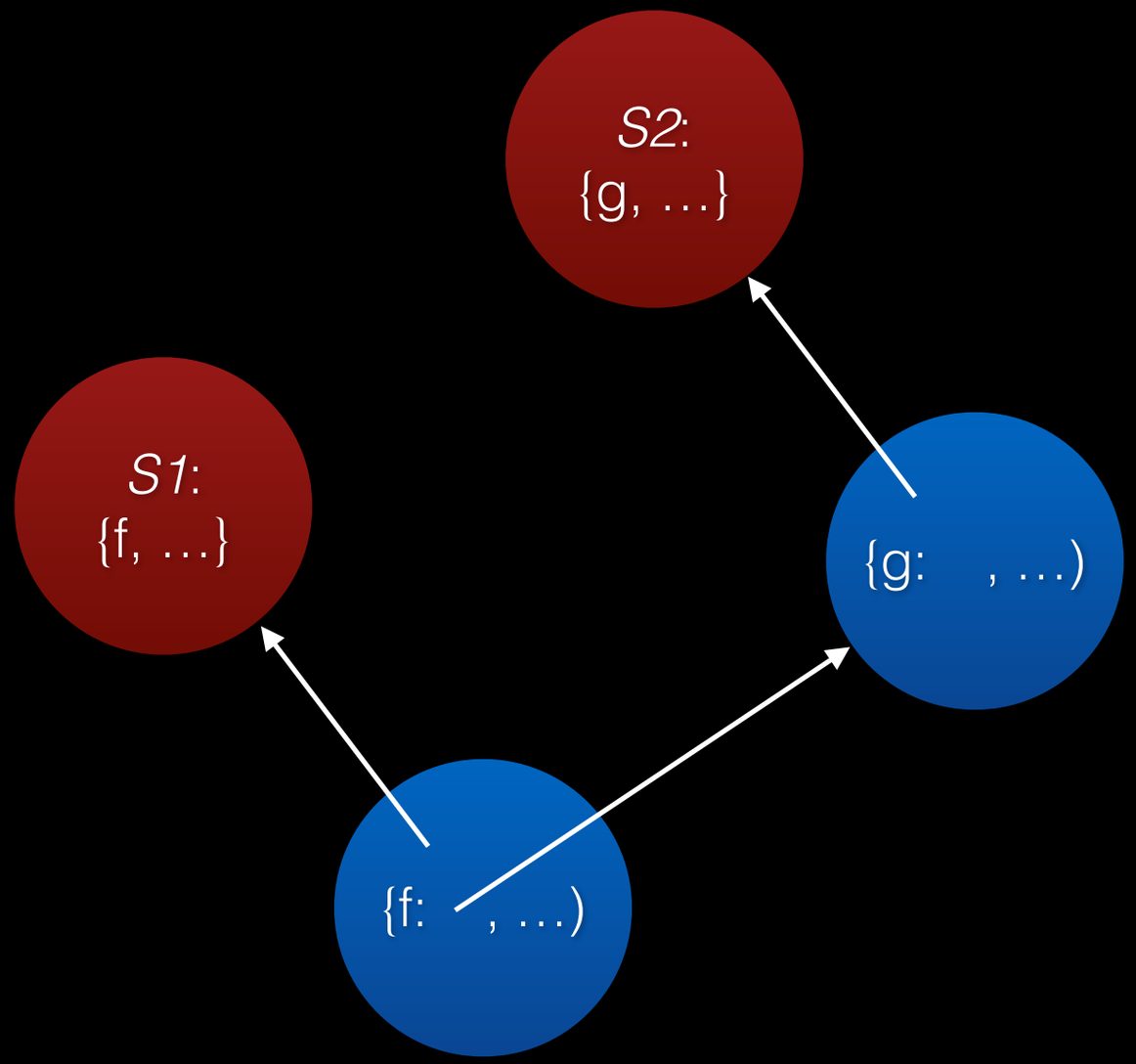
Property Type Inference



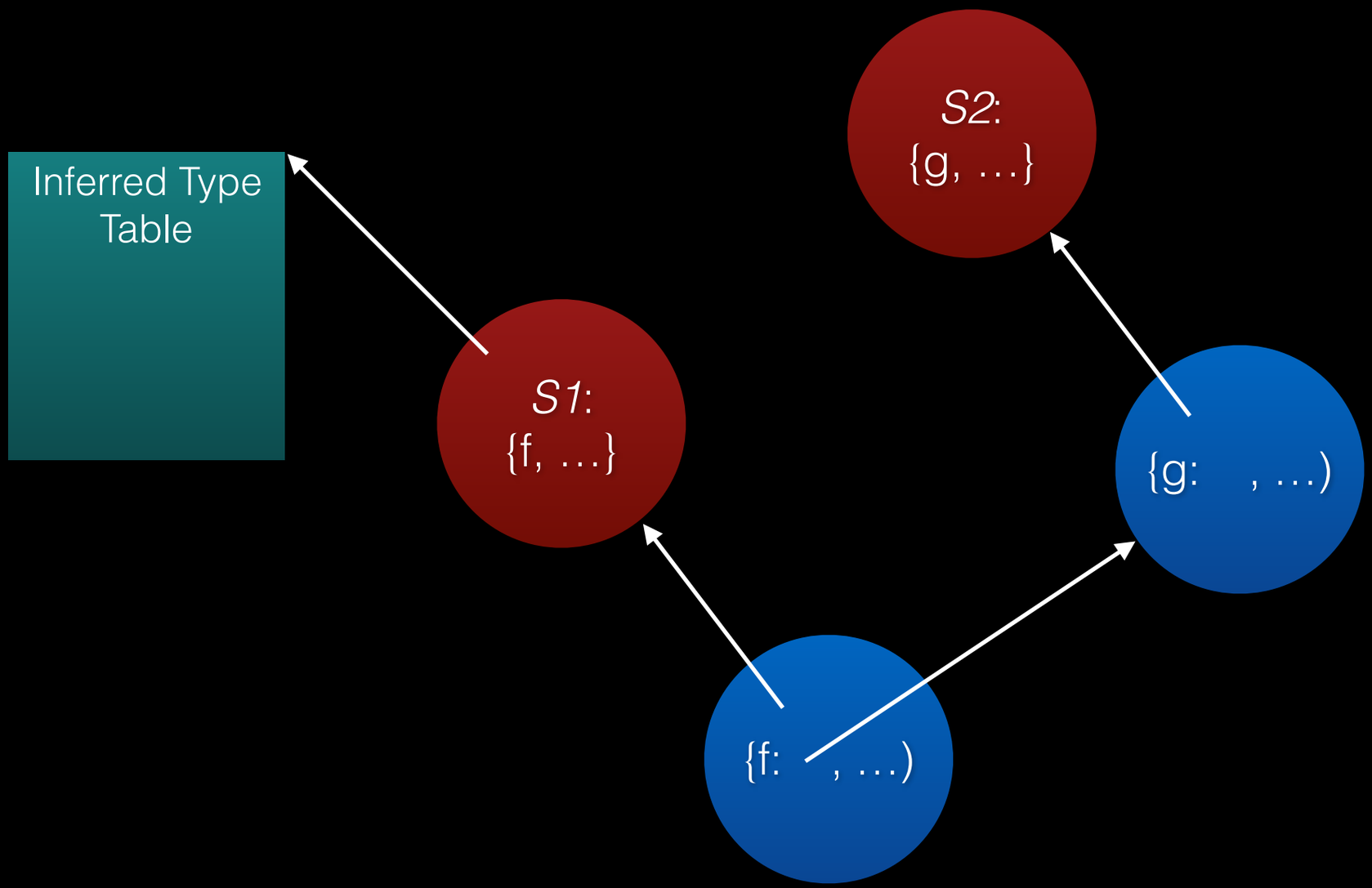
Property Type Inference



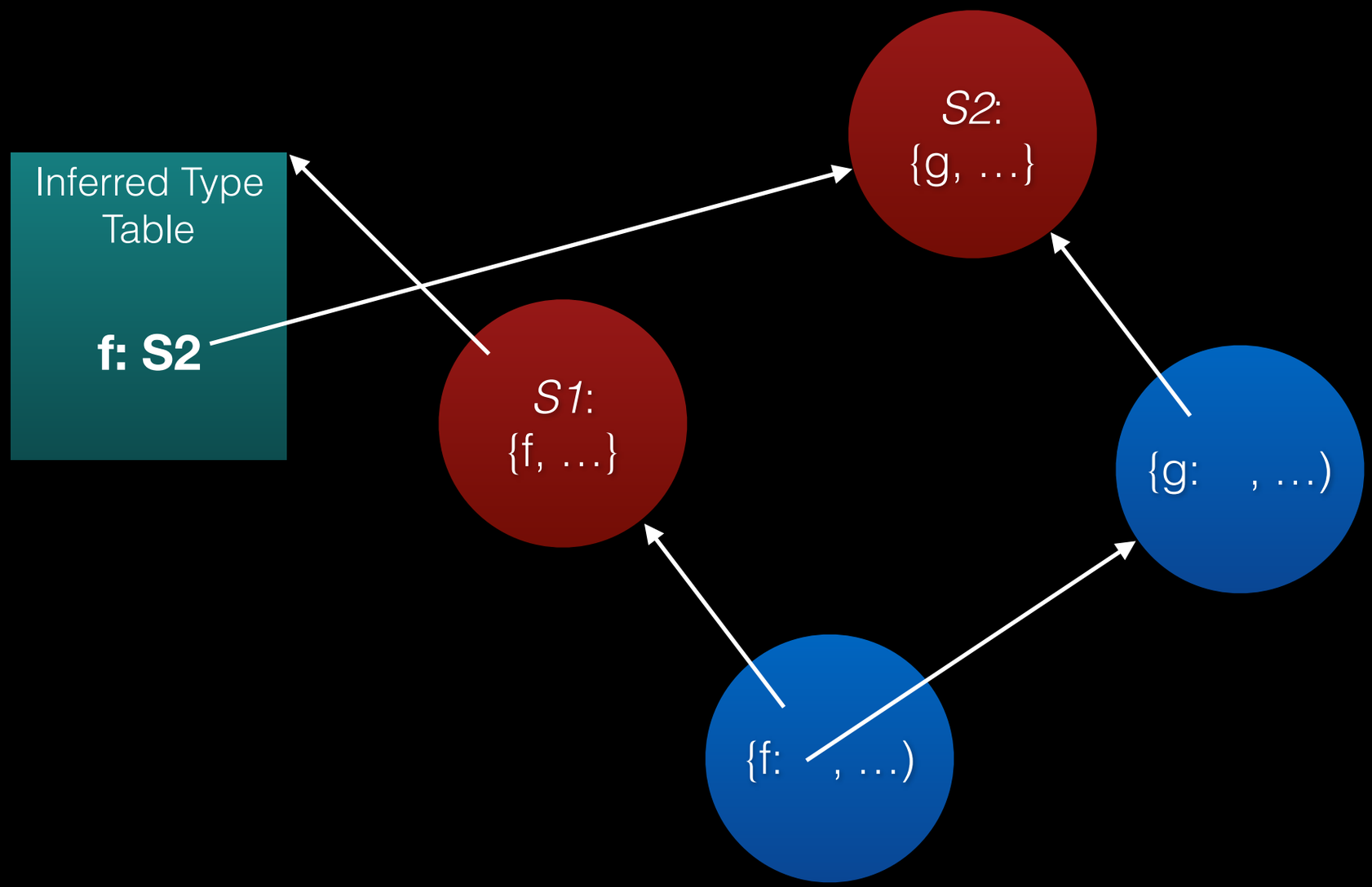
Property Type Inference



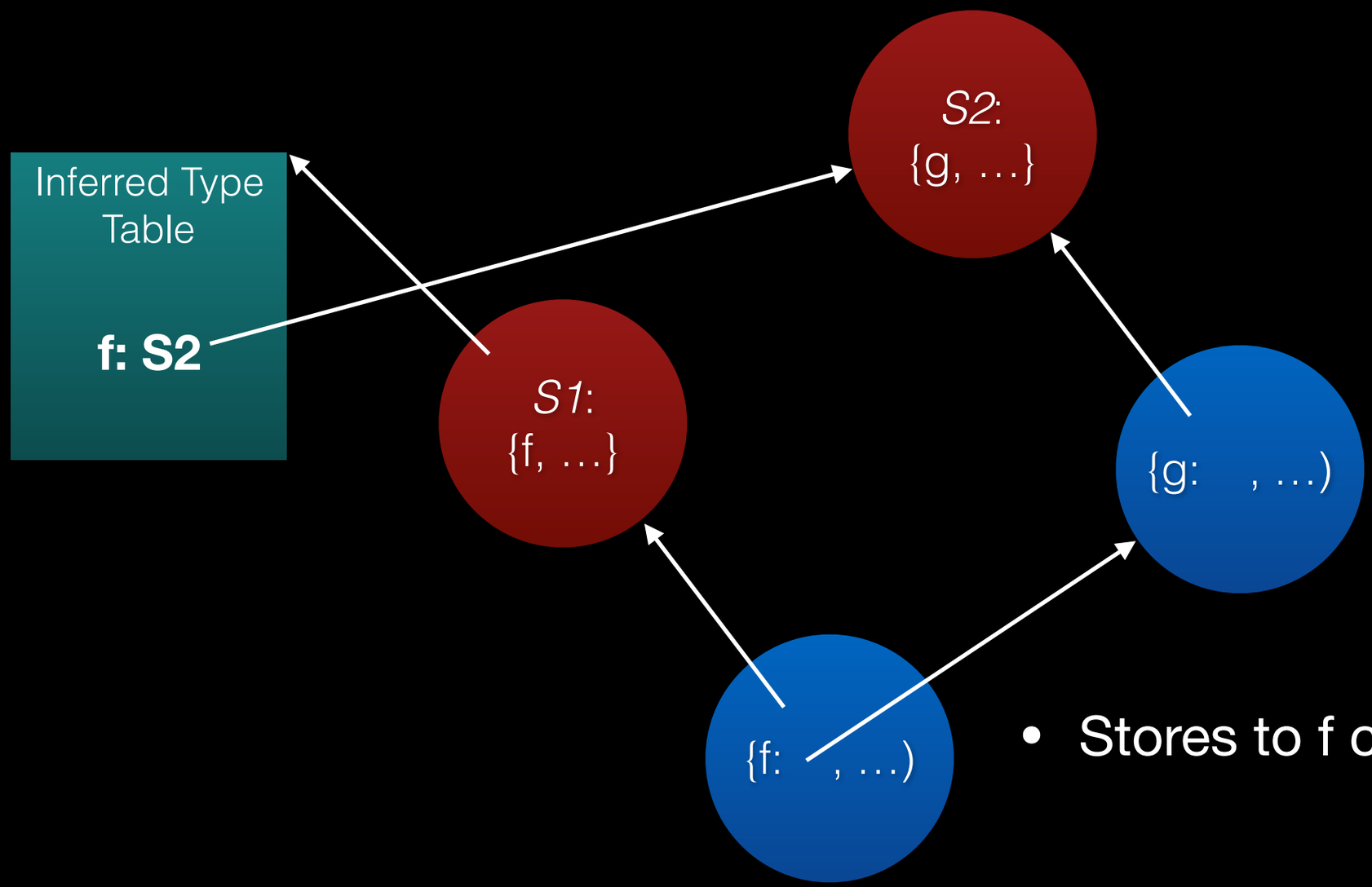
Property Type Inference



Property Type Inference

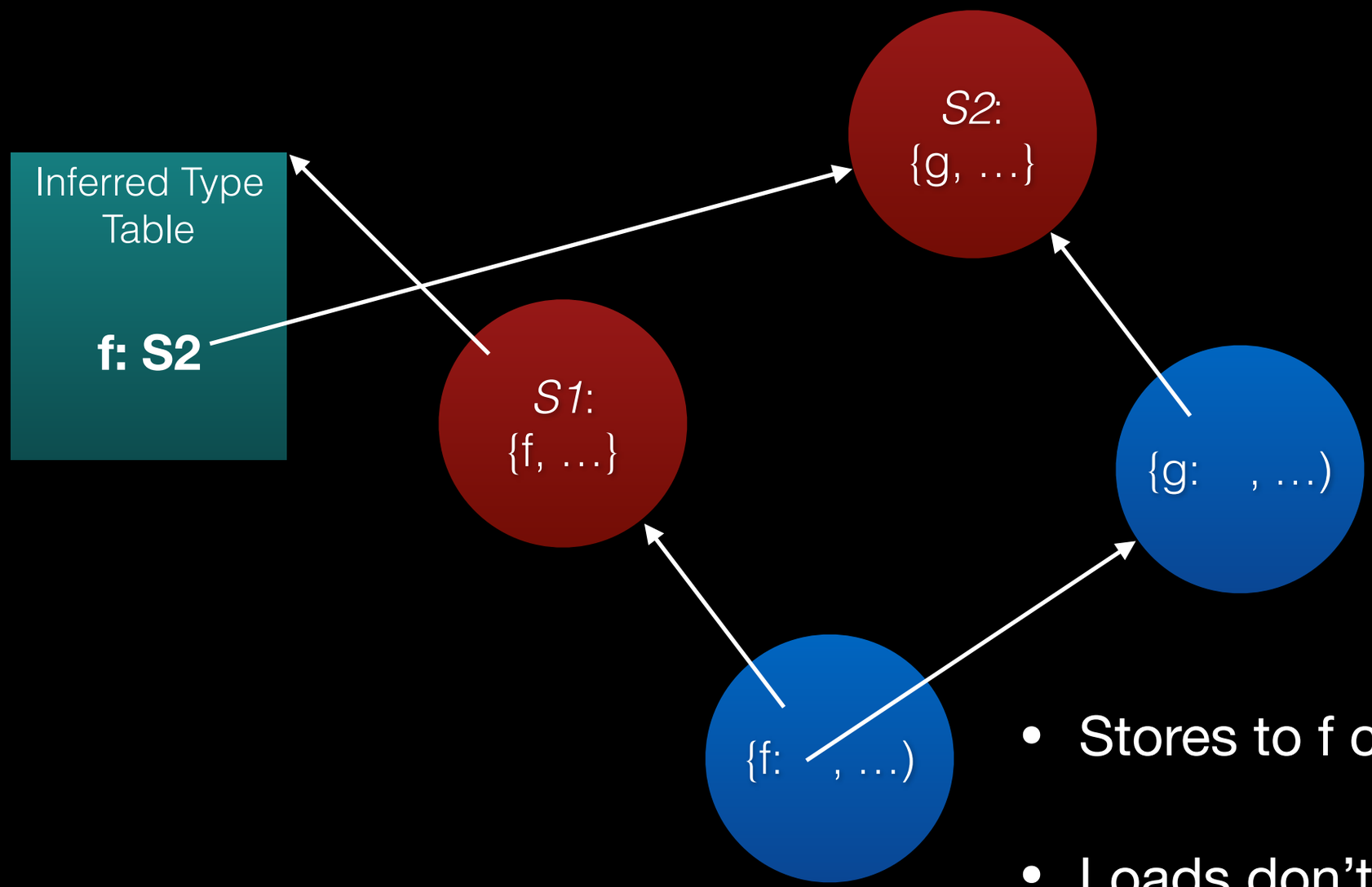


Property Type Inference



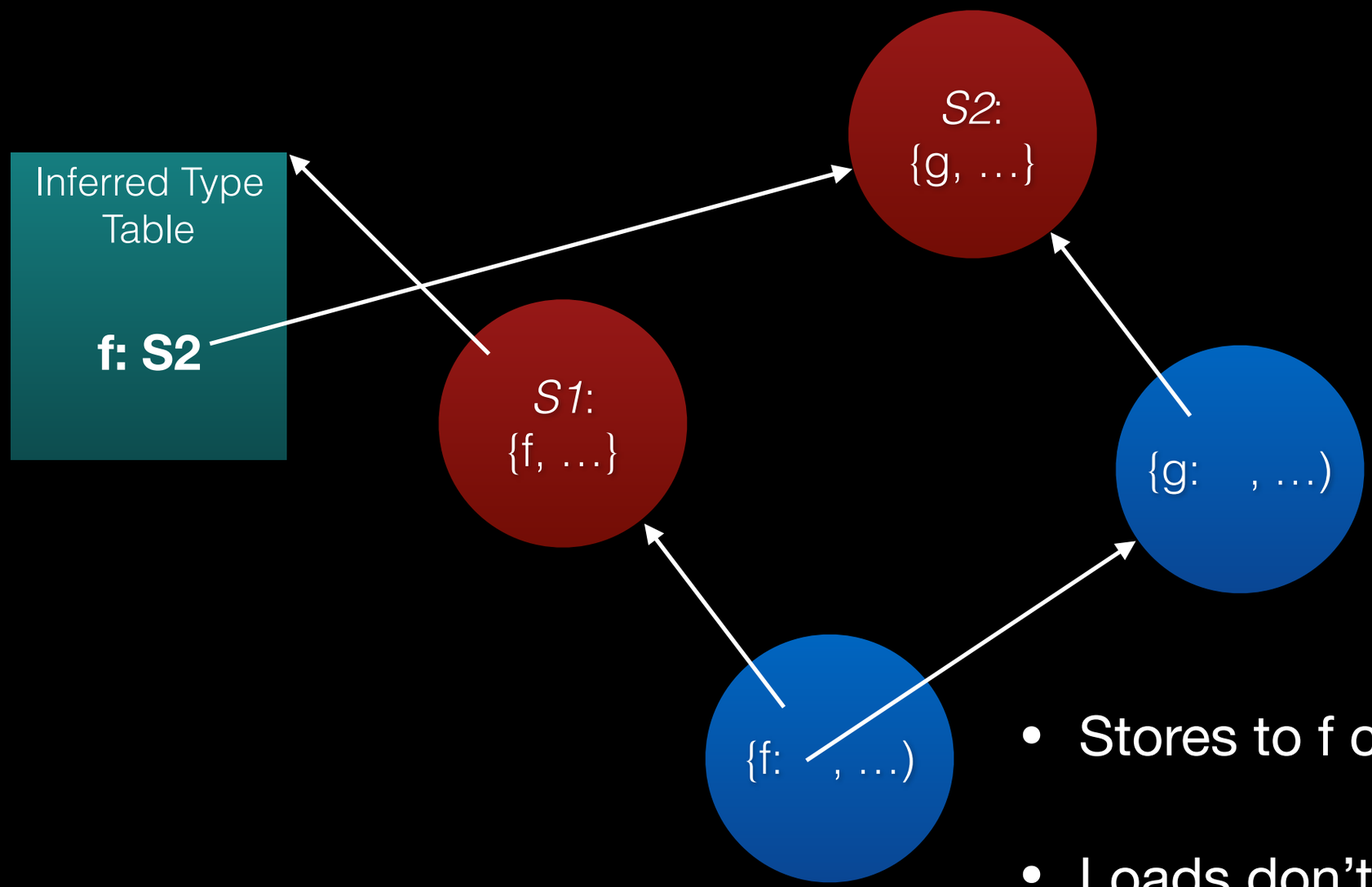
- Stores to f check type

Property Type Inference



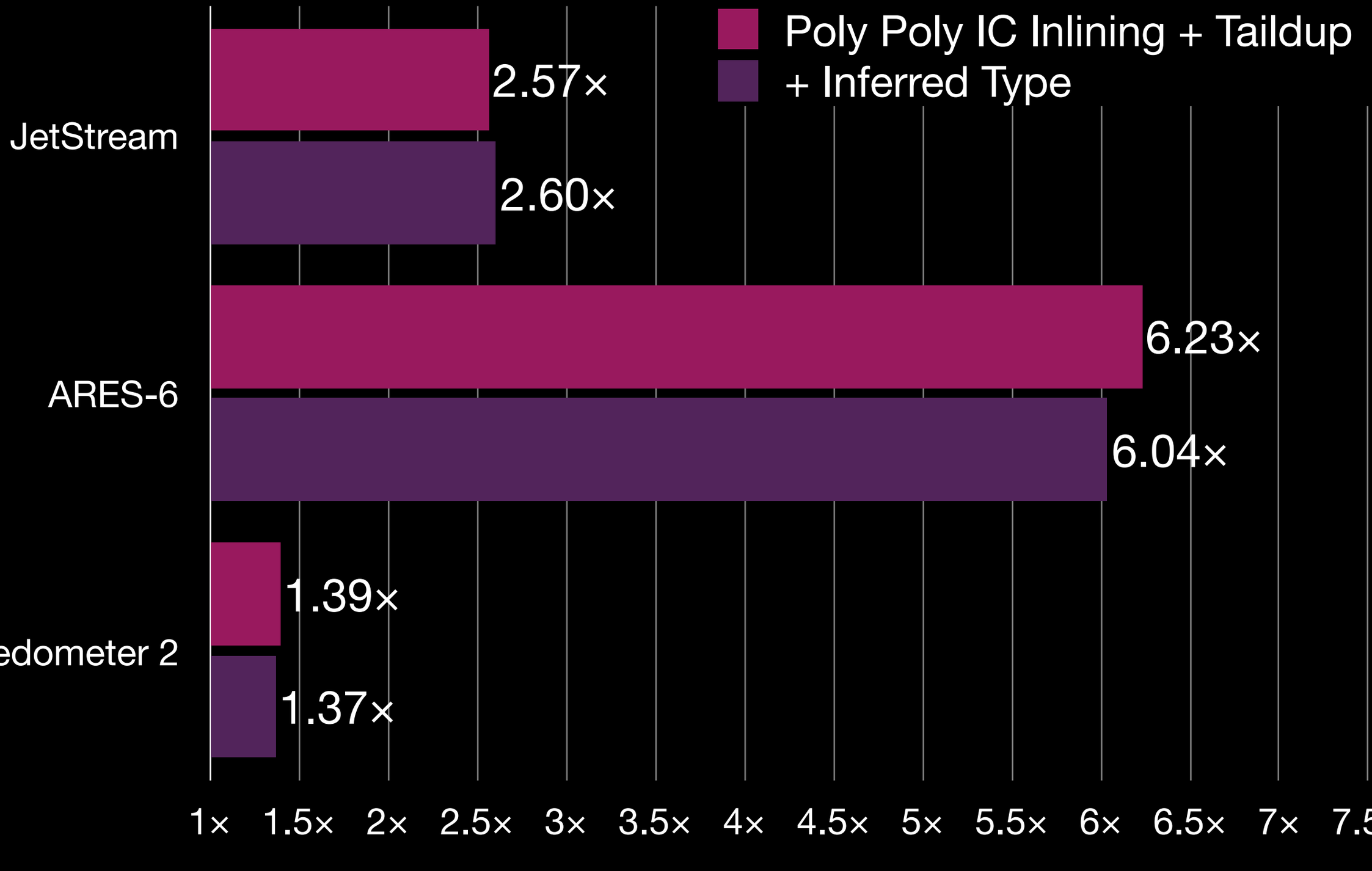
- Stores to f check type
- Loads don’t have to

Property Type Inference

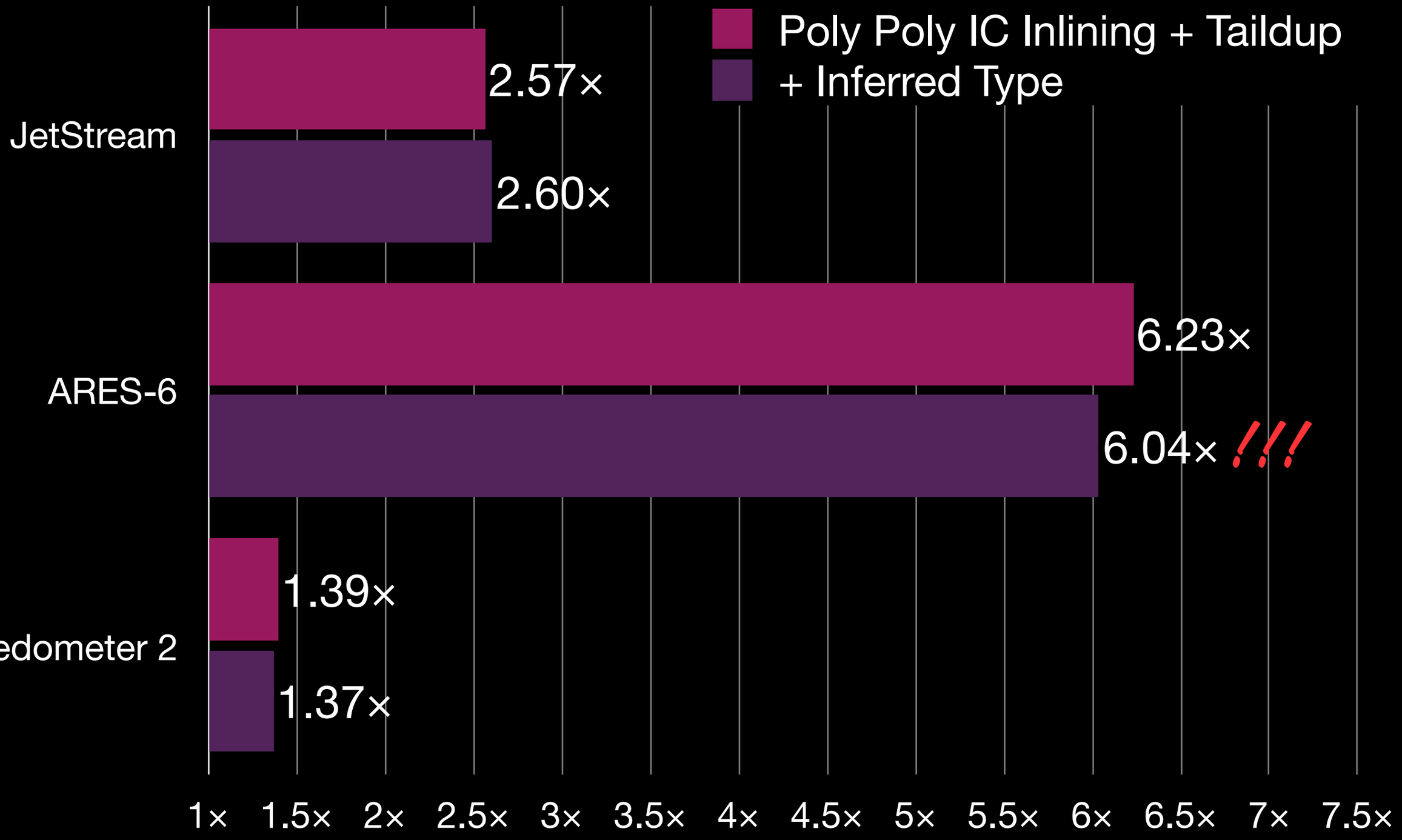


- Stores to f check type
- Loads don’t have to
- S1’s inferred type table watches S2’s transitions

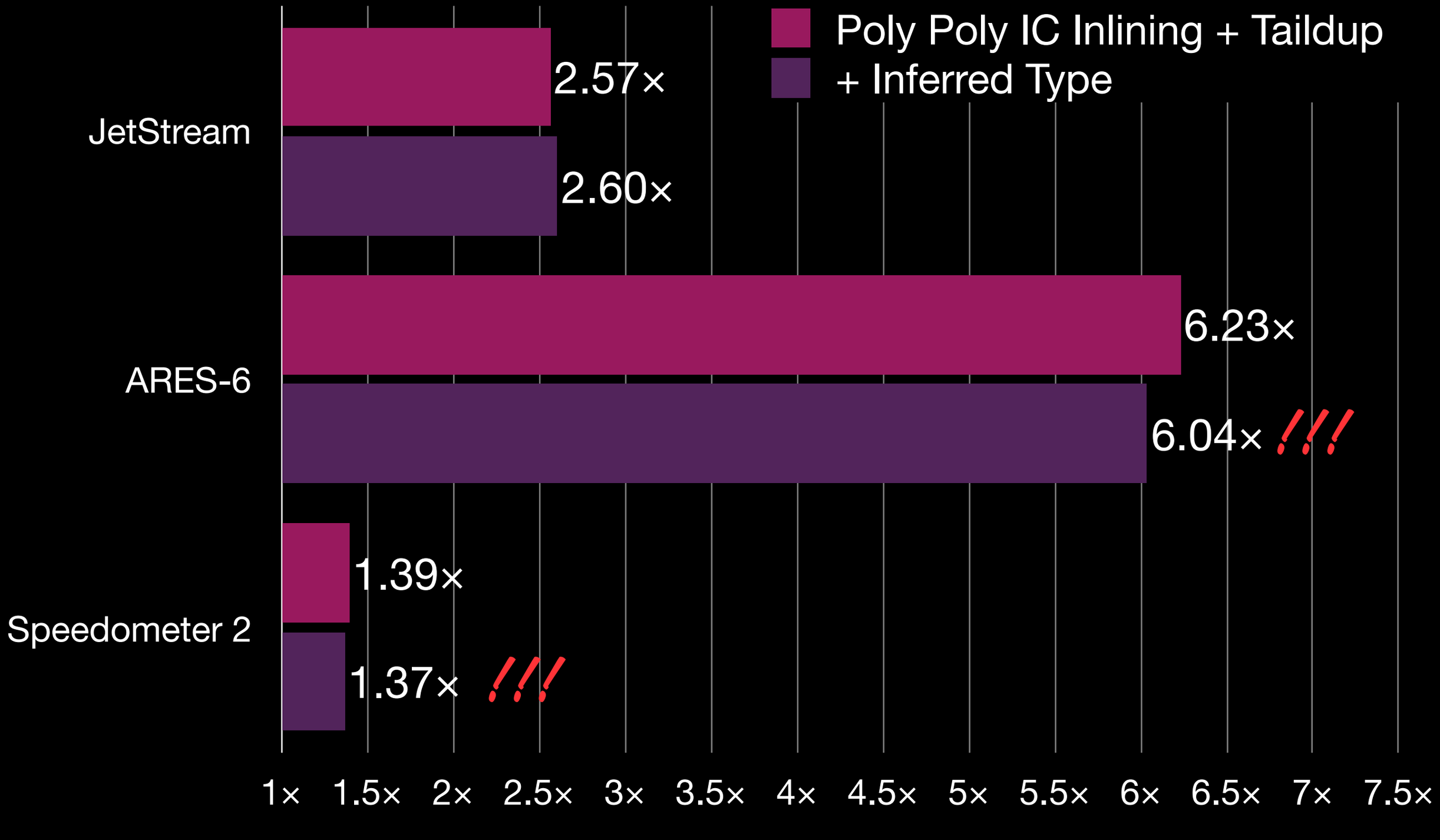
Inferred Type “Speed-up”



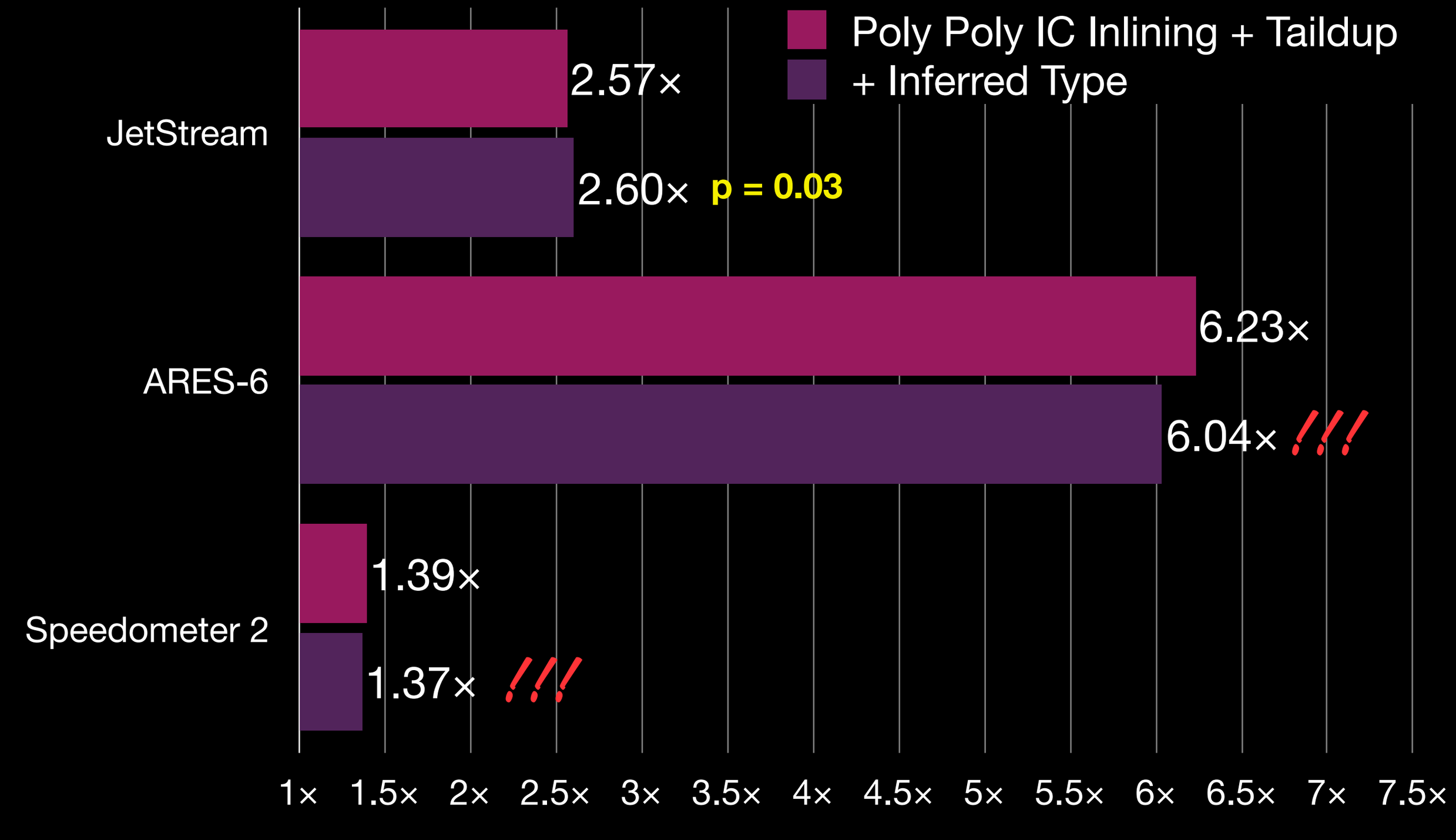
Inferred Type “Speed-up”



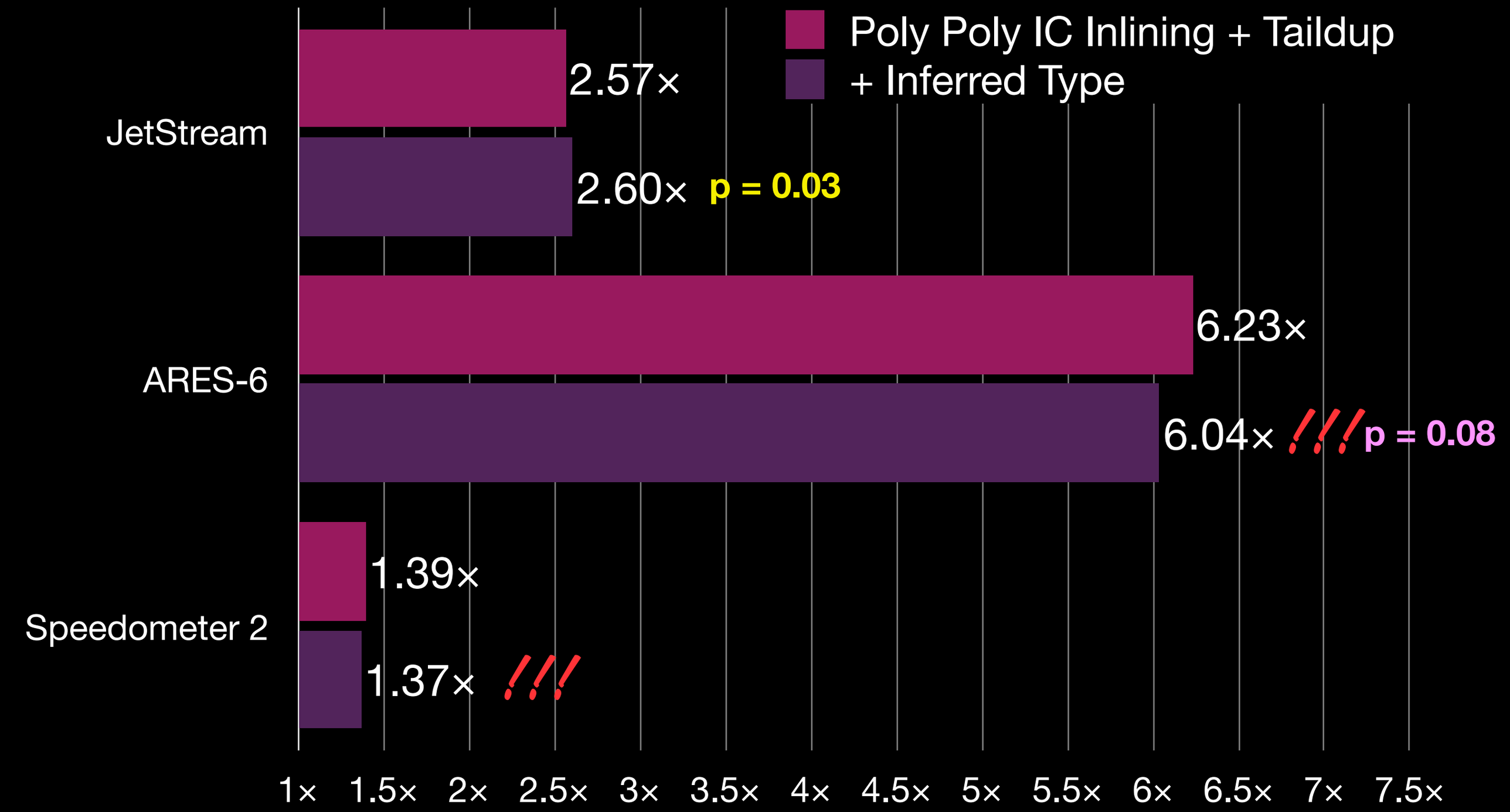
Inferred Type “Speed-up”



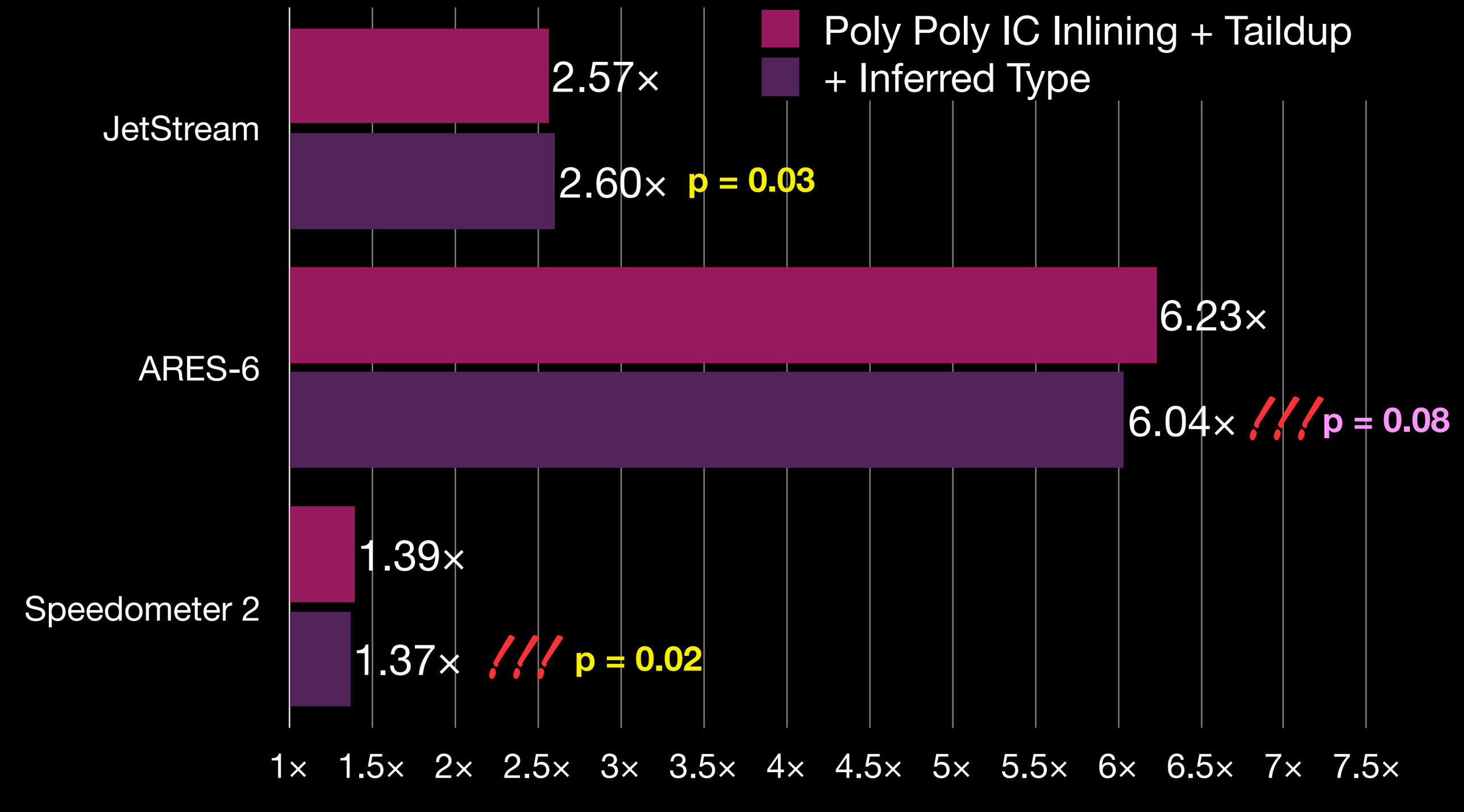
Inferred Type “Speed-up”



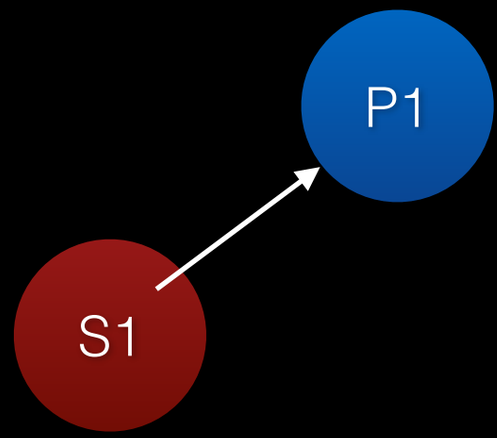
Inferred Type “Speed-up”



Inferred Type “Speed-up”

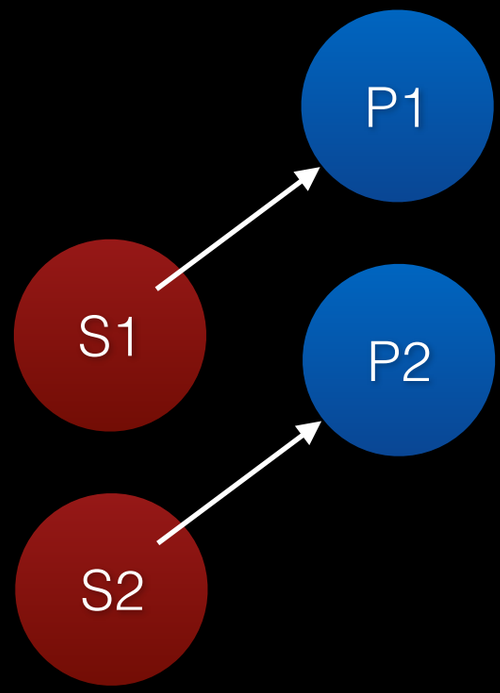


```
function foo()
{
    class Helper {
        …
    }
    var h = new Helper();
    …
}
```

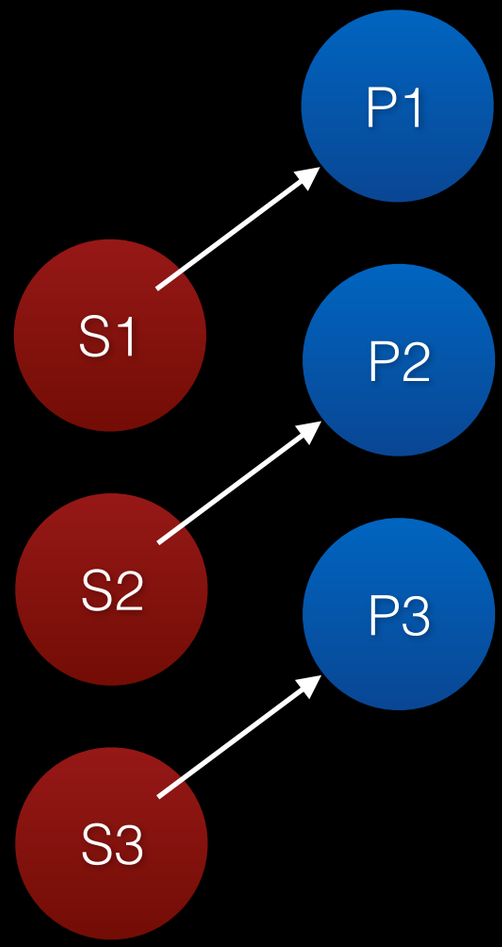



```
function foo()
{
    class Helper {
        …
    }
    var h = new Helper();
    …
}
```

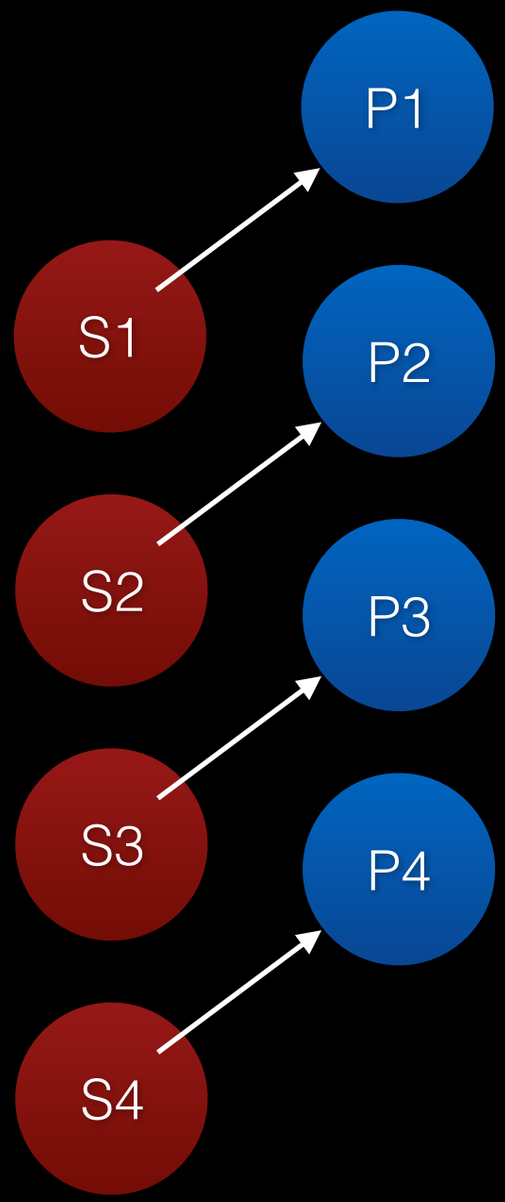
```
function foo()
{
    class Helper {
        …
    }
    var h = new Helper();
    …
}
```



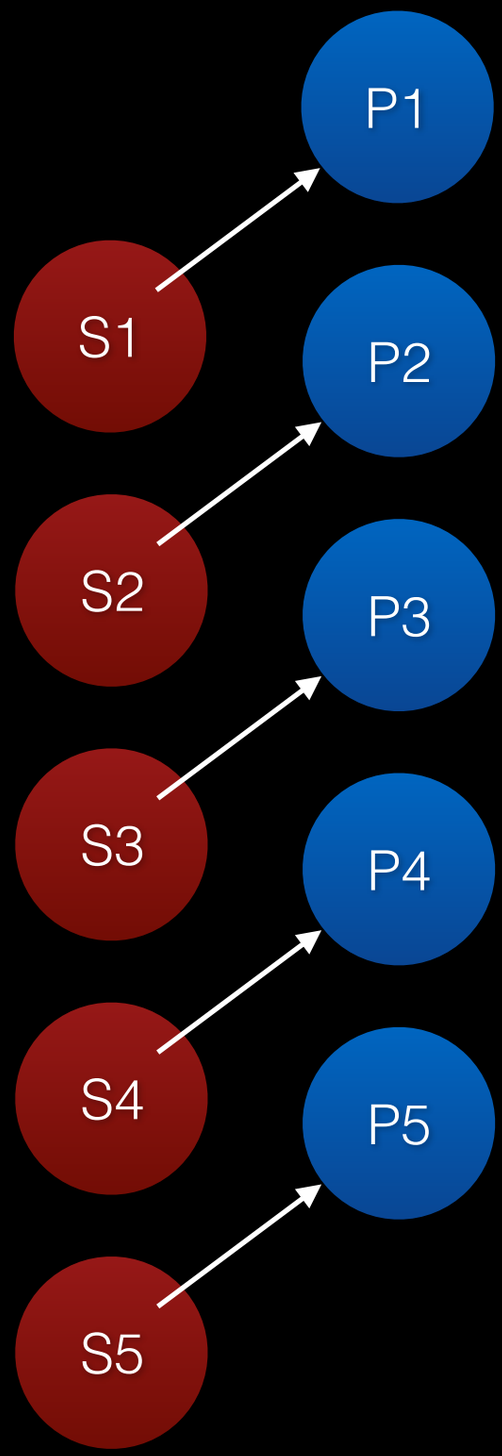
```
function foo()
{
    class Helper {
        …
    }
    var h = new Helper();
    …
}
```



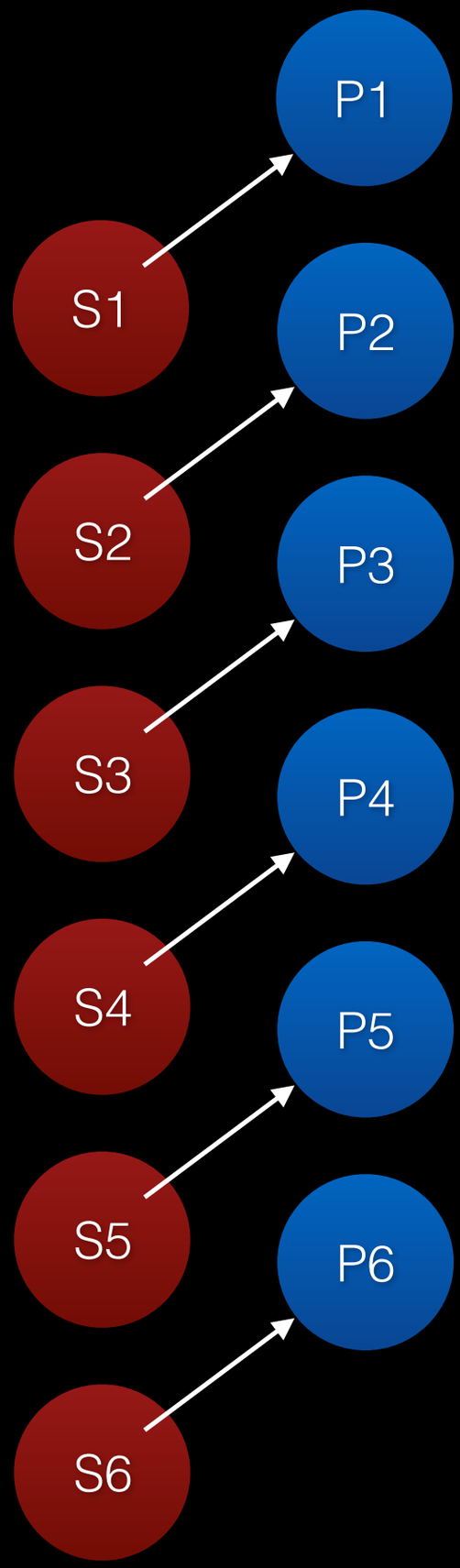
```
function foo()
{
    class Helper {
        …
    }
    var h = new Helper();
    …
}
```



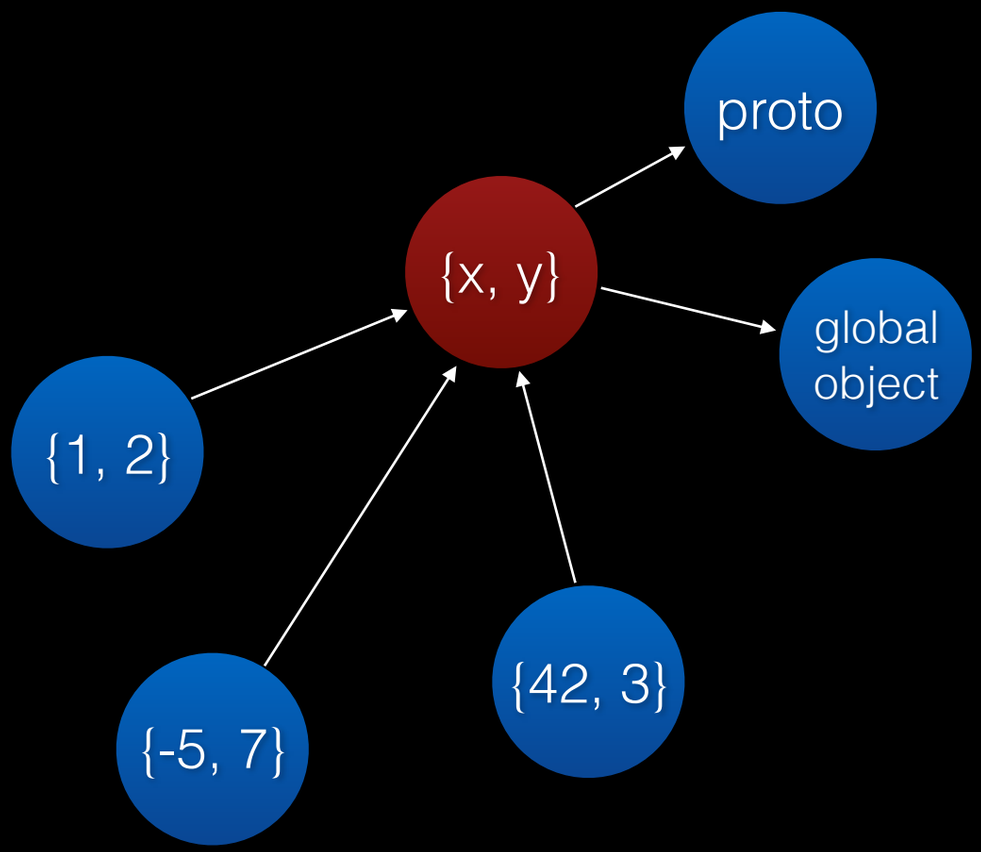
```
function foo()
{
    class Helper {
        …
    }
    var h = new Helper();
    …
}
```



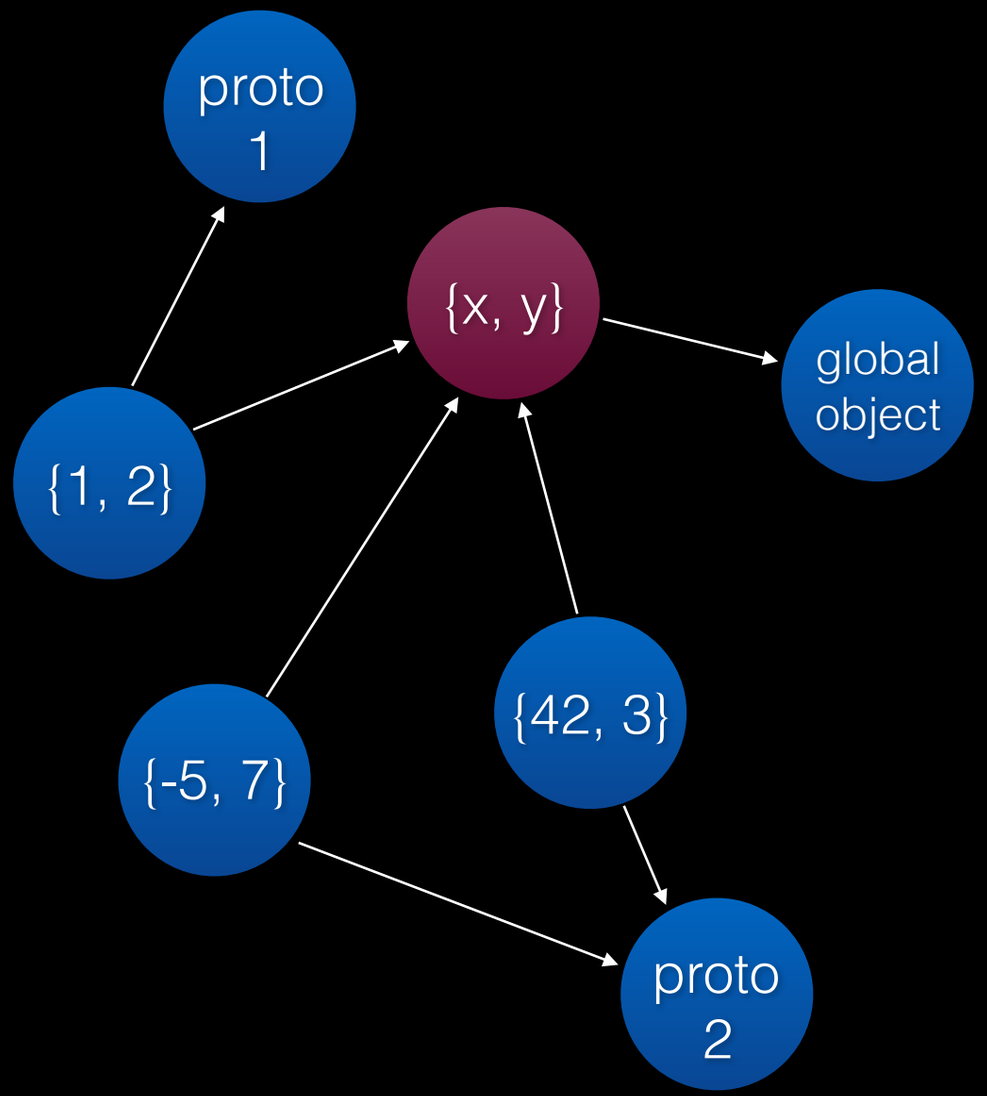
```
function foo()
{
    class Helper {
        …
    }
    var h = new Helper();
    …
}
```



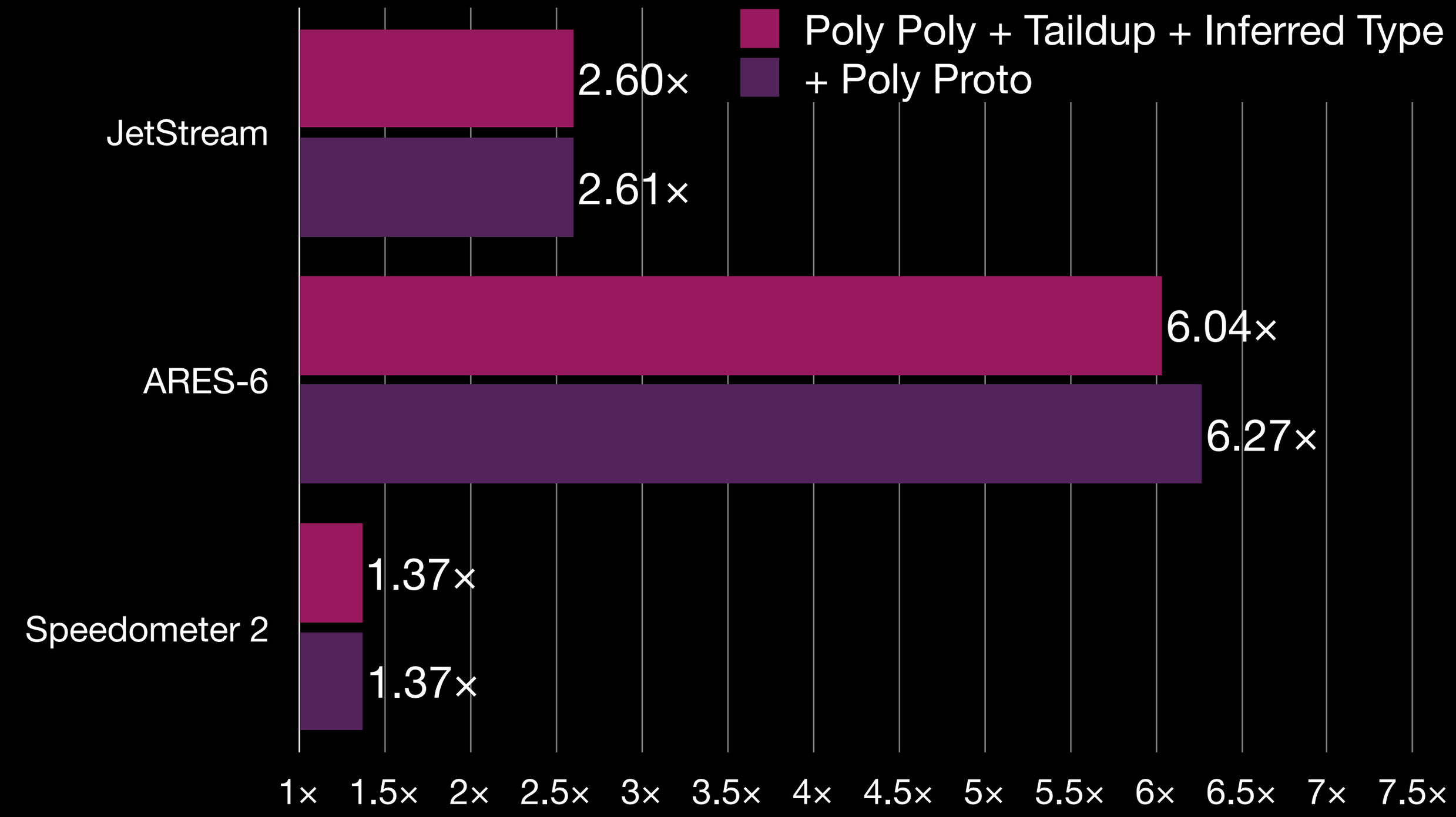
Mono Proto



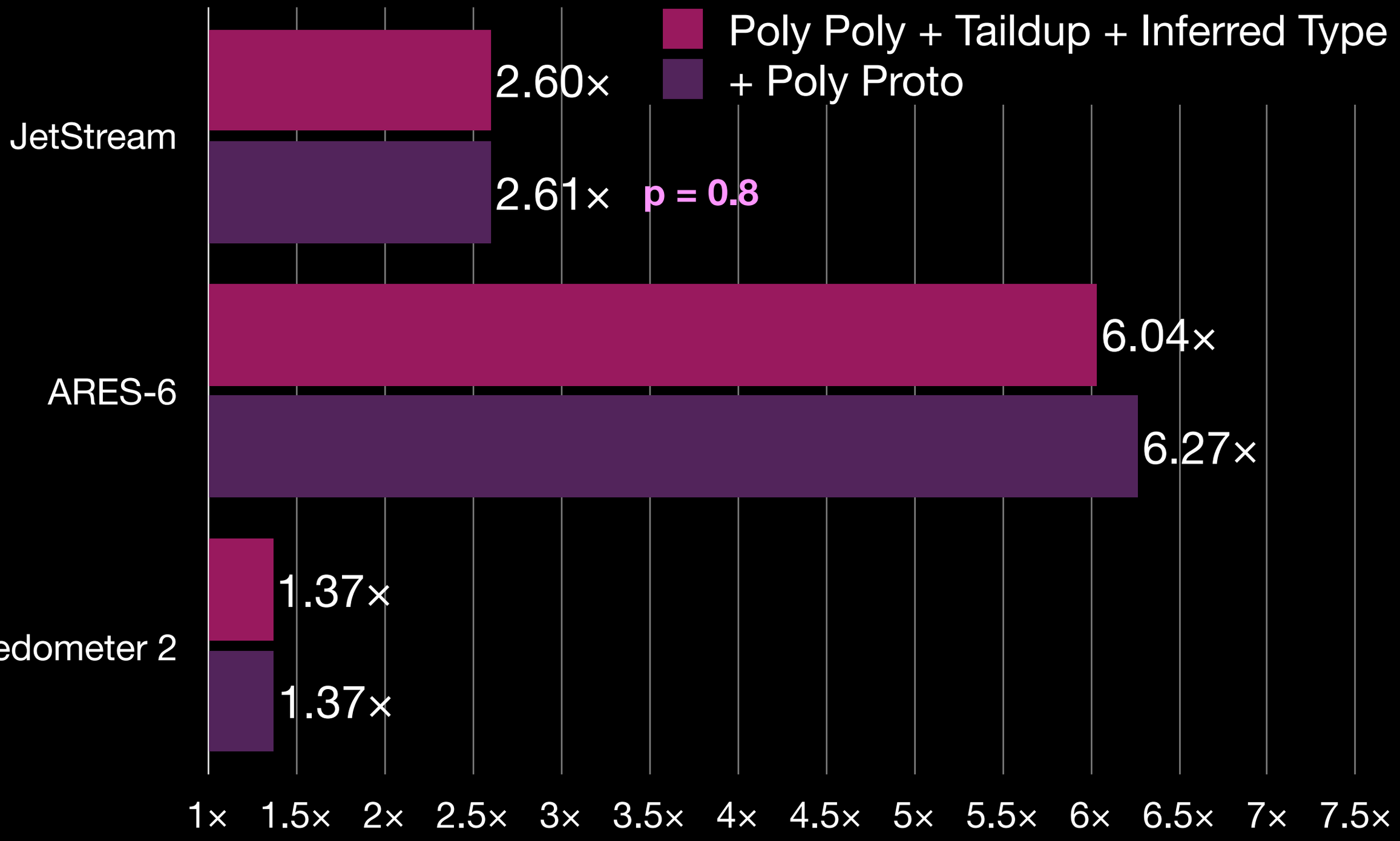
Poly Proto



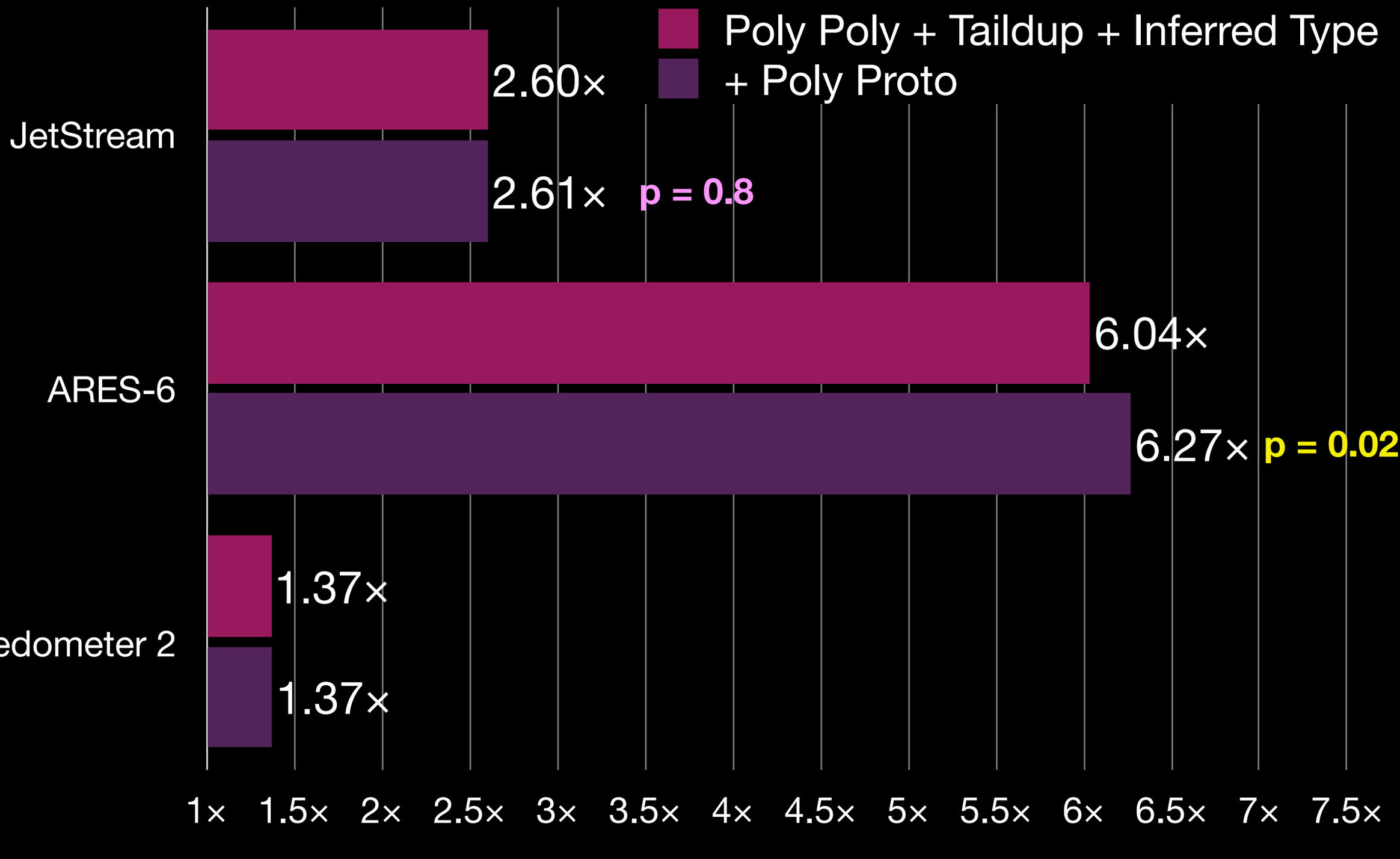
Poly Proto Speed-up



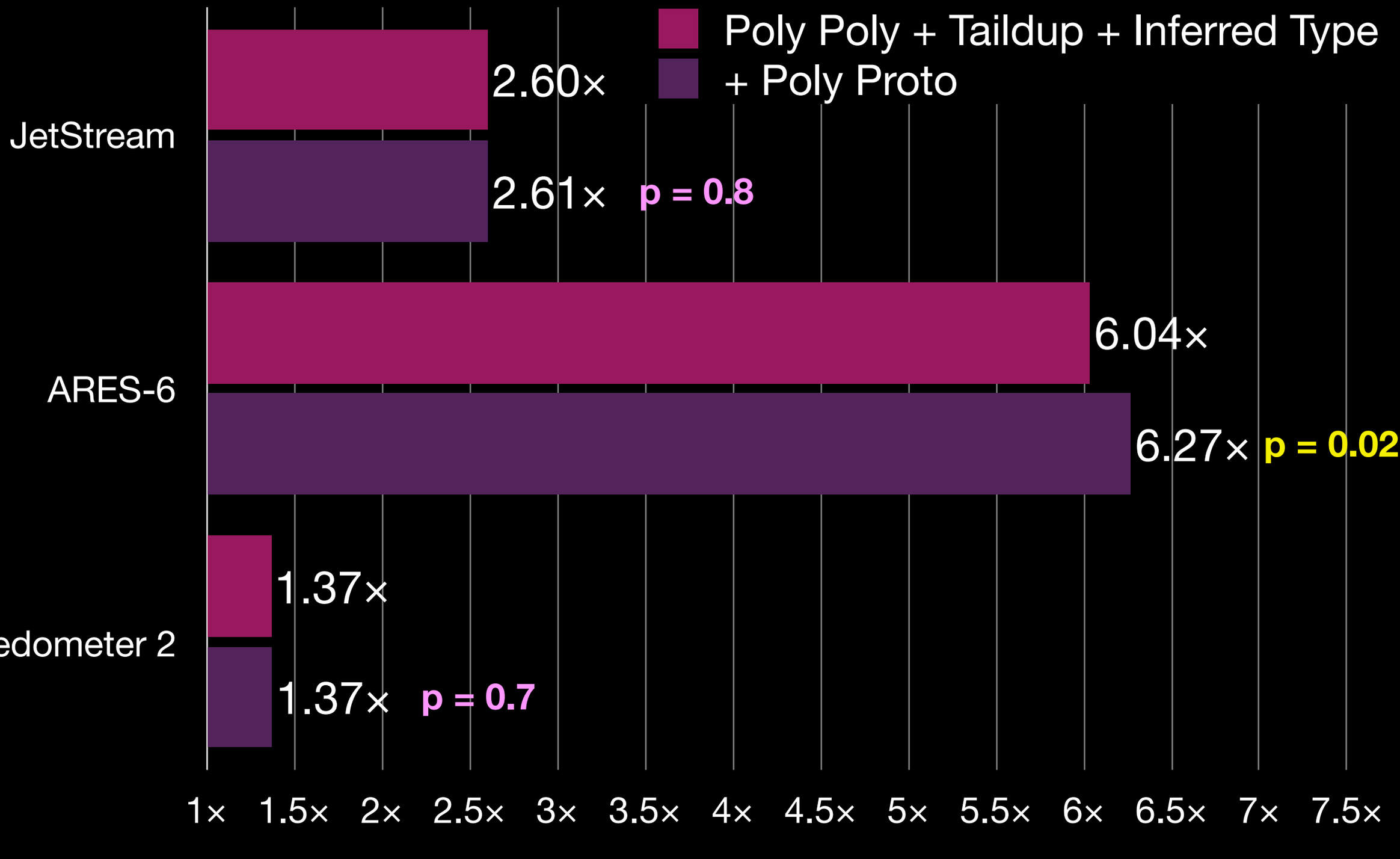
Poly Proto Speed-up



Poly Proto Speed-up

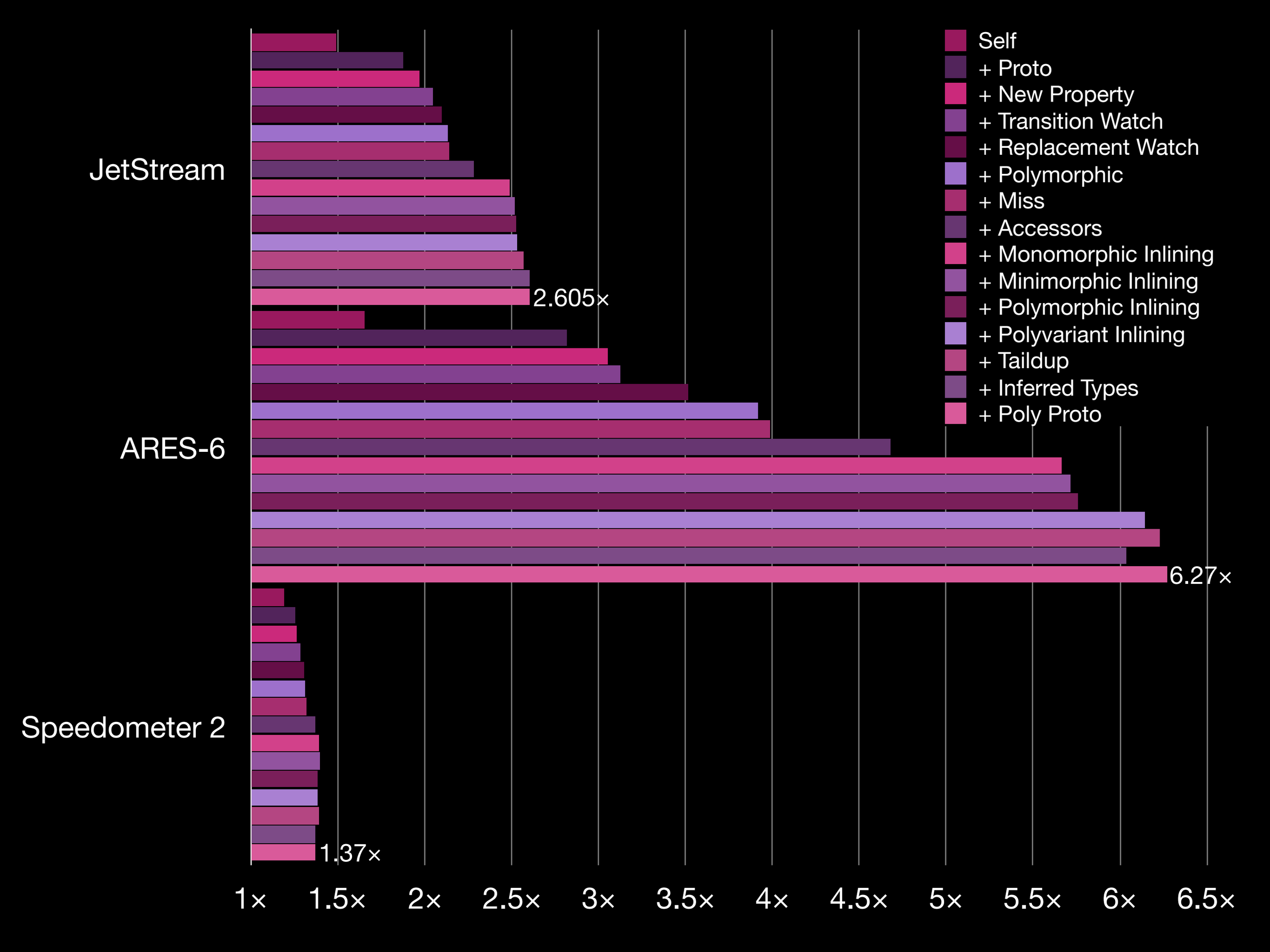


Poly Proto Speed-up



How I Ran These Experiments

- WebKit trunk revision 233406
- Safari trunk hash dd8296dbaac7afa5ed9a699aa261033ea5f5577c
- macOS Internal SDK
- make release (not a root, no shared cache)
- Patch, scripts, and raw data at webkit.org/b/187414
- MacBookPro11,5 2.8GHz 16GB RAM 1TB SSD


JetStream
ARES-6
Speedometer 2
2.605×
6.27×
1.37×
Self
+ Proto
+ New Property
+ Transition Watch
+ Replacement Watch
+ Polymorphic
+ Miss
+ Accessors
+ Monomorphic Inlining
+ Minimorphic Inlining
+ Polymorphic Inlining
+ Polyvariant Inlining
+ Taildup
+ Inferred Types
+ Poly Proto
1×
1.5×
2×
2.5×
3×
3.5×
4×
4.5×
5×
5.5×
6×
6.5×