Low-Overhead Multi-Language Dynamic Taint Analysis on Managed Runtimes through Speculative Optimization

Jacob Kreindl[*], Daniele Bonetta[†], David Leopoldseder[†], Lukas Stadler[†], Hanspeter Mössenböck[*]

[*] JKU Institute for System Software, [†] Oracle Labs

JOHANNES KEPLER UNIVERSITY LINZ

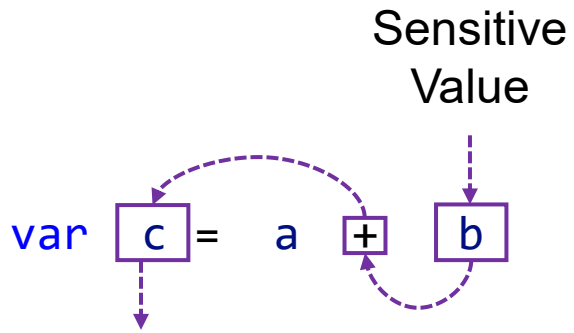
SSW

ORACLE Labs

Dynamic Taint Analysis

```
var  c =  a  +   b
```

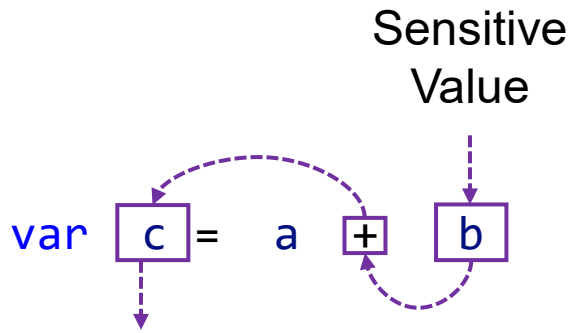
Dynamic Taint Analysis




```
taint(c) = taint(a) | taint(b)
```

Dynamic Taint Analysis

At Runtime




```
taint(c) = taint(a) | taint(b)
```

GraalVM™

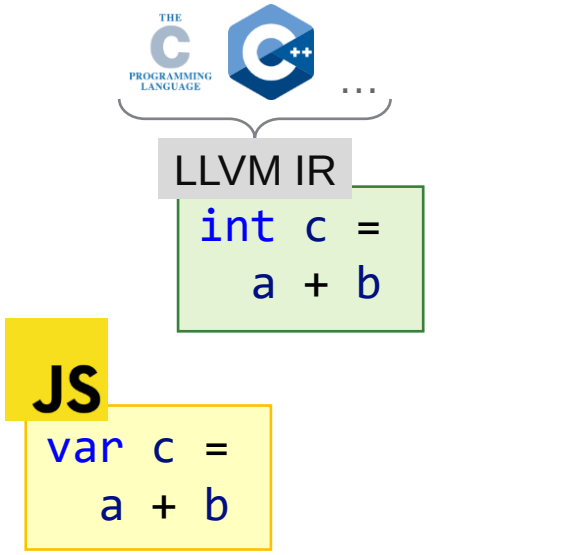
```
var c =
  a + b
```

GraalVM™

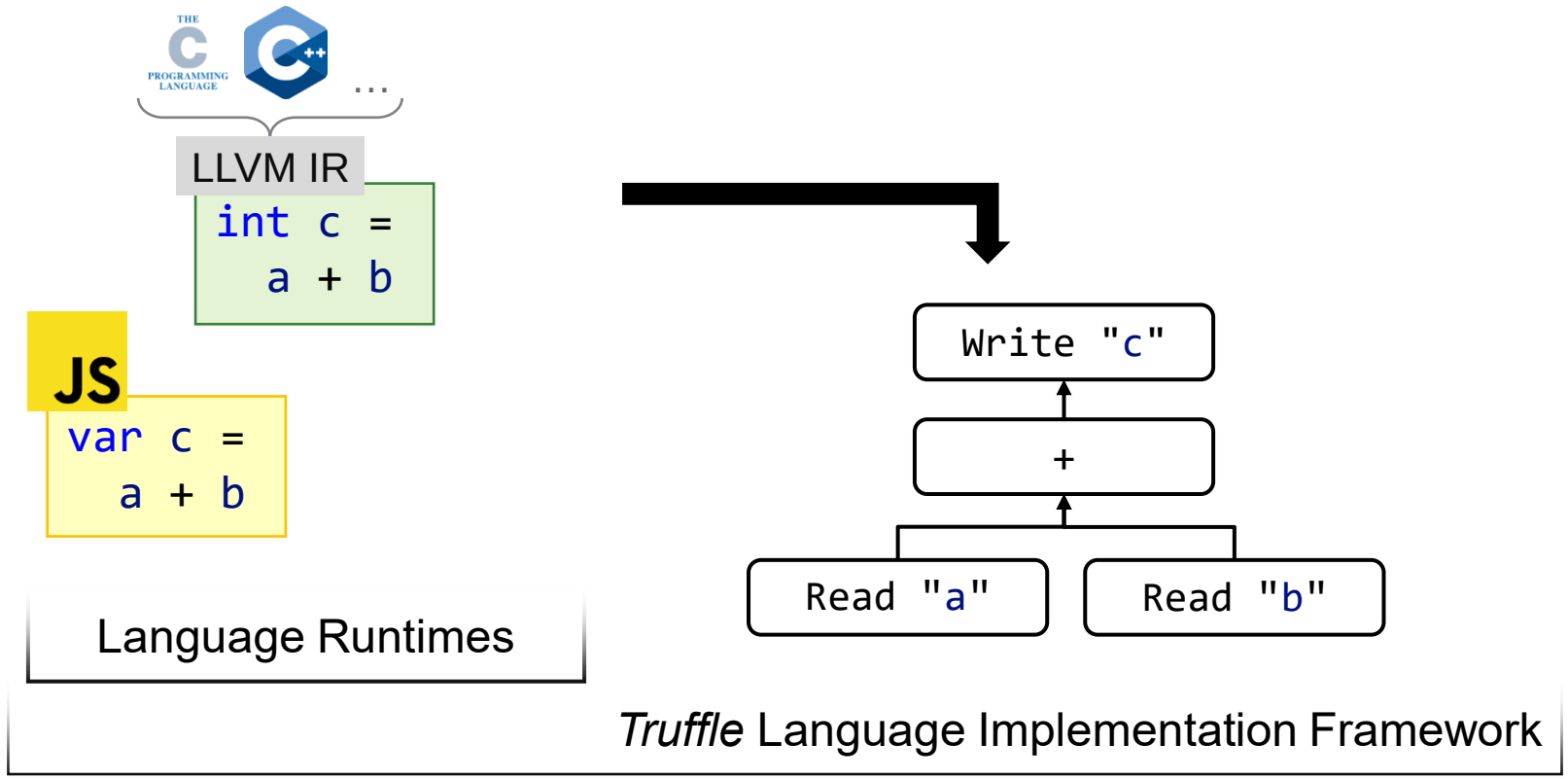
JS

```
var c =
  a + b
```

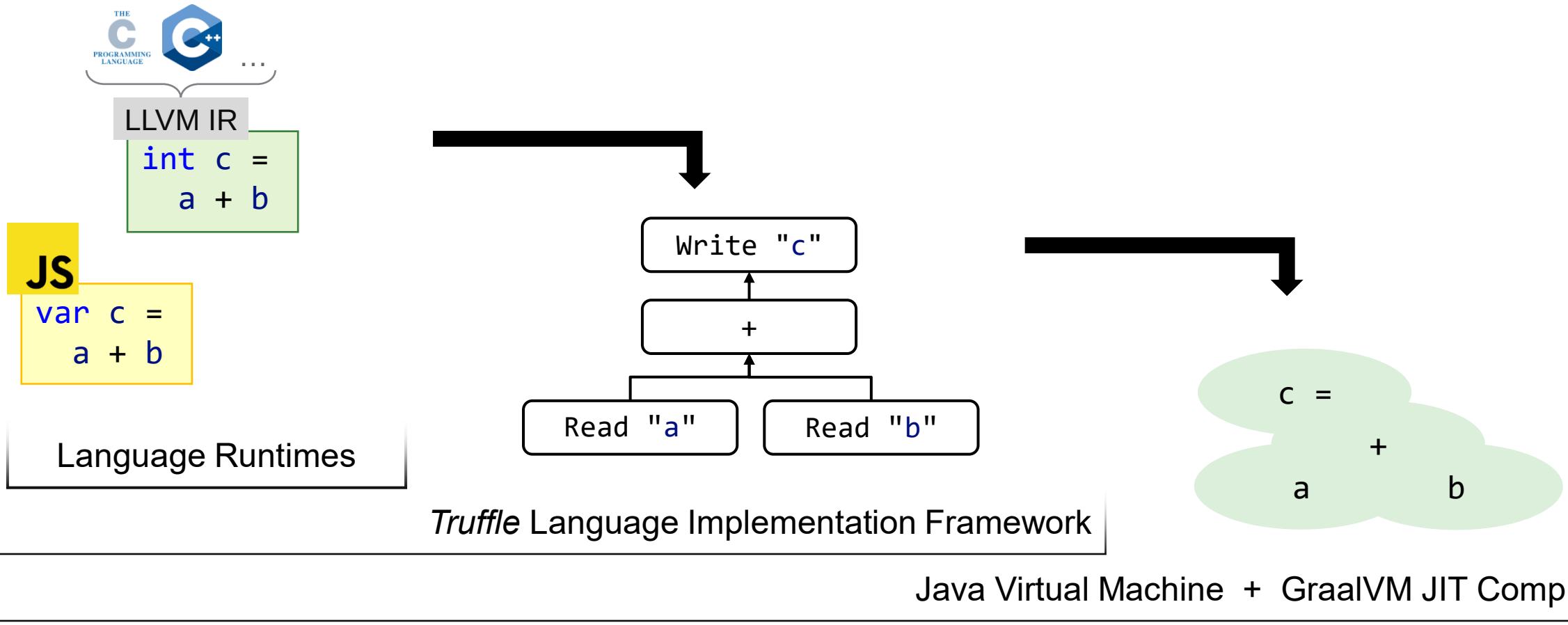
GraalVM™


THE C PROGRAMMING LANGUAGE
C++
...
LLVM IR
int c =
a + b
JS
var c =
a + b
Language Runtimes

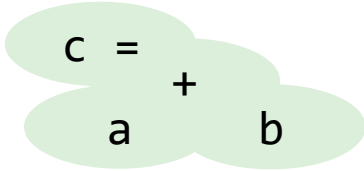
GraalVM™

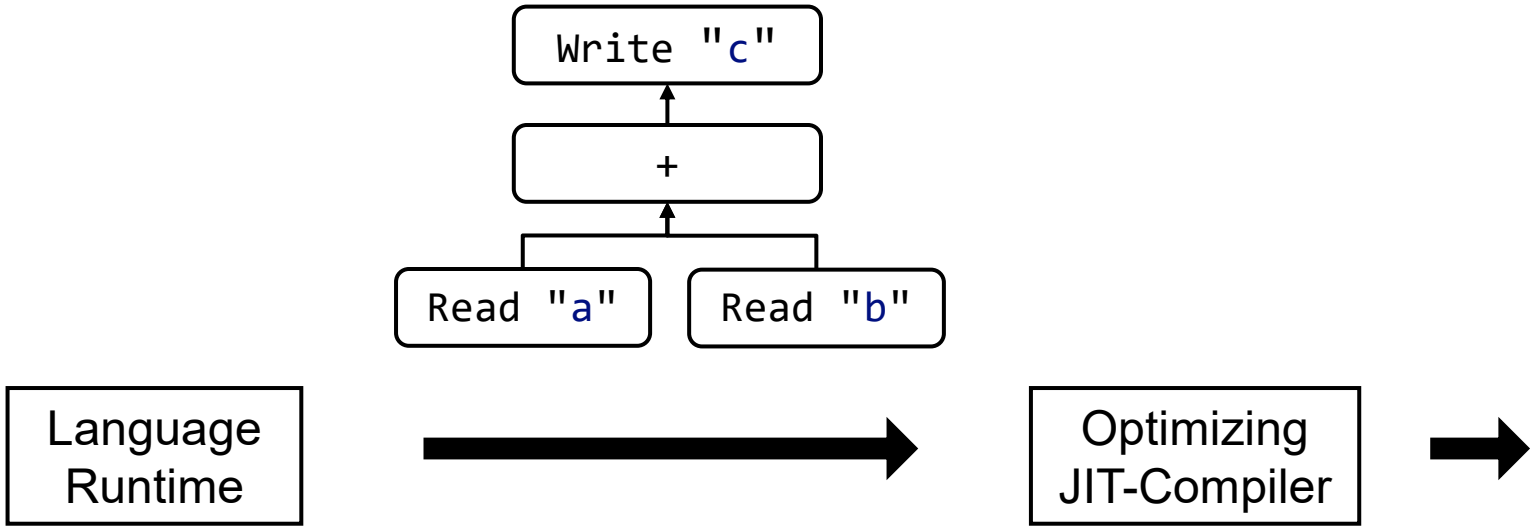

THE C PROGRAMMING LANGUAGE
C++
...
LLVM IR
int c =
a + b
JS
var c =
a + b
Language Runtimes
Write "c"
+
Read "a"
Read "b"
Truffle Language Implementation Framework

GraalVM™

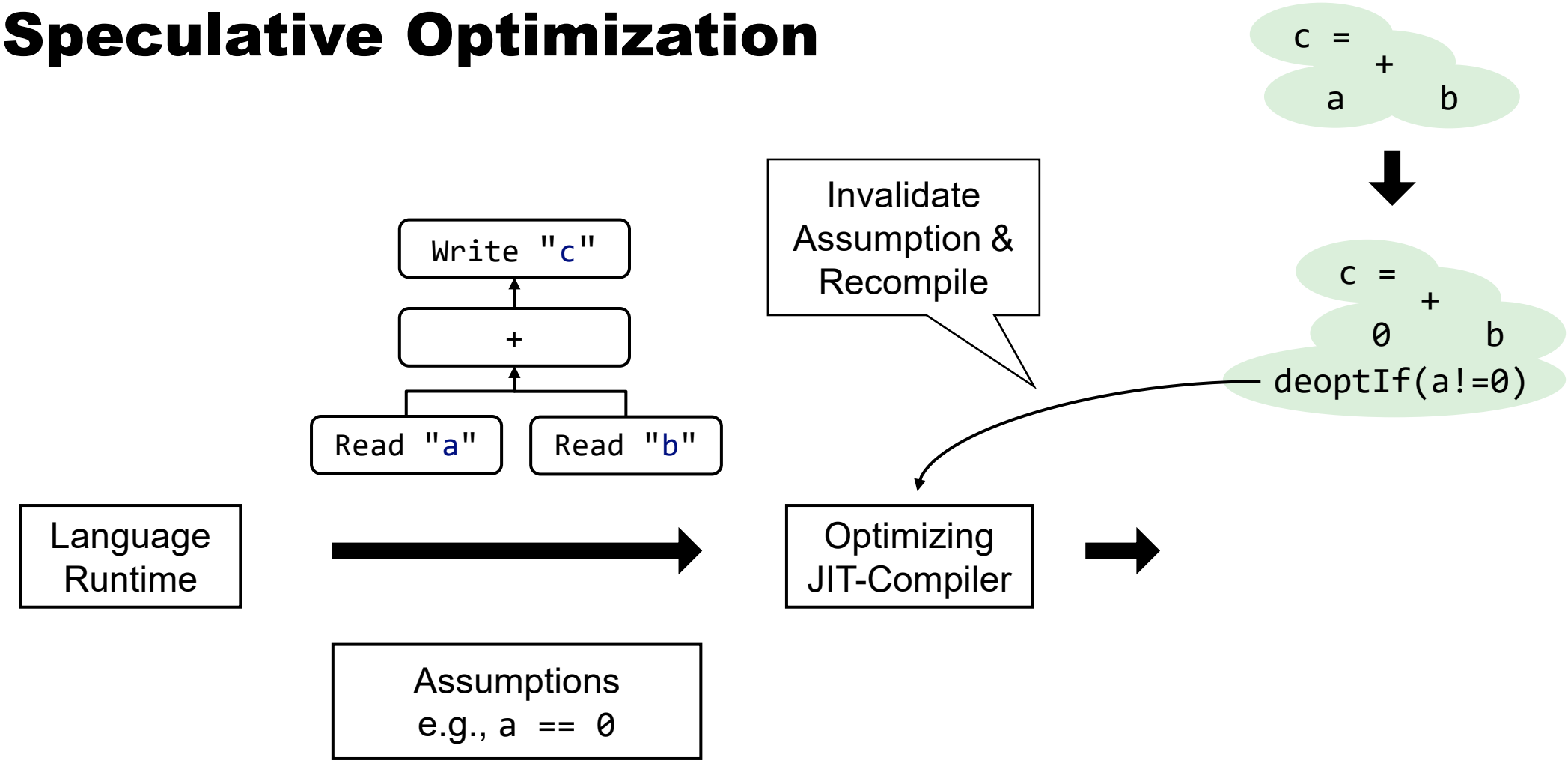

LLVM IR
int c =
a + b
JS
var c =
a + b
Language Runtimes
Write "c"
+
Read "a"
Read "b"
c =
+
a
b
Truffle Language Implementation Framework
Java Virtual Machine + GraalVM JIT Compiler

Speculative Optimization

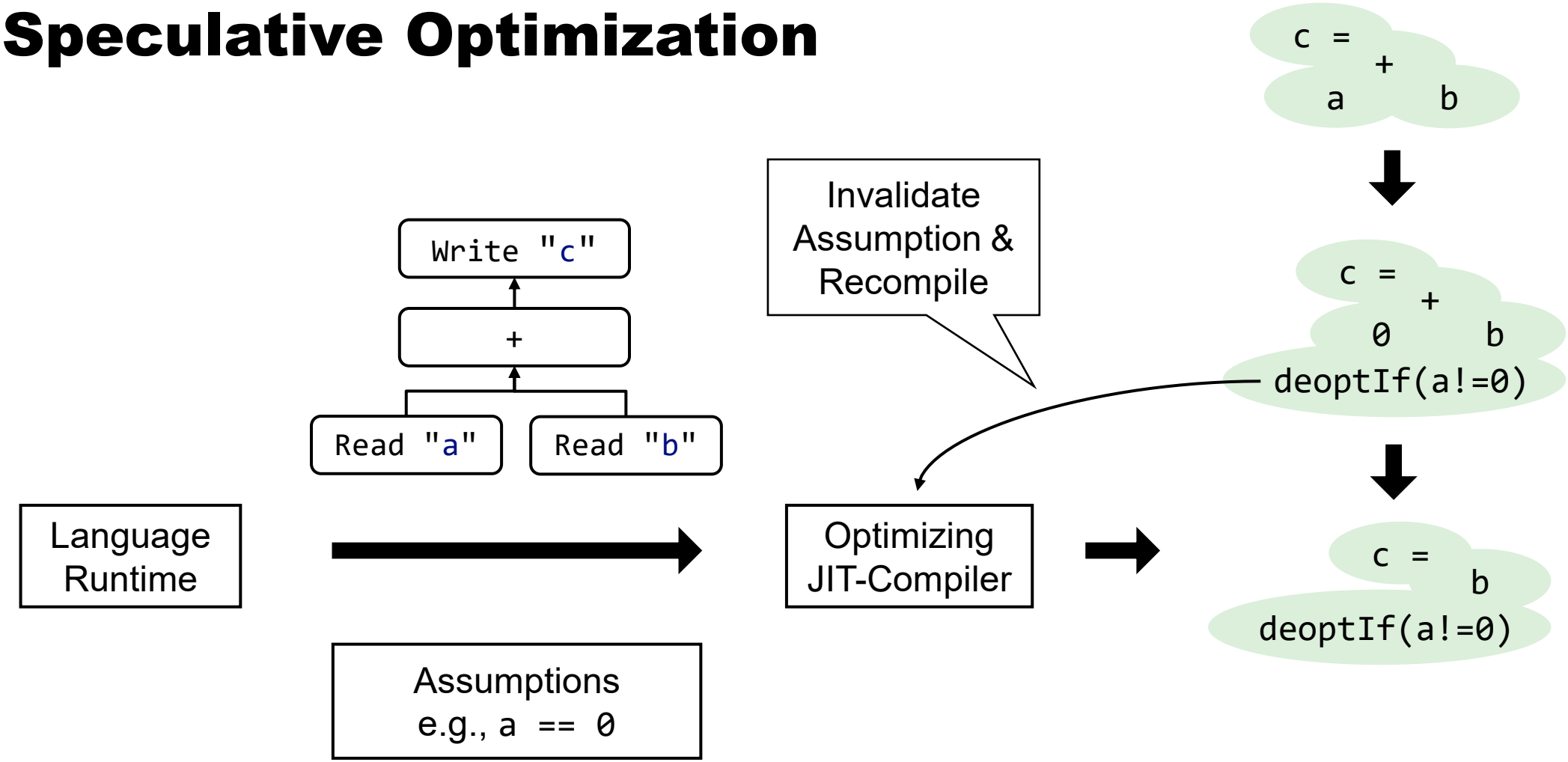




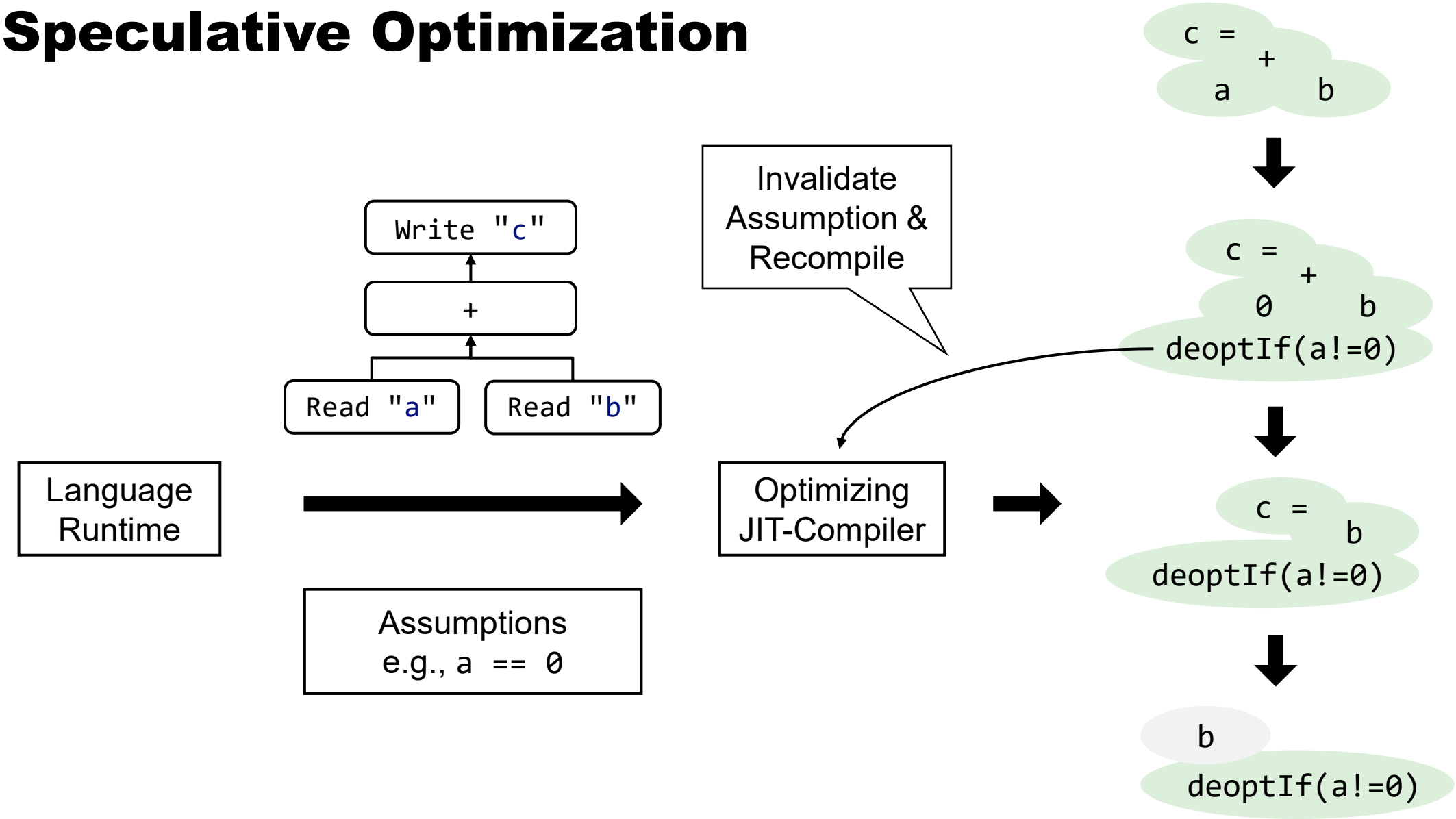
Speculative Optimization



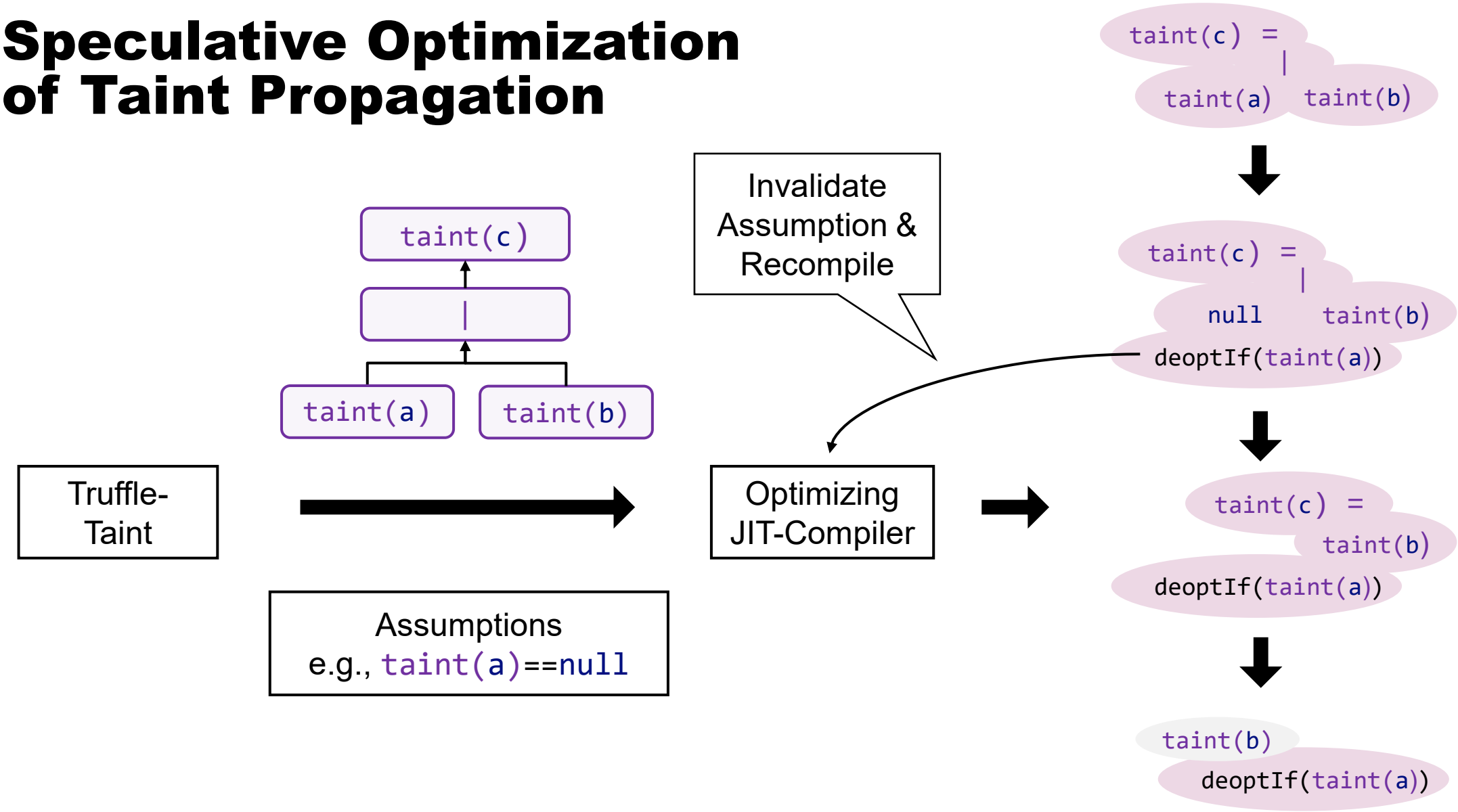
Speculative Optimization



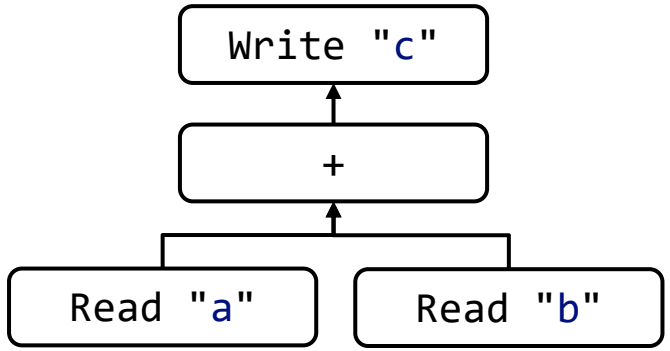
Speculative Optimization



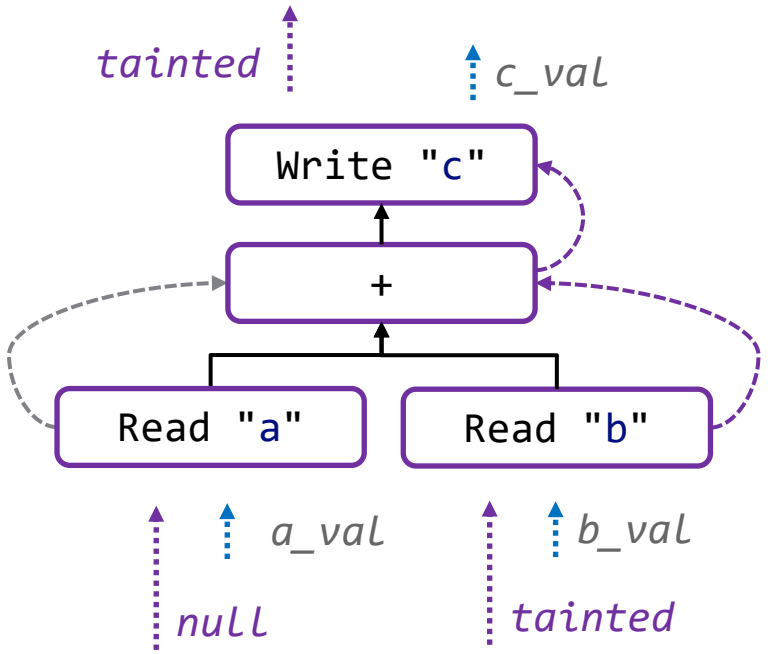
Speculative Optimization of Taint Propagation



TruffleTaint



TruffleTaint



Shadow Storage

- Local / Global Variables

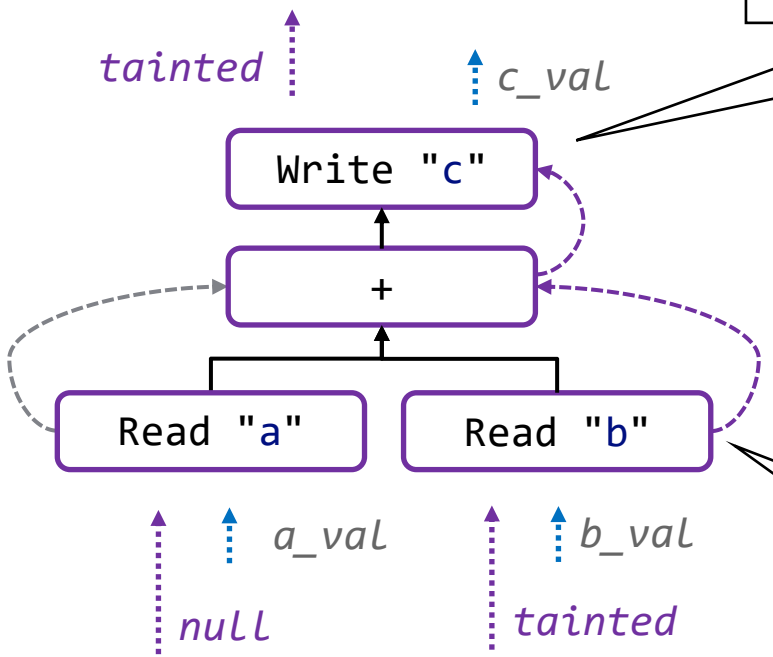
```
» "b": 1
» shadow["b"]: <label>
  ...
```

- Heap Memory

```
» data:
» shadow[data]:
  ...
```

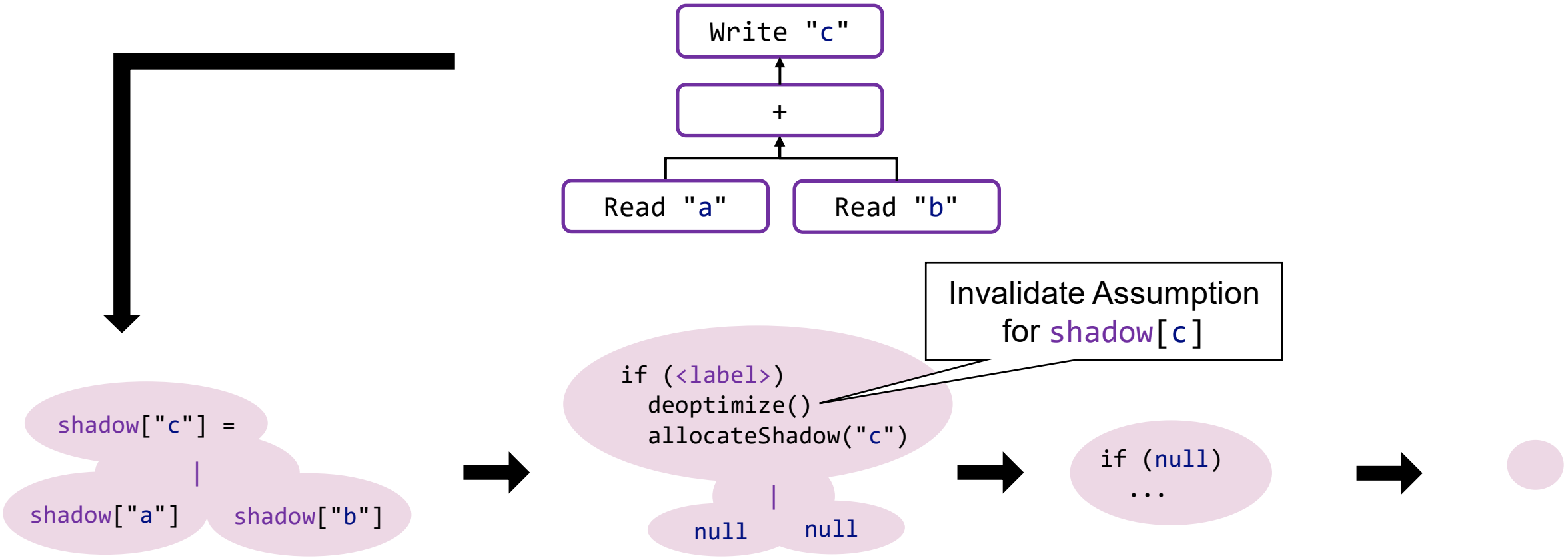
Optimization: On-Demand Shadow Storage

```
if (hasShadow("c"))
    updateShadow("c", <label>)
else if (<label>)
    allocateShadow("c", <label>)
```

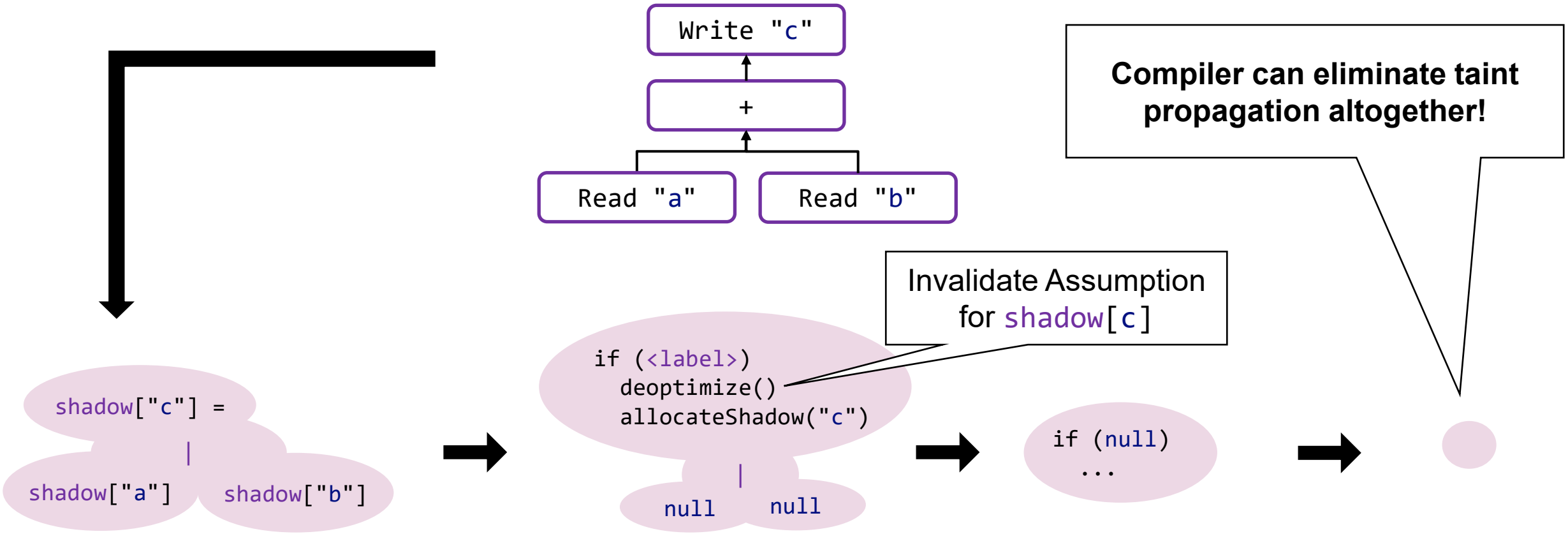



```
if (hasShadow("b"))
    label = readShadow("b")
else
    label = null
```

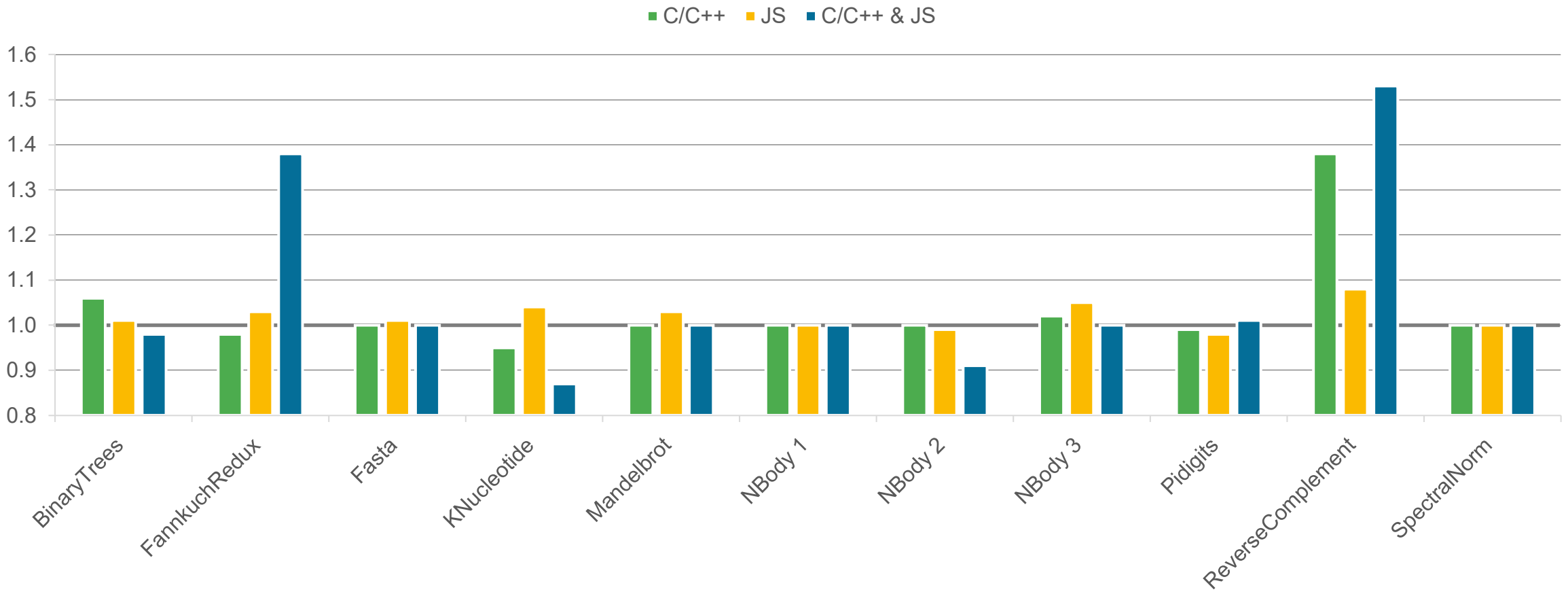
Optimization: Assume shadow[x] not yet allocated

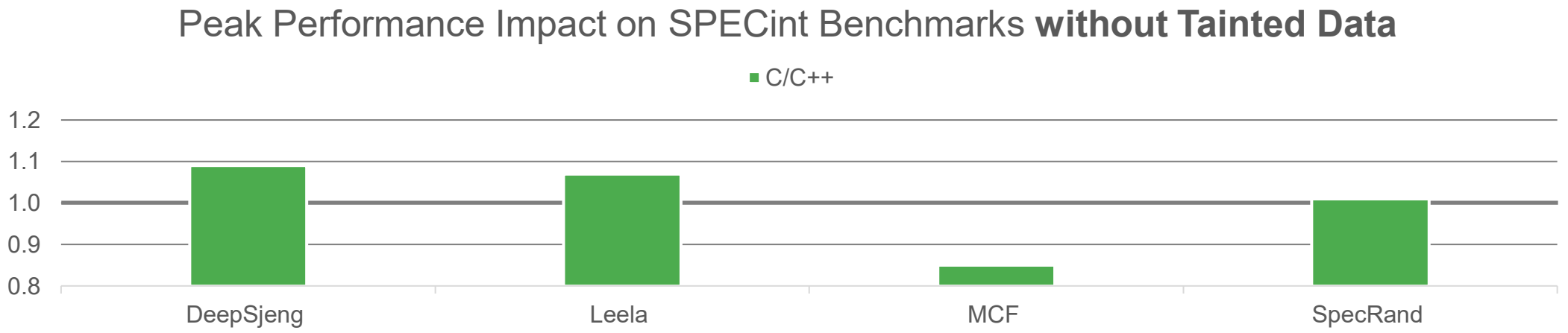


Optimization: Assume shadow[x] not yet allocated

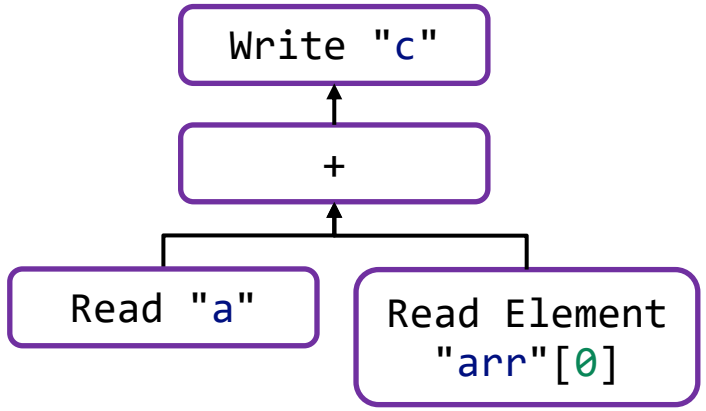


Peak Performance Impact on Shootouts Benchmarks **without Tainted Data**

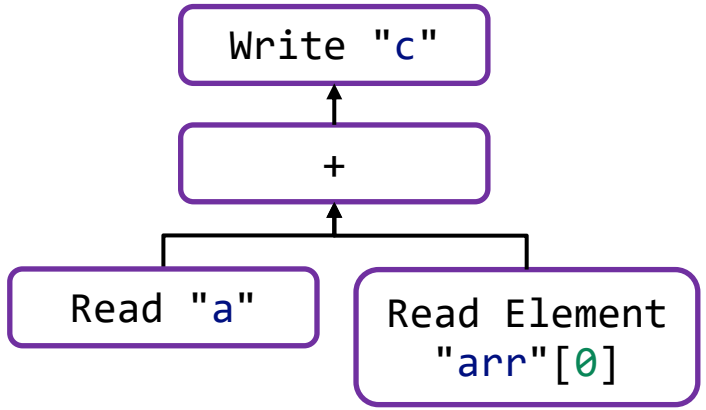



Peak Performance Impact on SPECint Benchmarks **without Tainted Data**
C/C++
1.2
1.1
1.0
0.9
0.8
DeepSjeng
Leela
MCF
SpecRand

Optimization: Profiling Shadow Storage Presence

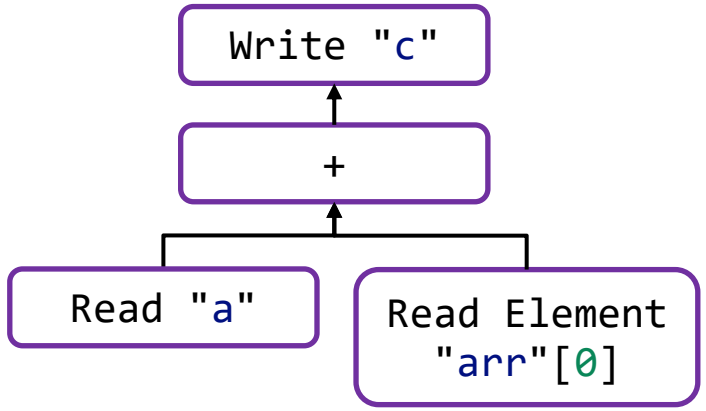


Optimization: Profiling Shadow Storage Presence

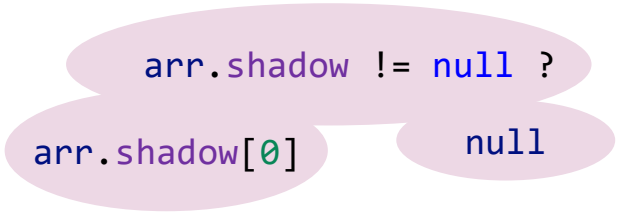



```
if (profiled(arr.shadow != null))
    label = arr.shadow[0]
else
    label = null
```

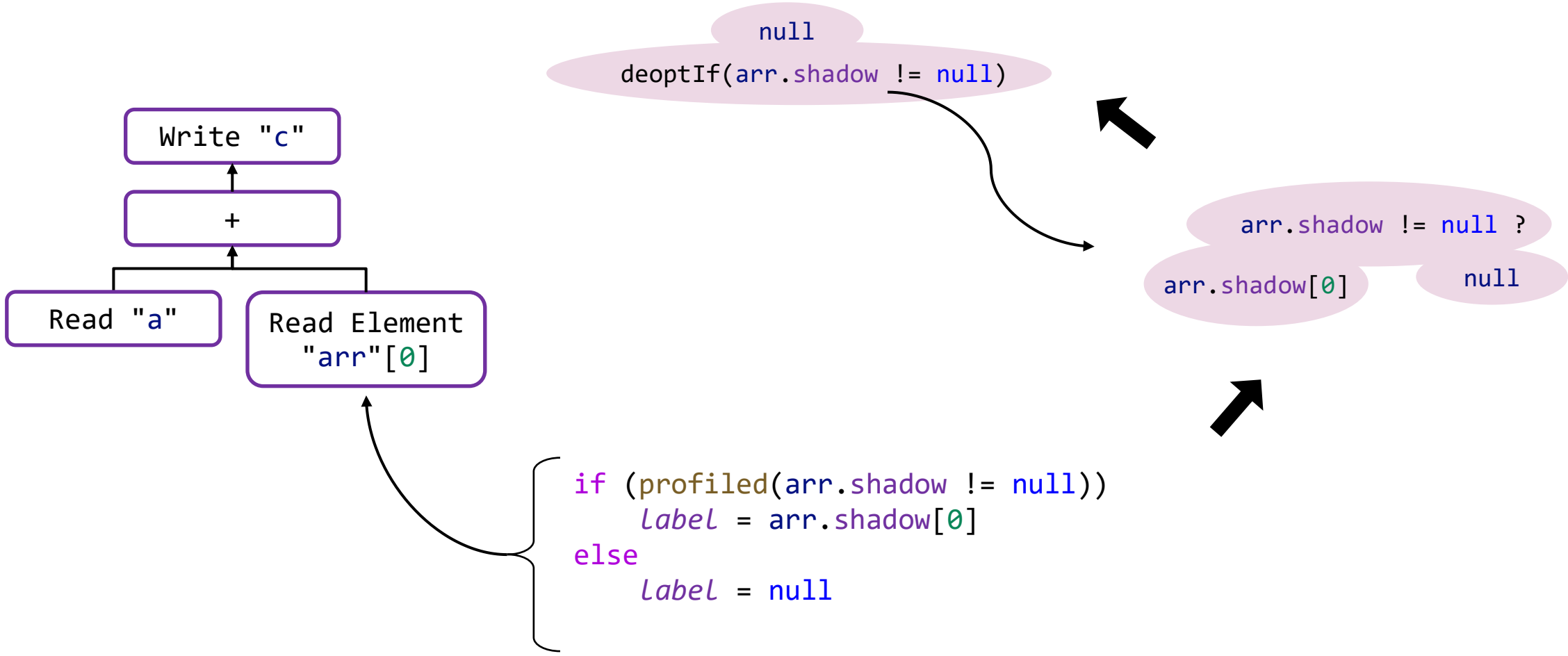
Optimization: Profiling Shadow Storage Presence




```
if (profiled(arr.shadow != null))
    label = arr.shadow[0]
else
    label = null
```

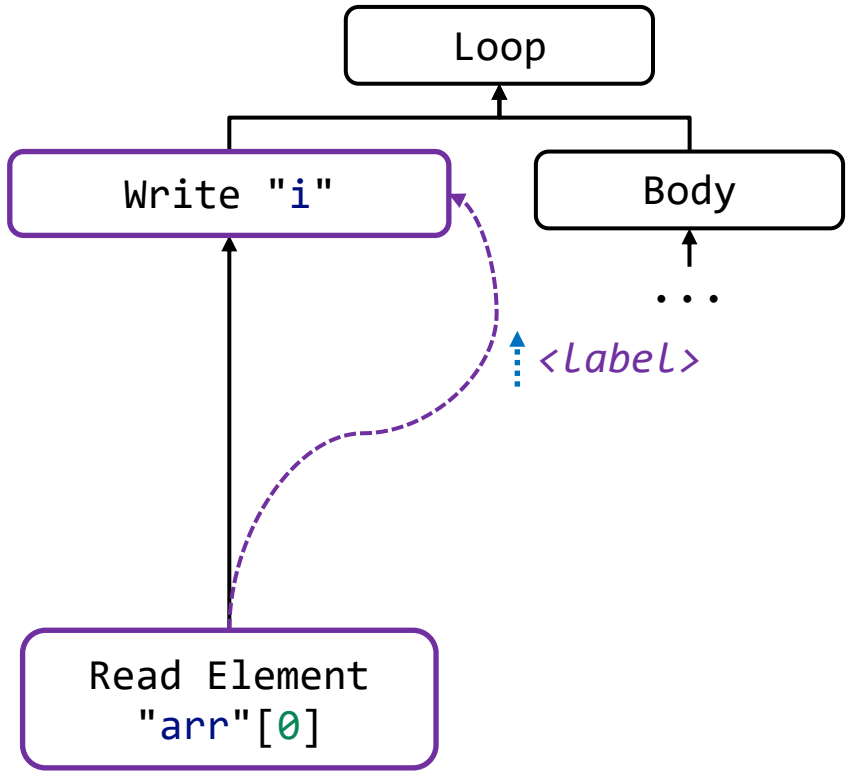


Optimization: Profiling Shadow Storage Presence



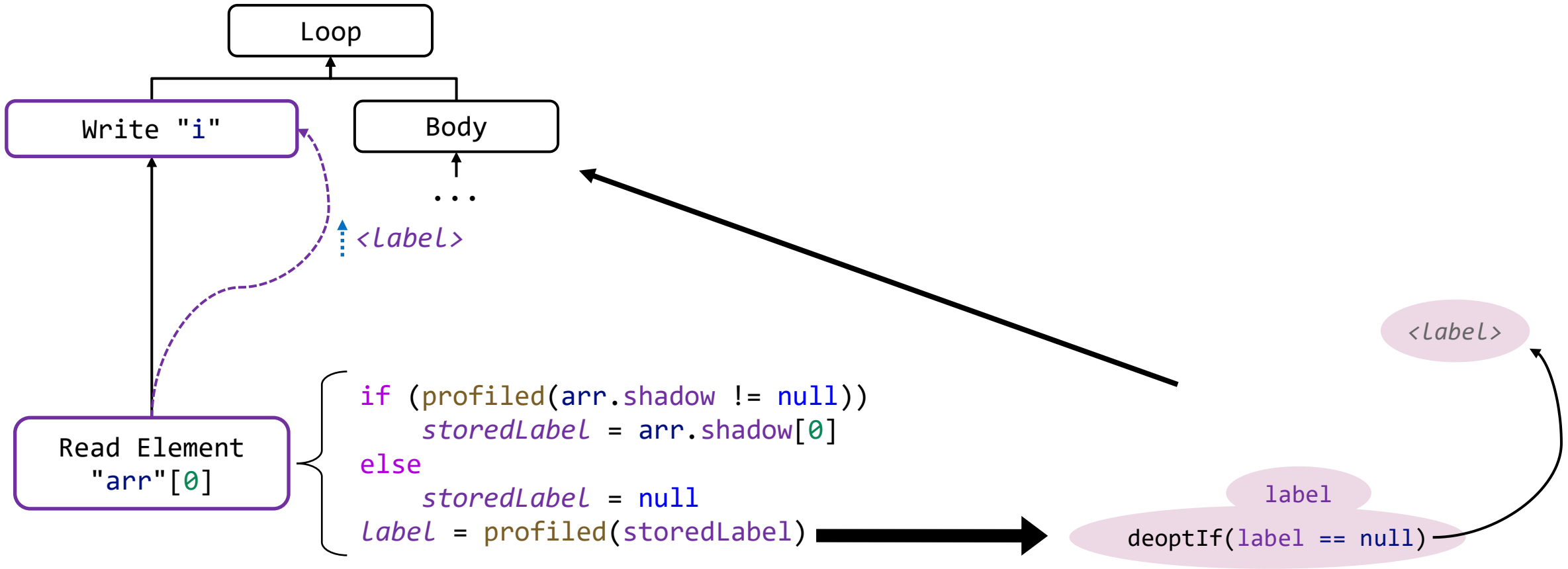
Optimization: Taint Labels of Statement Inputs

```
while (i = arr[0]) { ... }
```



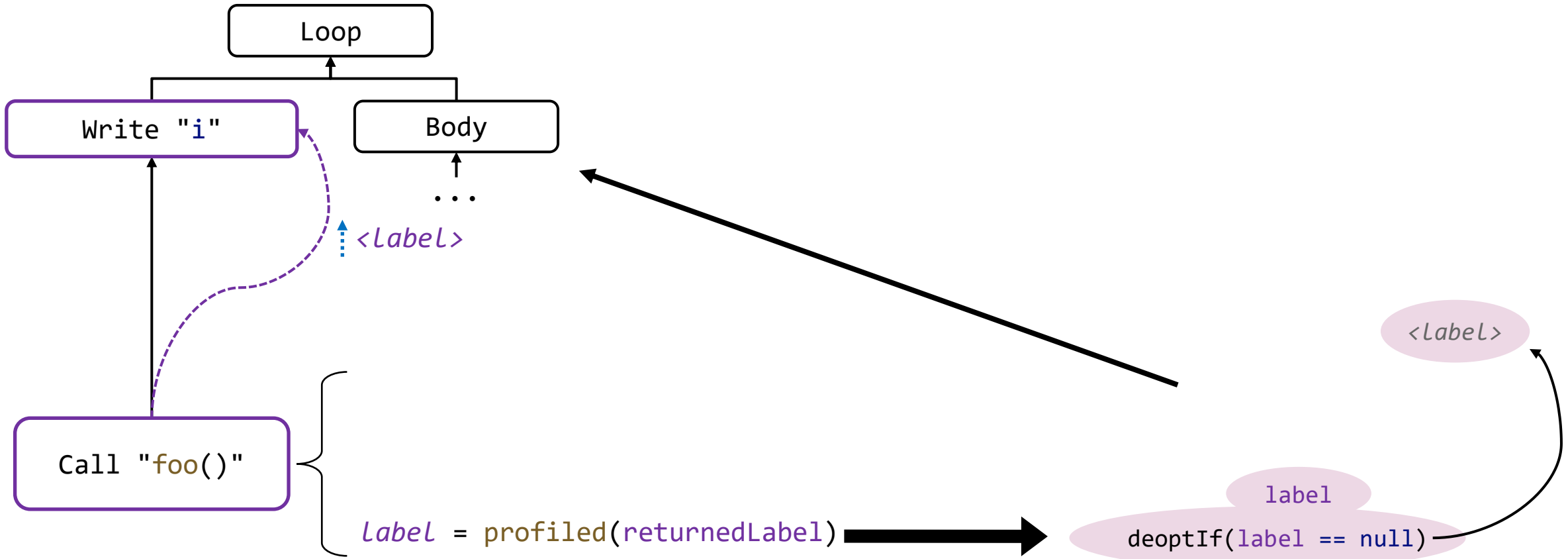
Optimization: Taint Labels of Statement Inputs

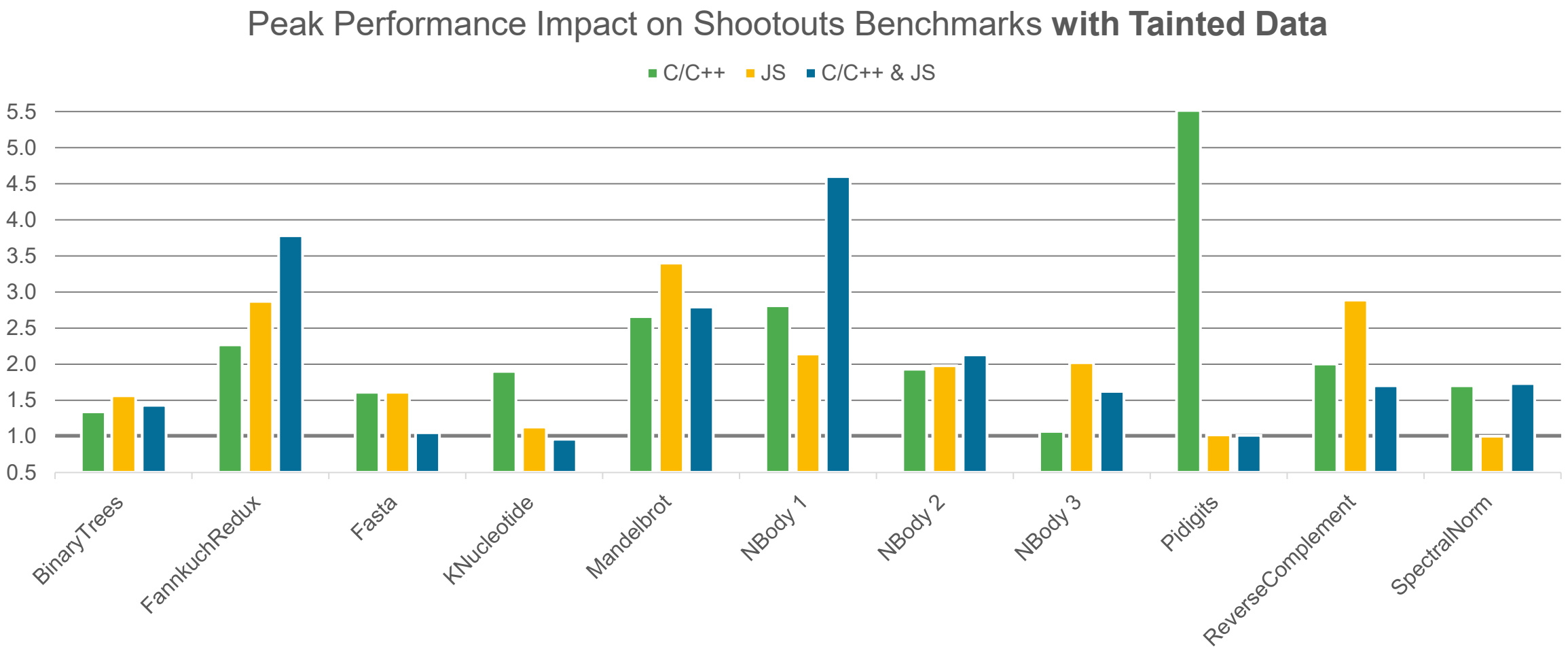
```
while (i = arr[0]) { ... }
```

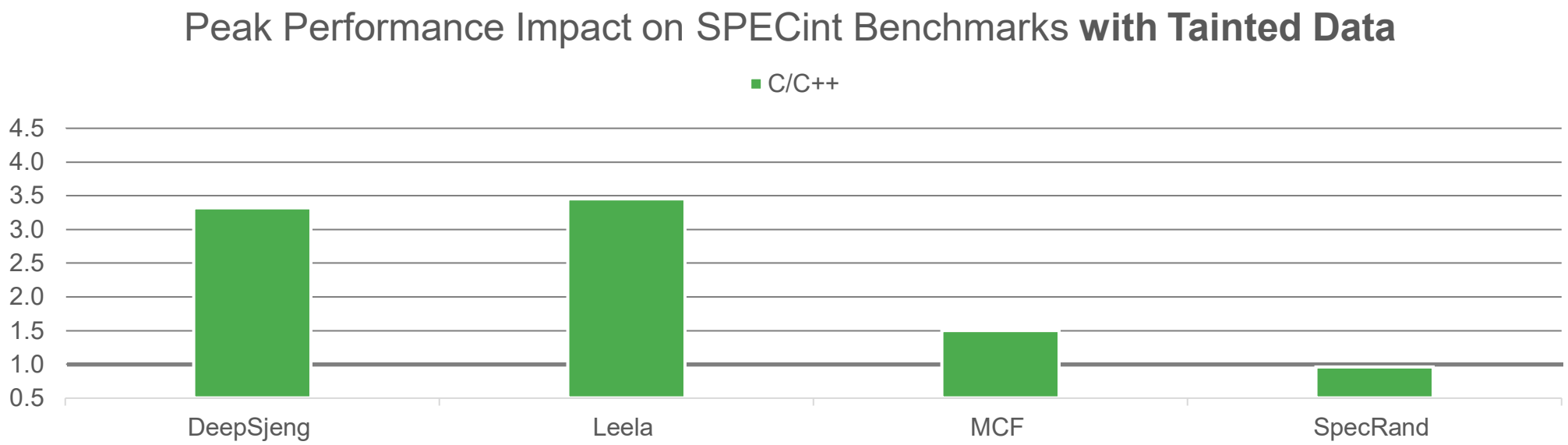


Optimization: Taint Labels of Statement Inputs

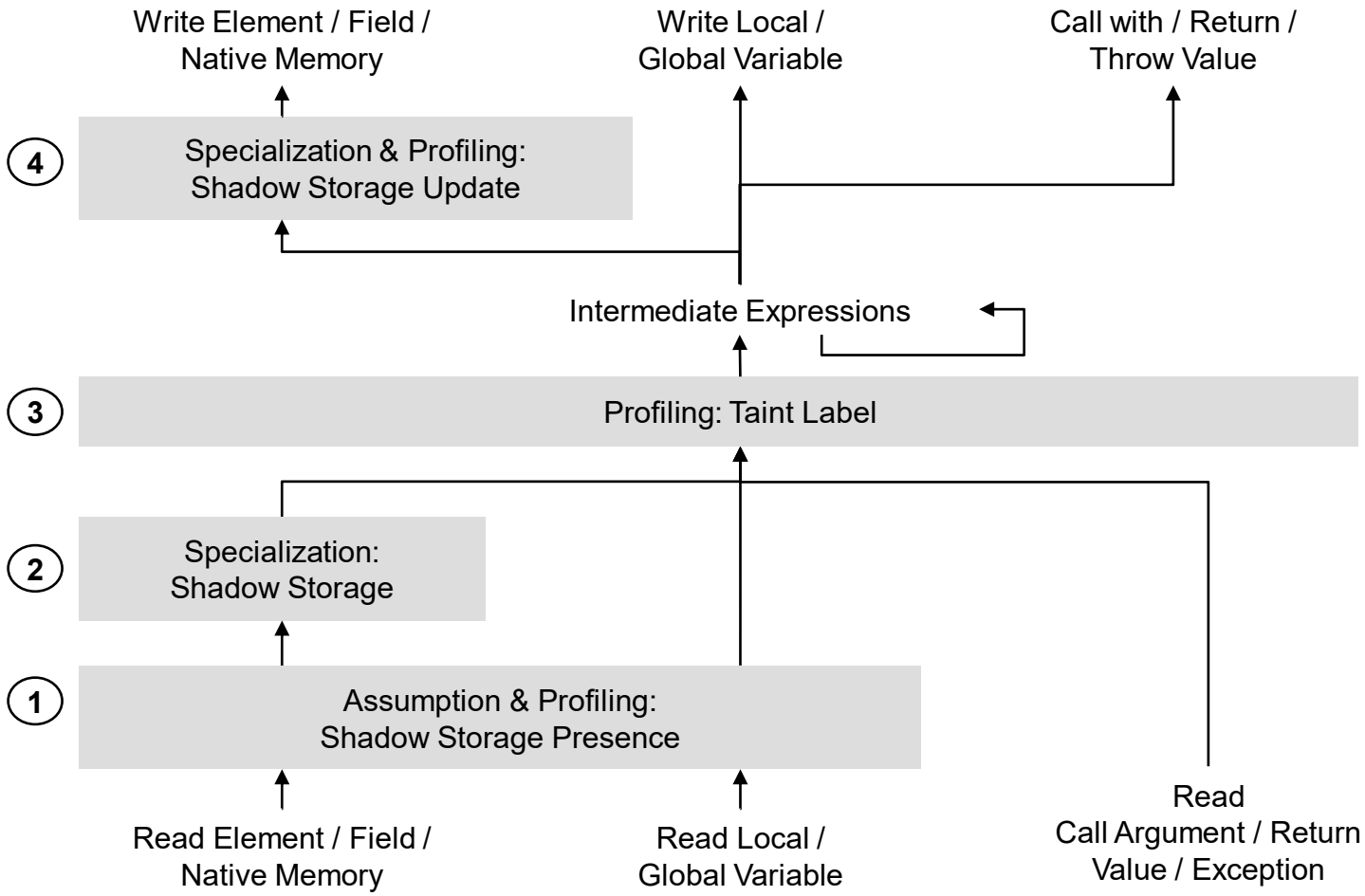
```
while (i = foo()) { ... }
```



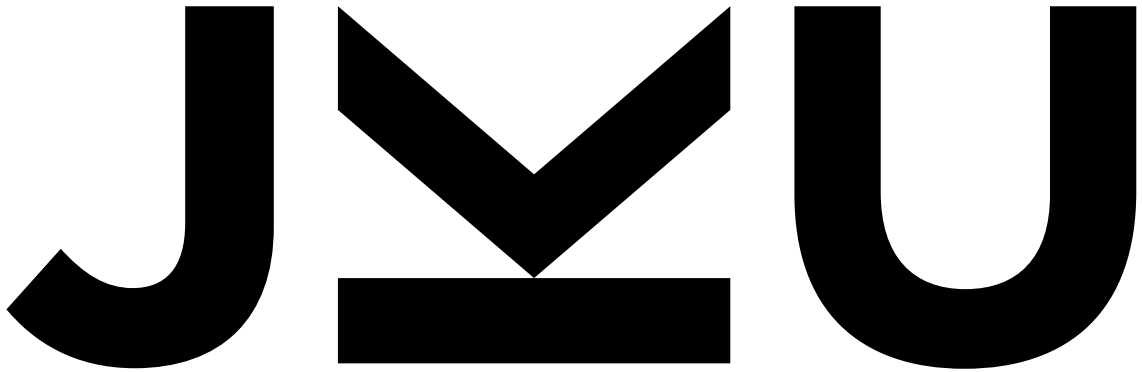

Peak Performance Impact on Shootouts Benchmarks with Tainted Data
C/C++
JS
C/C++ & JS
5.5
5.0
4.5
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
BinaryTrees
FannkuchRedux
Fasta
KNucleotide
Mandelbrot
NBody 1
NBody 2
NBody 3
Pidigits
ReverseComplement
SpectralNorm


Peak Performance Impact on SPECint Benchmarks **with Tainted Data**
C/C++
4.5
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
DeepSjeng
Leela
MCF
SpecRand

Summary



- Language-agnostic speculative assumptions enable optimization opportunities
- Down to 0% slowdown when no taint needs to be propagated
- Up to 5.5x slowdown when taint needs to be propagated

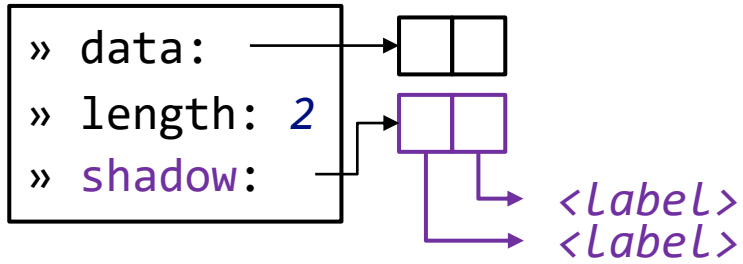


JOHANNES KEPLER
UNIVERSITY LINZ

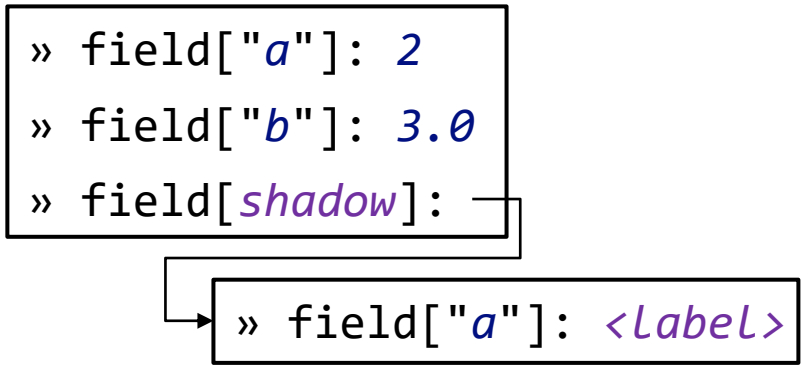
Backup Slides

Shadow Storage Specialization

- Dynamic Array Value



- Dynamic Object Value



- Native Allocation