Sampling Optimized Code for Type Feedback

Olivier Flückiger, Andreas Wälchli,
Sebastián Krynski, Jan Vitek

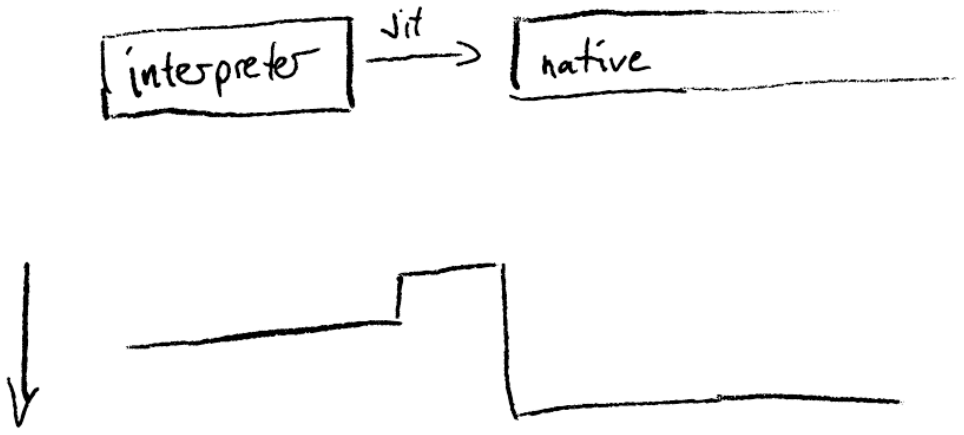
Northeastern University
Czech Technical University

JIT 101

```
add <- function(a, b) {
    a + b
}

add(1, 2)
```

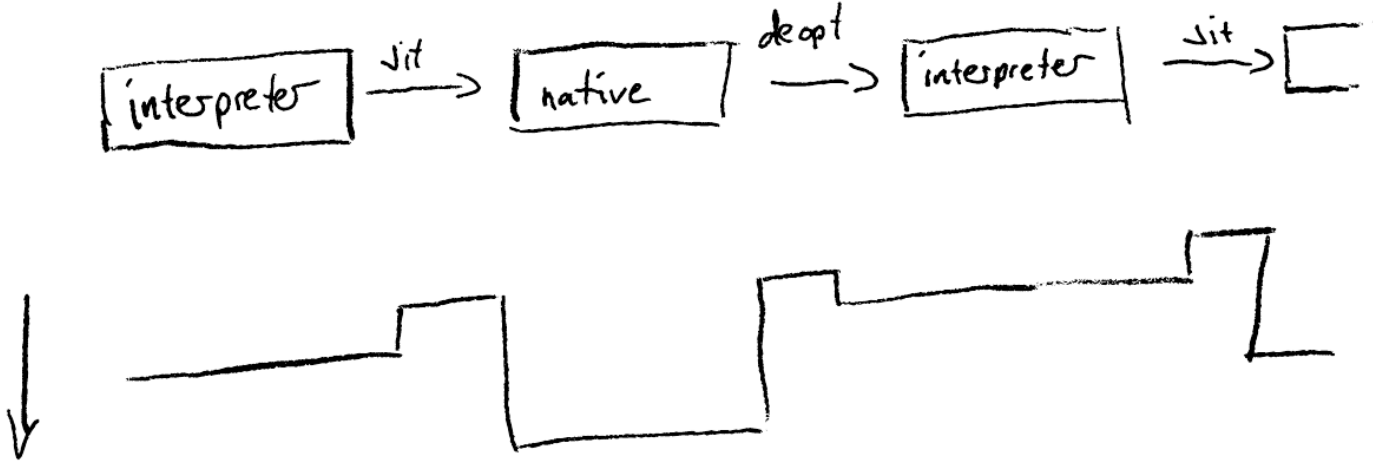
add (1, 2)



add (1, 2)

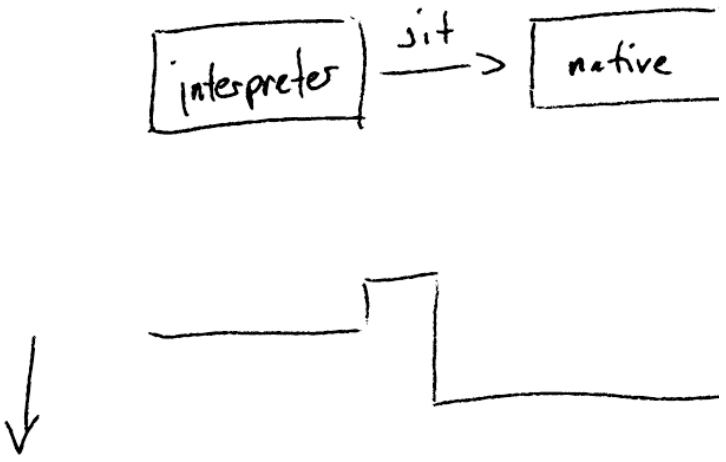
add (1, 2)

add(c(1, 2), c(1, 2))



add (1, 2)

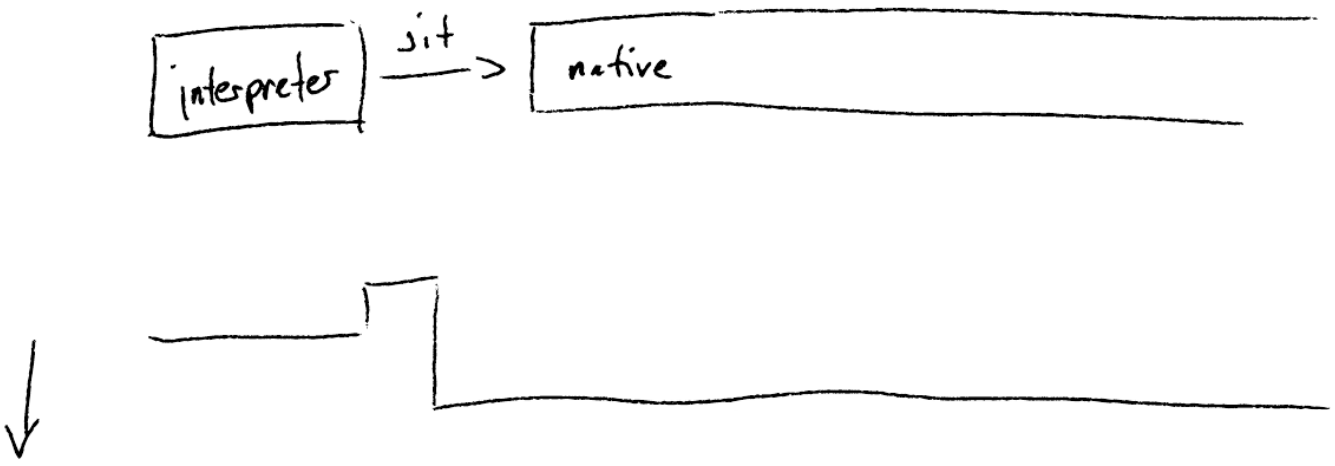
add(c(1, 2), c(1, 2))



add (1, 2)

add (1, 2)

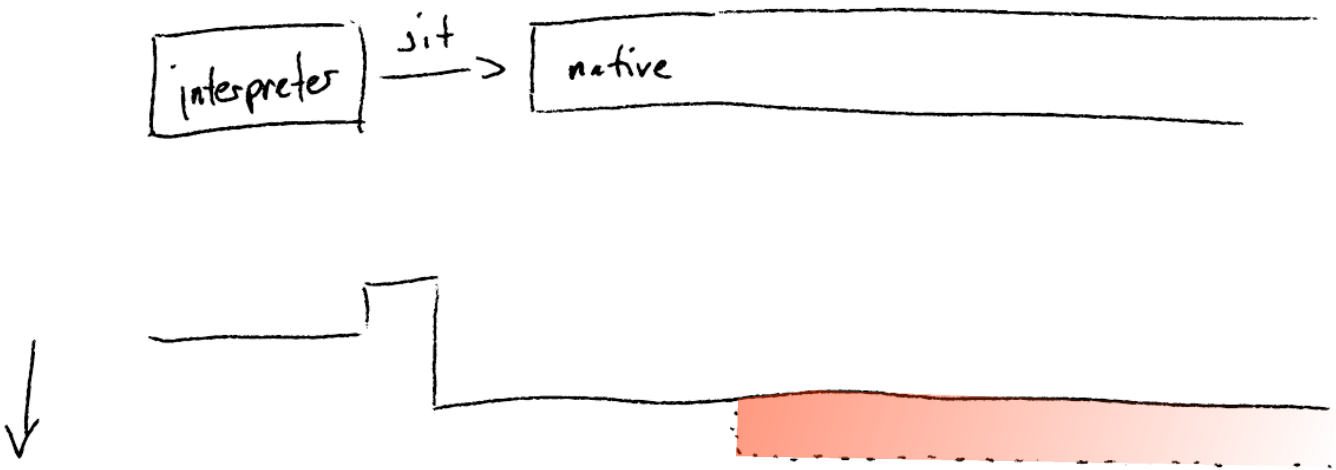
add(c(1, 2), c(1, 2))



add (1, 2)

add (1, 2)

add(c(1, 2), c(1, 2))



Problem

No type feedback from optimized code,

because recording incurs overhead.

Sampling

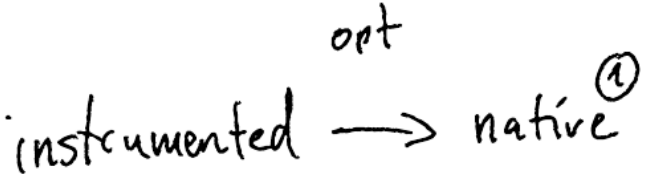
Type feedback needs 100% accuracy,

but sampling can help to detect changes.

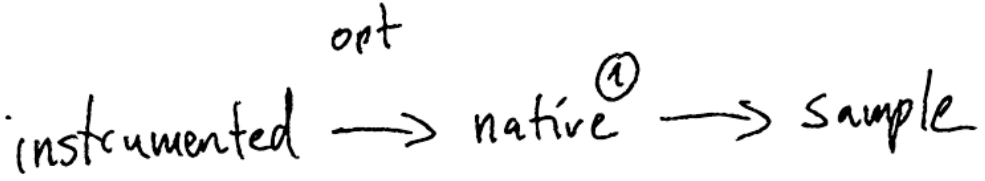
Strategy

instrumented

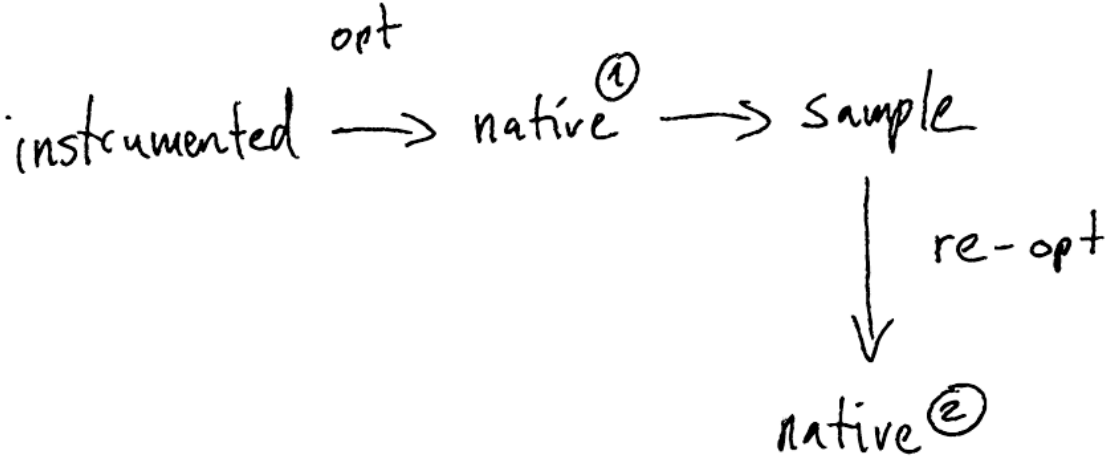
Strategy



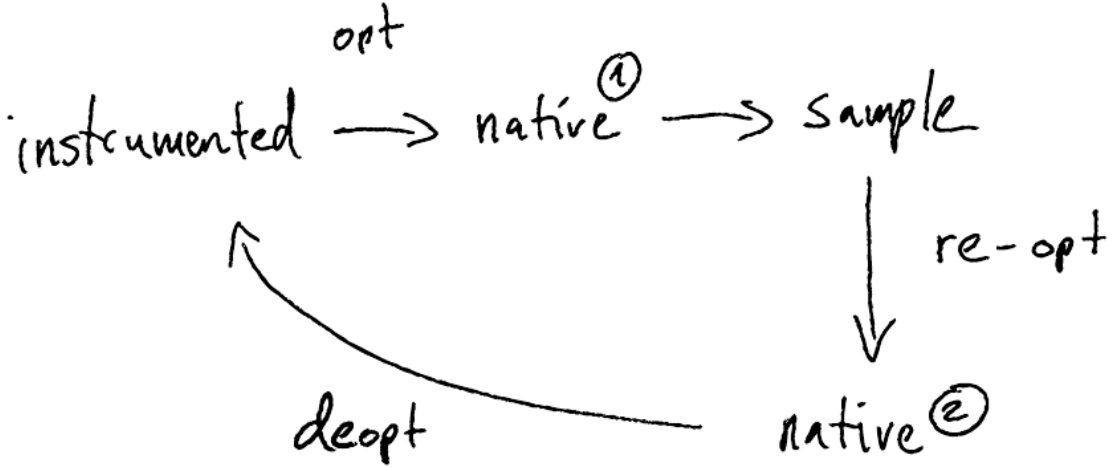
Strategy



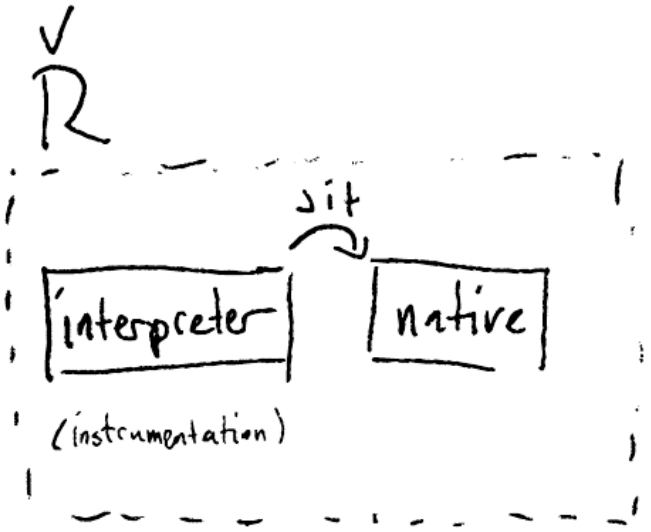
Strategy



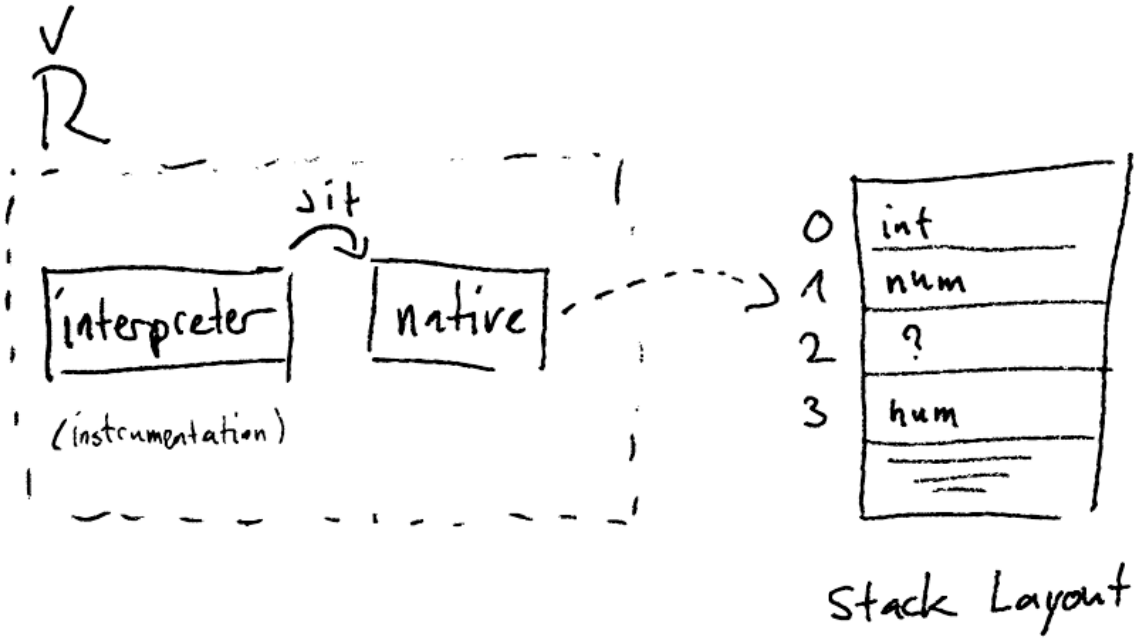
Strategy



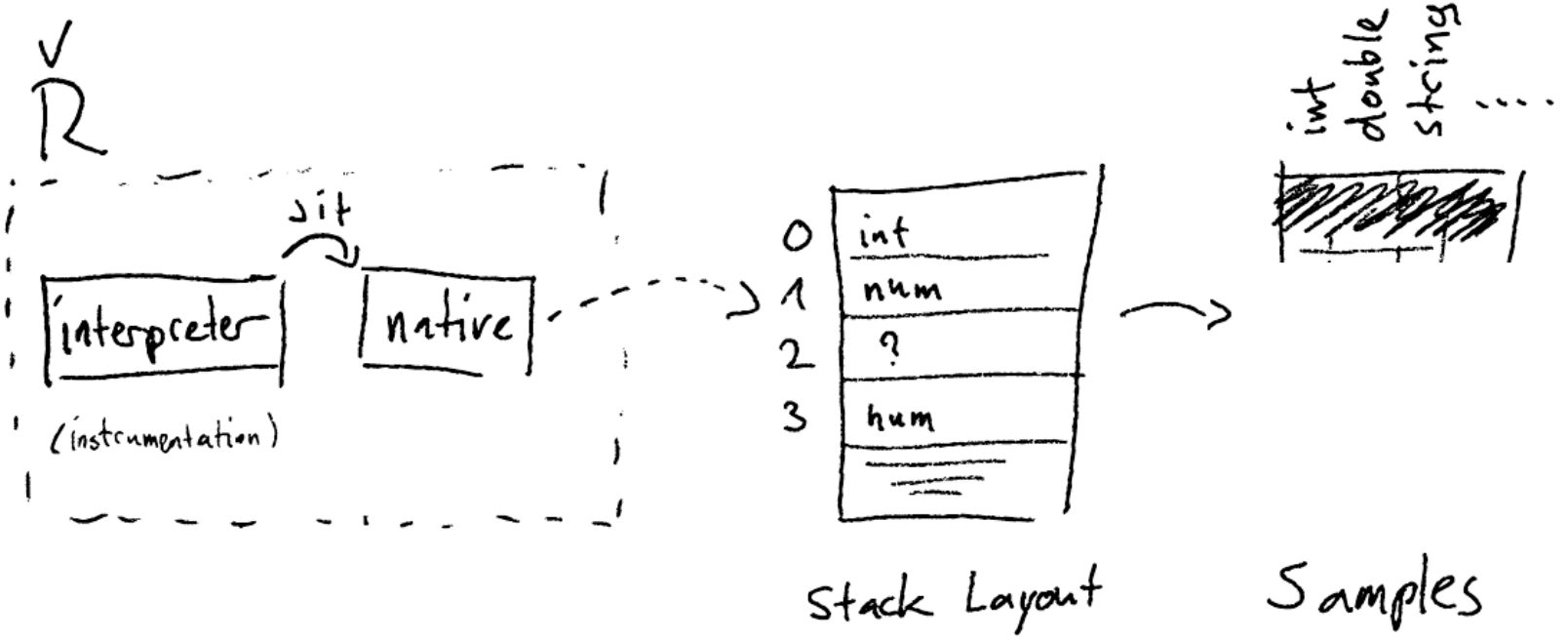
Our Approach



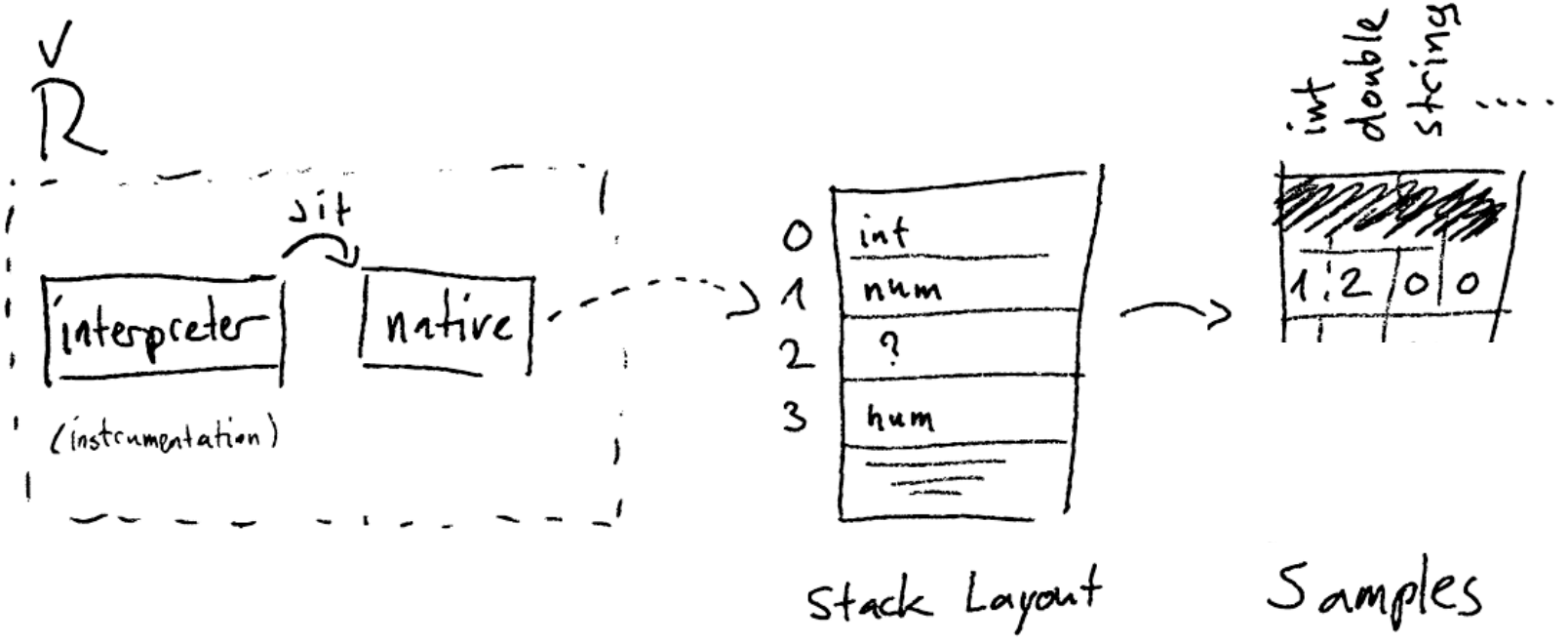
Our Approach



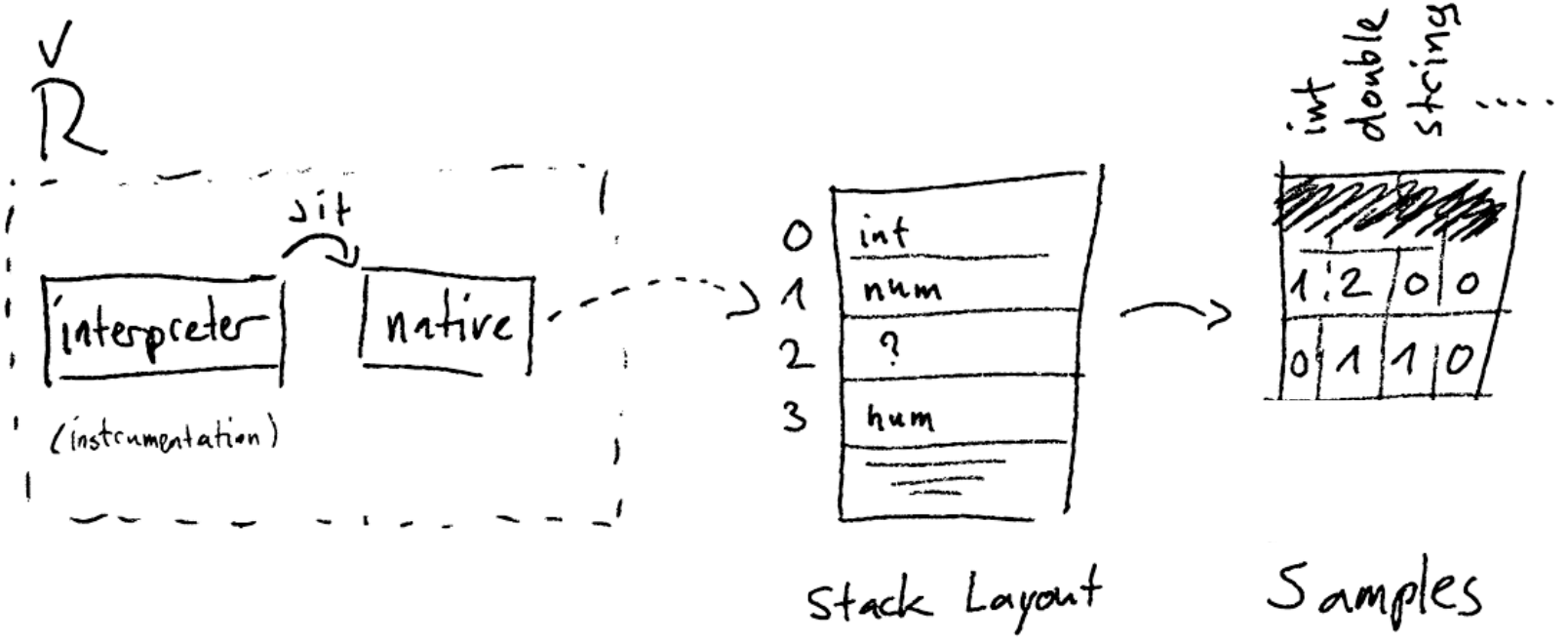
Our Approach



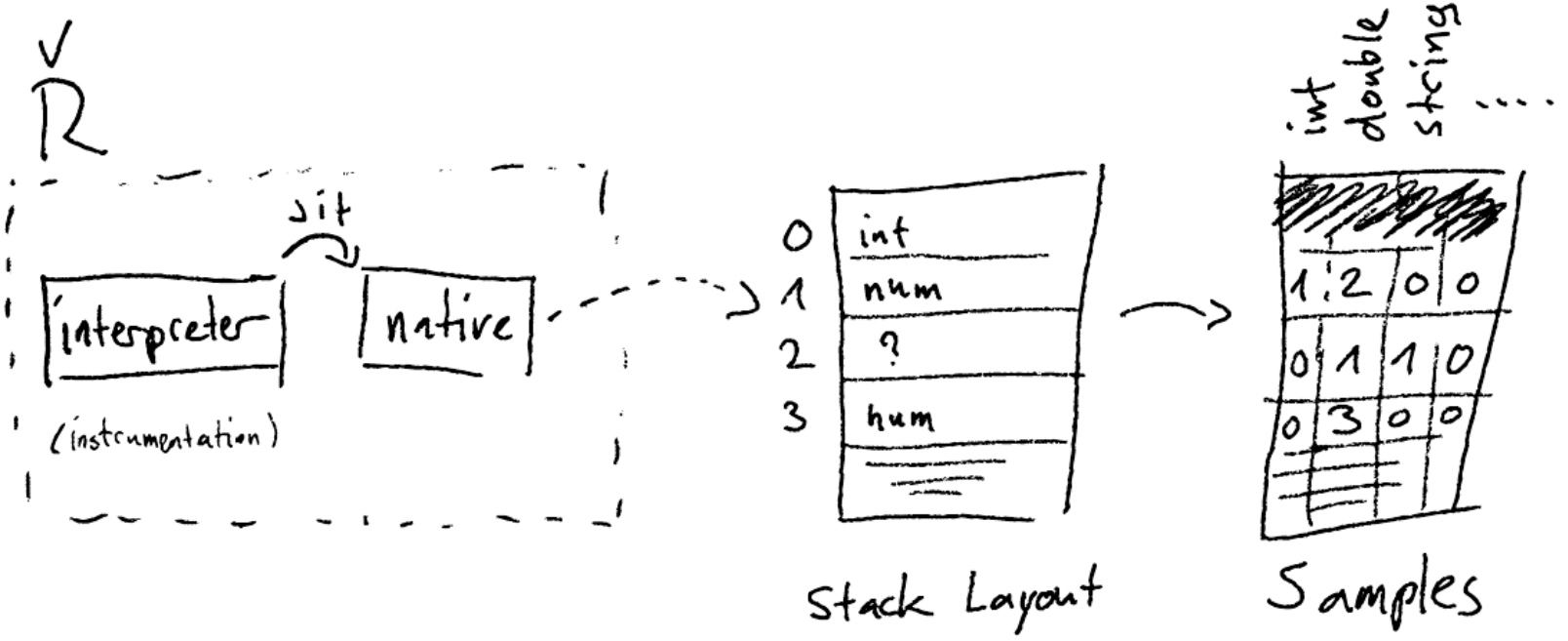
Our Approach



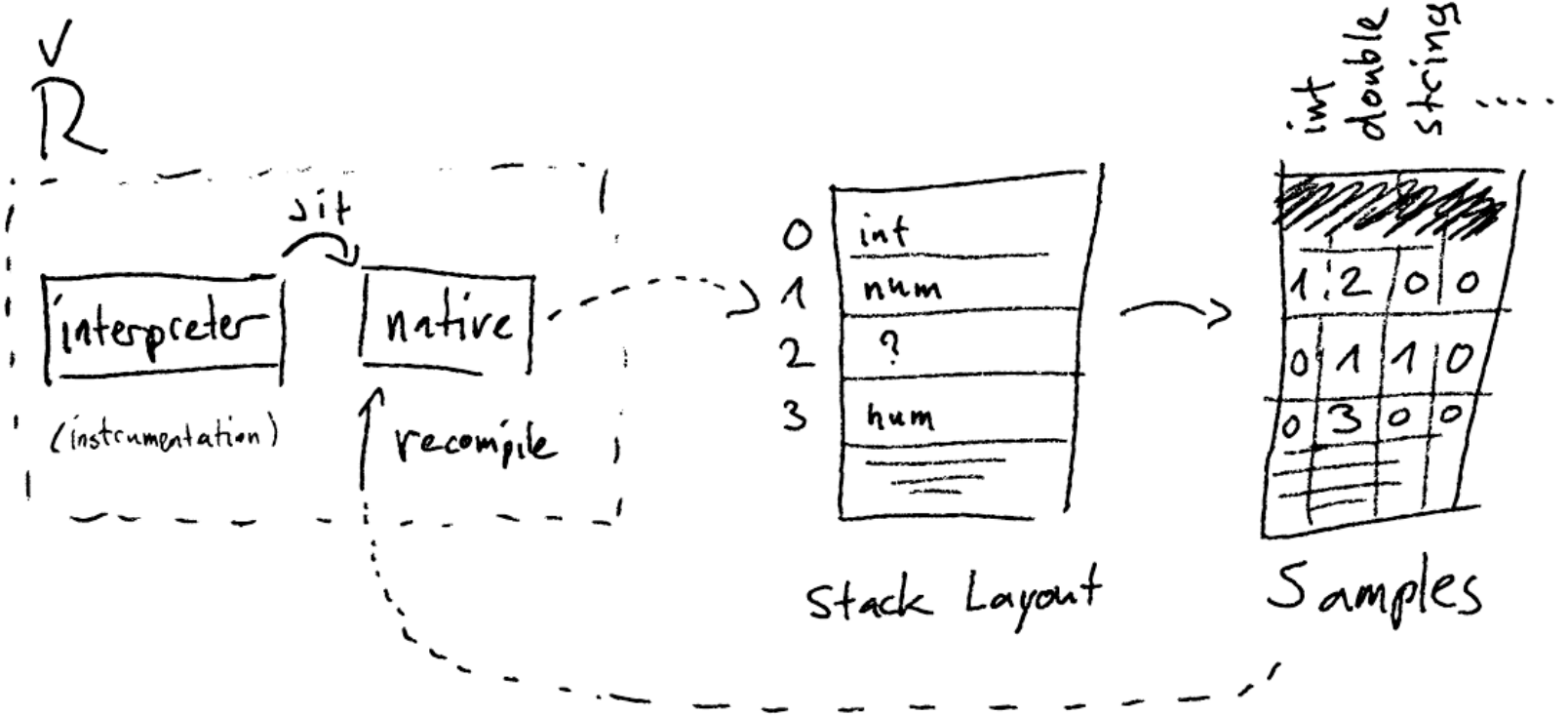
Our Approach



Our Approach

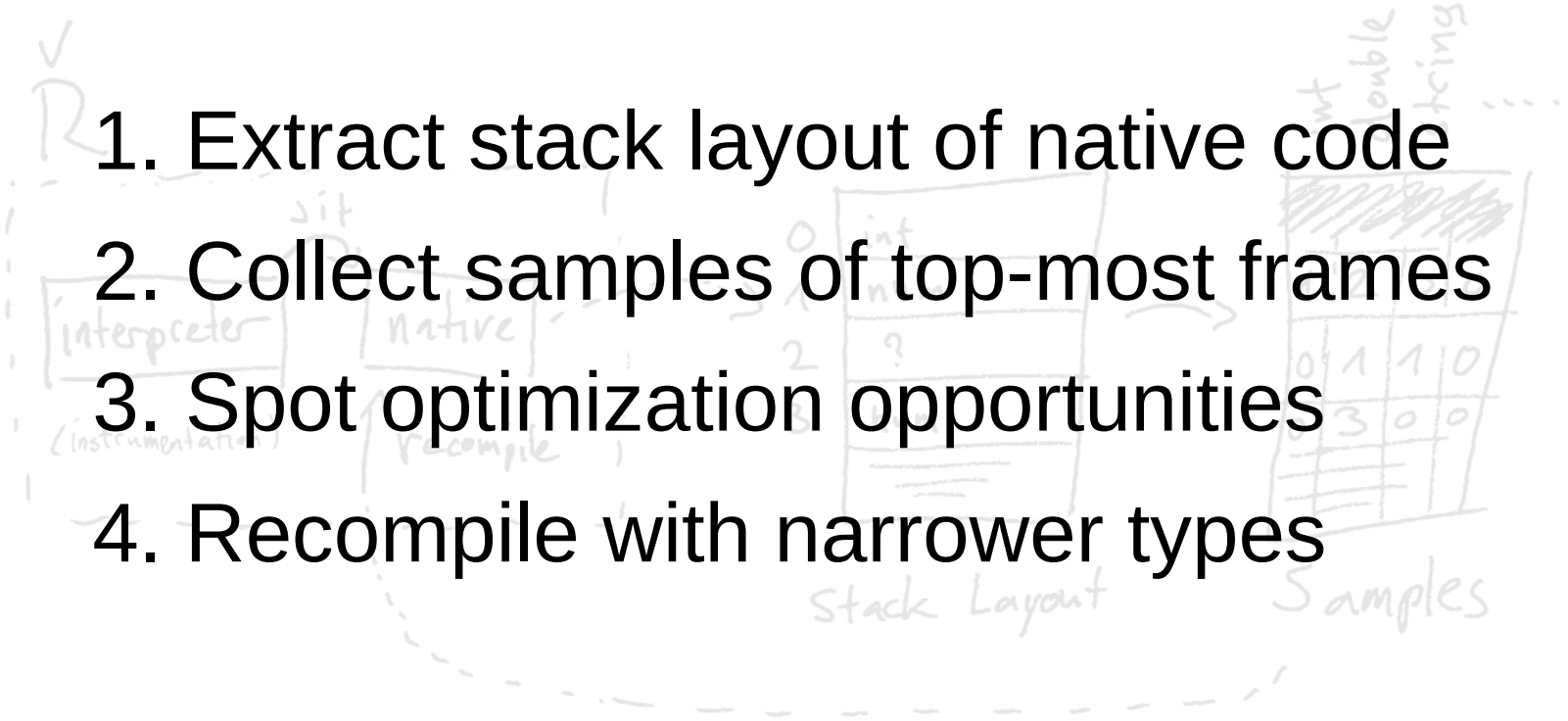


Our Approach



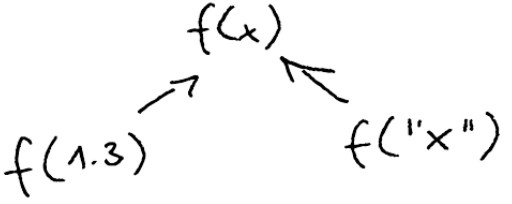
Our Approach

1. Extract stack layout of native code
2. Collect samples of top-most frames
3. Spot optimization opportunities
4. Recompile with narrower types



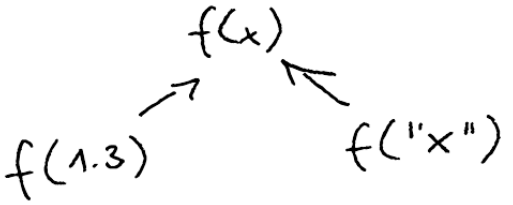
Success

— type pollution



Success

— type pollution

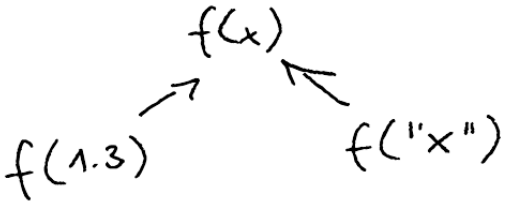


— int to real autocow.

```
x ← 1L
x ← x + 1
```

Success

- type pollution



- int to real autocow.

$x \leftarrow 1L$

$x \leftarrow x + 1$

- interactive session

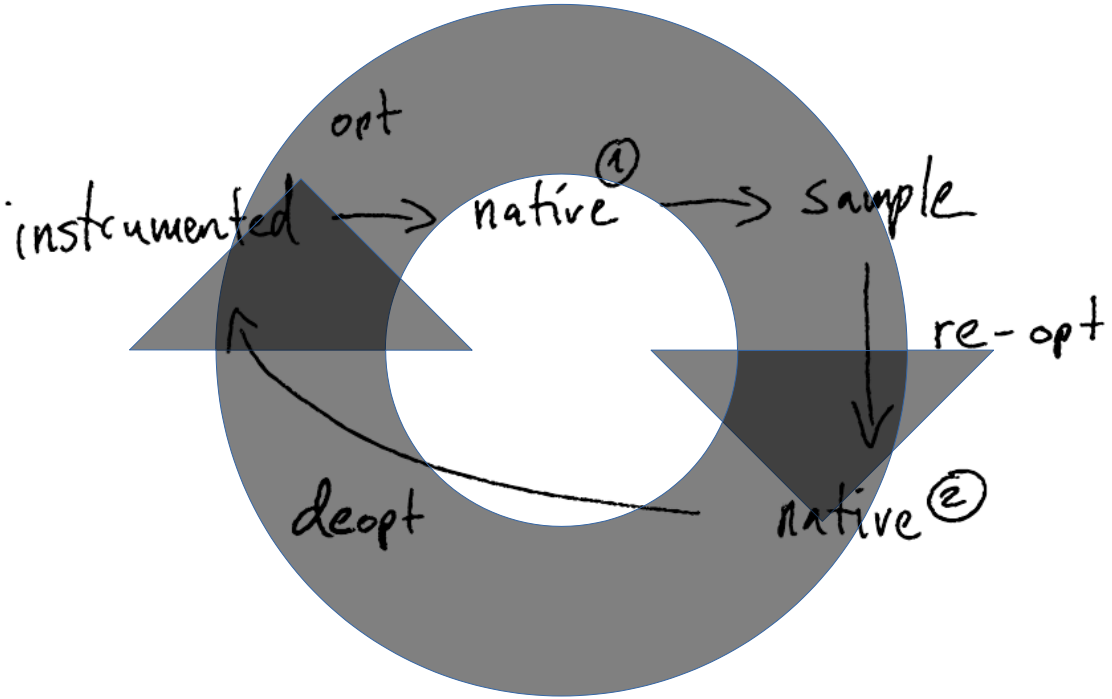
Success

End-to-end prototype, which detects these situations in ms, has a low overhead (<10%).

Success

End-to-end prototype, which detects these situations in ms, has a low overhead (<10%).

But, no robust end-to-end heuristics yet!


opt
instrumented
native ①
sample
re-opt
deopt
native ②

Uncooperative Environment

To prevent dead-locks due to restarted syscalls, we use the PMU to only interrupt the VM when executing in userspace.

Measurement Issues

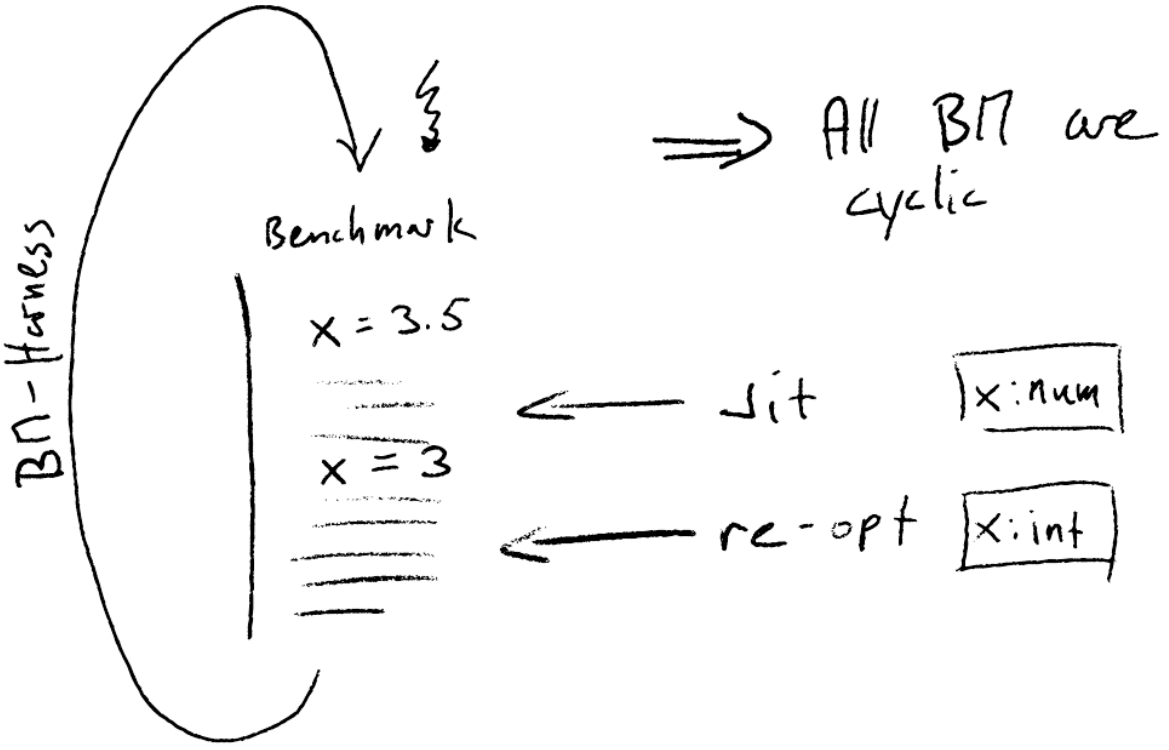
Typical BM:

```
for (i in 1:warmup)
  benchmark()

for (i in 1:measurements)
  t = t + system.time(benchmark())

t / measurements
```

Measurement Issues



Profiling Optimized Code?

Normally the VM flies blind once the last tier is reached, there are vast possibilities to monitor the optimized code quality.

We need diversification of BM methodology!

The PMU is underused!!