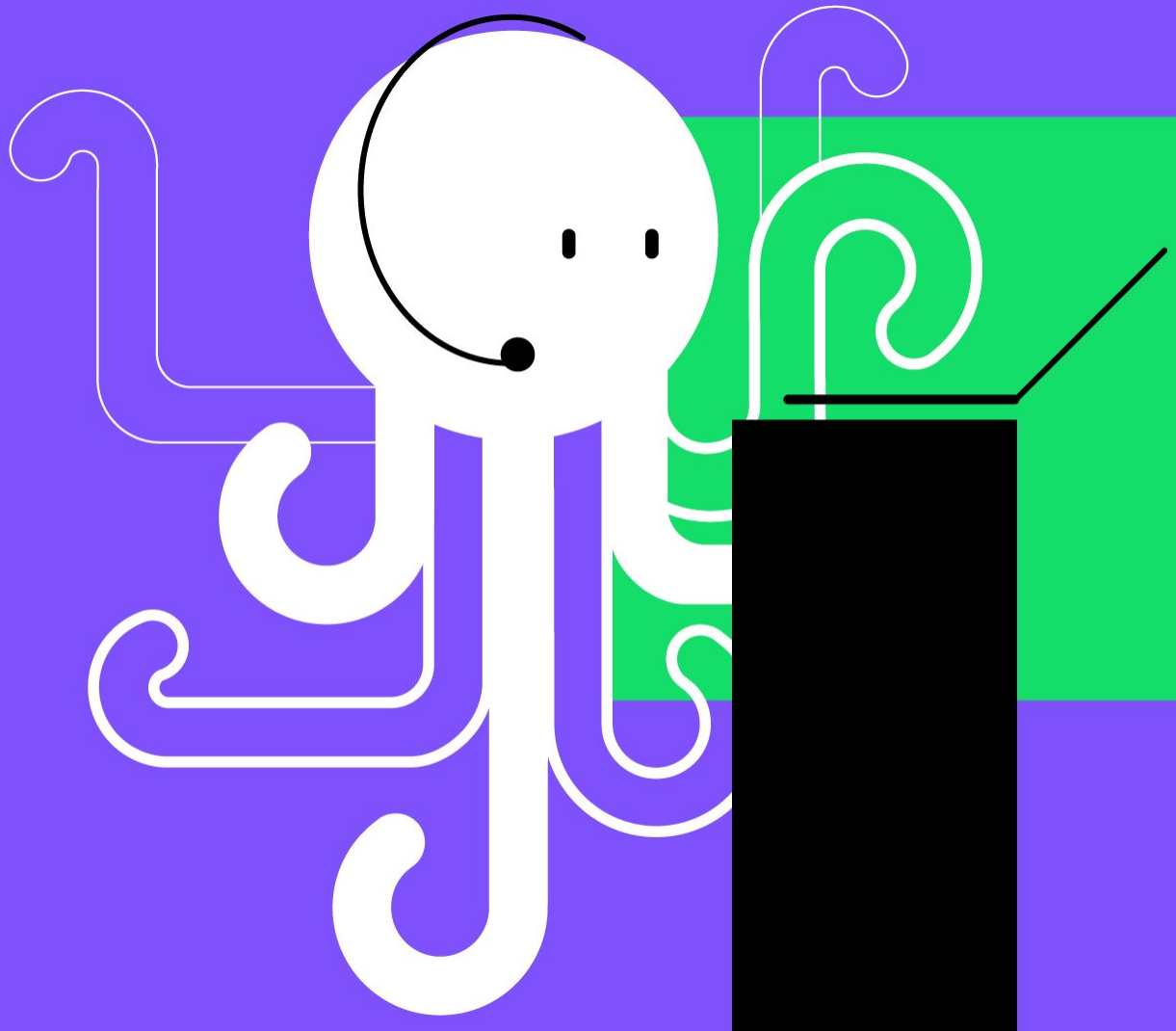


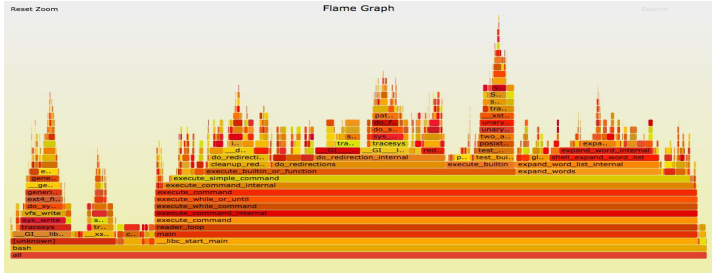
Finding performance bugs in Kotlin compiler via fuzzing

WIP report

Стажер: Дмитрий Волков
Ментор: Виктор Петухов
Тимофей Брыксин
Ярослав Соколов

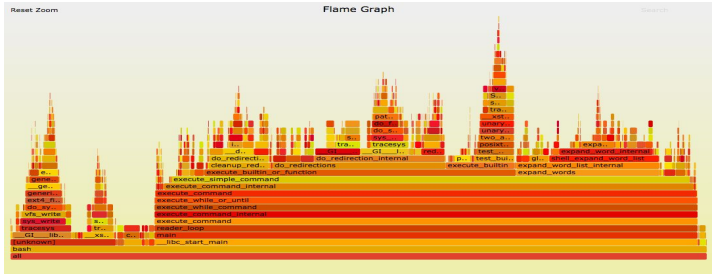
```
$ time kotlinc ./my.kt
Slow!
```

```
Premature optimisation is
the root
```



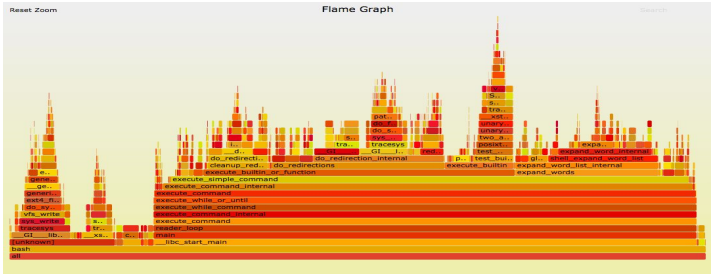
```
$ time kotlinc ./my.kt
Slow!
```

```
Premature optimisation is
the root
```




```
kotlin.git/benchmarks
```

```
profiler
```

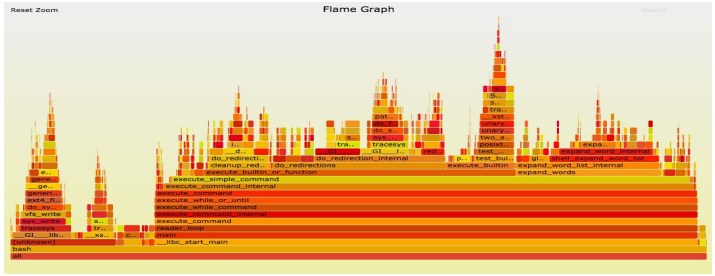


```
$ time kotlinc ./my.kt
Slow!
```

+

Premature optimisation is the root

=



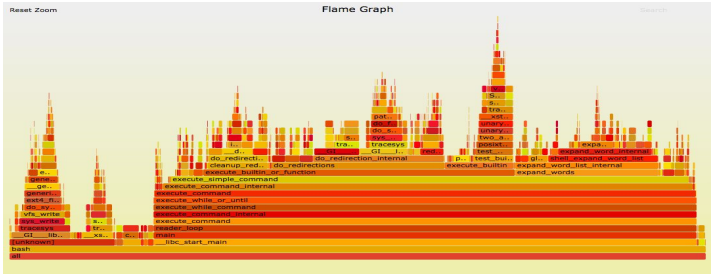

```
kotlin.git/benchmarks
```

Only cover what we know

+

profiler

=

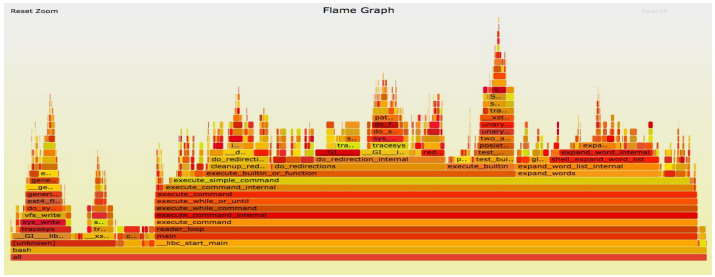


```
$ time kotlinc ./my.kt
Slow!
```

+

```
Premature optimisation is
the root
```

=




```
generate benchmarks
on-the-fly
```

Only cover what we know

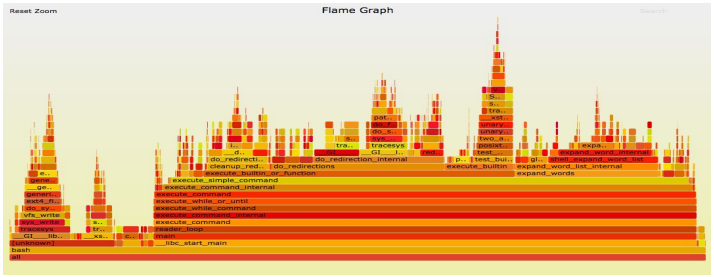
+

```
profiler
```

feedback

=

fuzzing



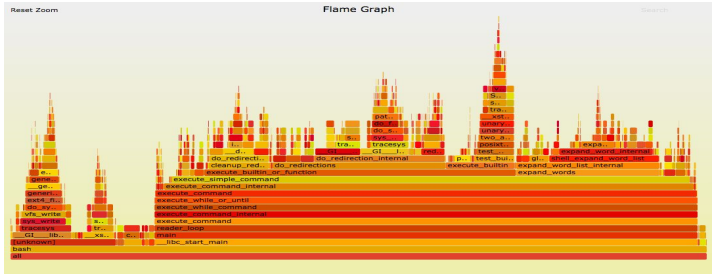
Prior work: American Fuzzy Lop

```
$ time kotlinc ./my.kt
Slow!
```

+

```
Premature optimisation is
the root
```

=



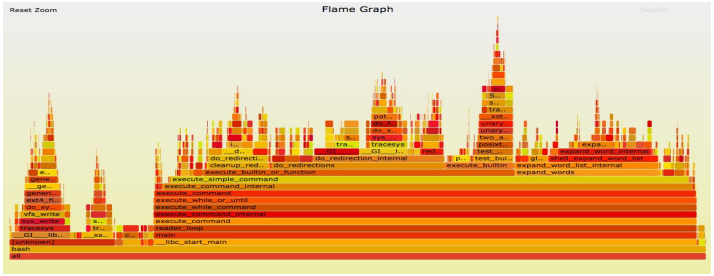

```
generate benchmarks
on-the-fly
```

+

```
profiler
```

=

fuzzing

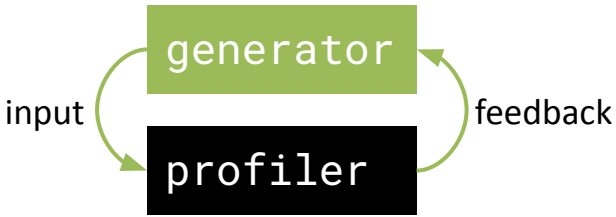


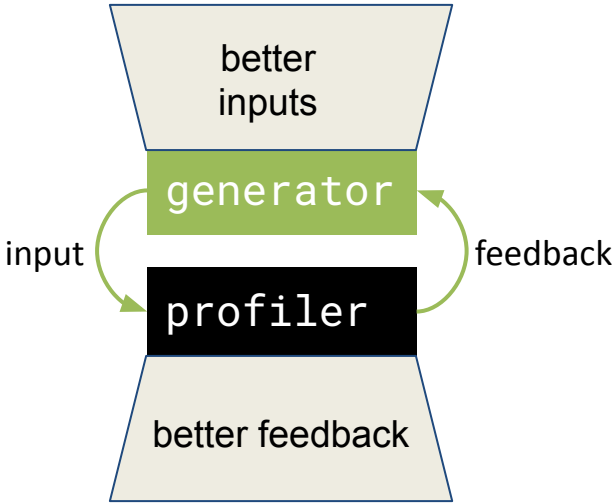
C
w
k

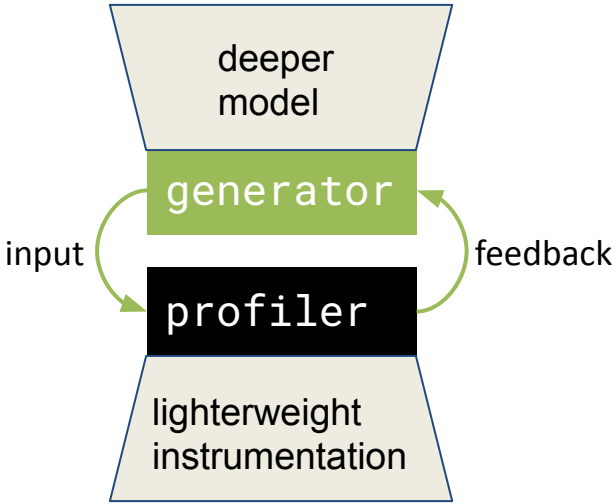
fe

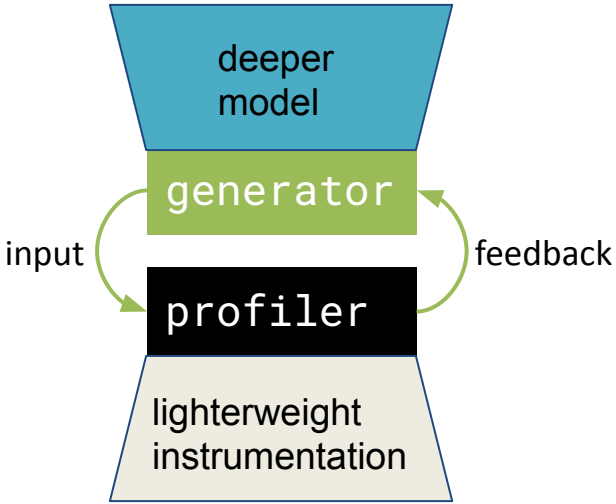
Prior work: American Fuzzy Lop

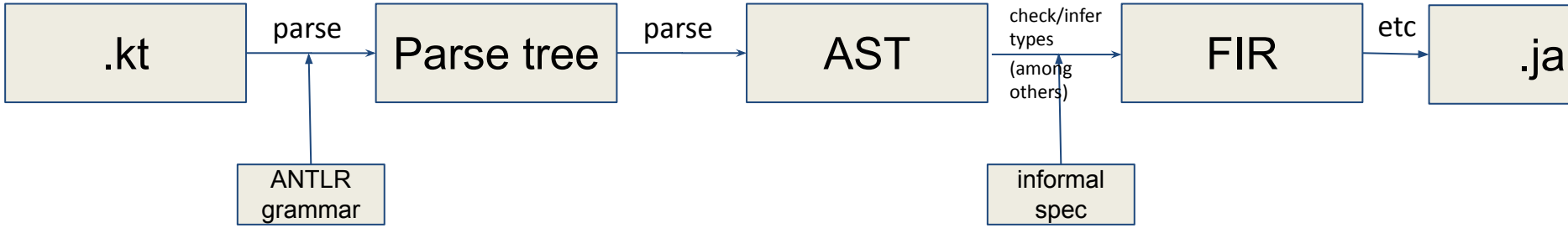
DOES NOT WORK

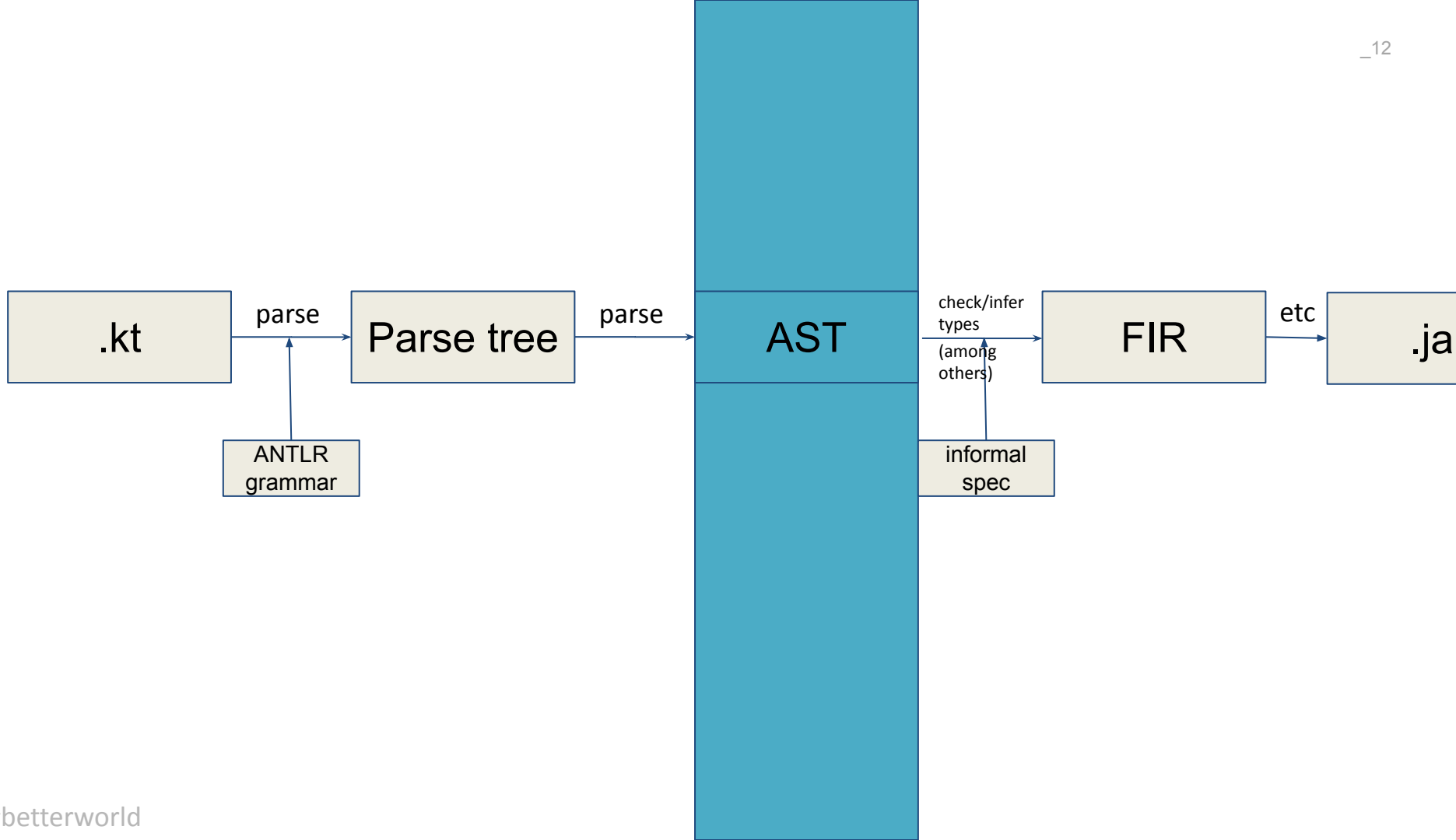

generator
input
feedback
profiler

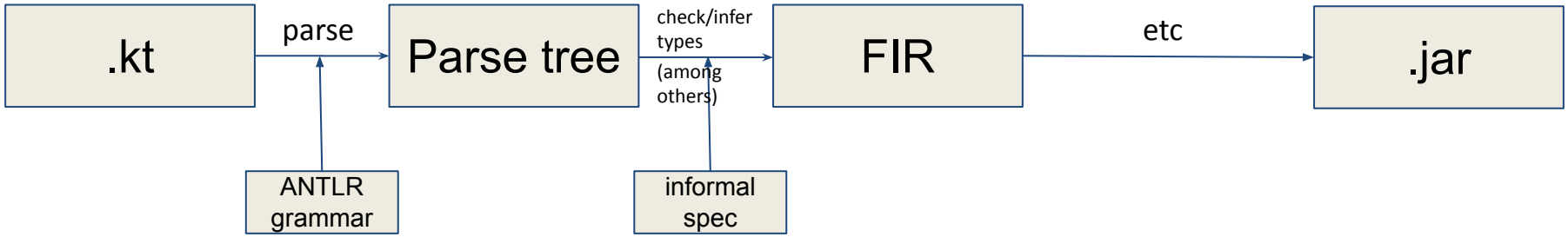

better
inputs
generator
input
feedback
profiler
better feedback

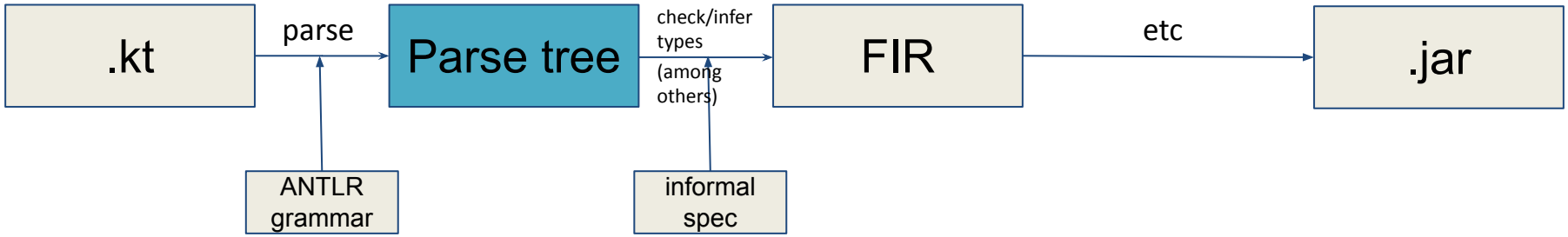

deeper
model
generator
input
feedback
profiler
lighterweight
instrumentation

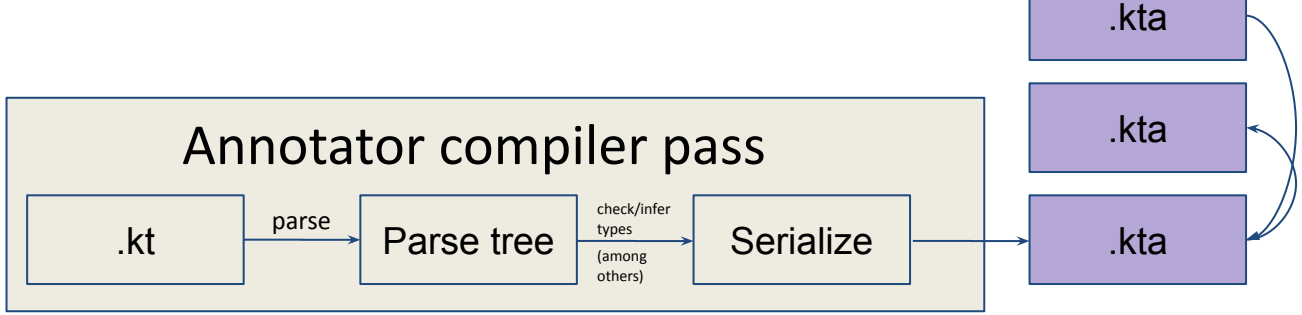

deeper
model
generator
input
feedback
profiler
lighterweight
instrumentation


.kt
parse
Parse tree
parse
AST
check/infer types
(among others)
FIR
etc
.ja
ANTLR grammar
informal spec

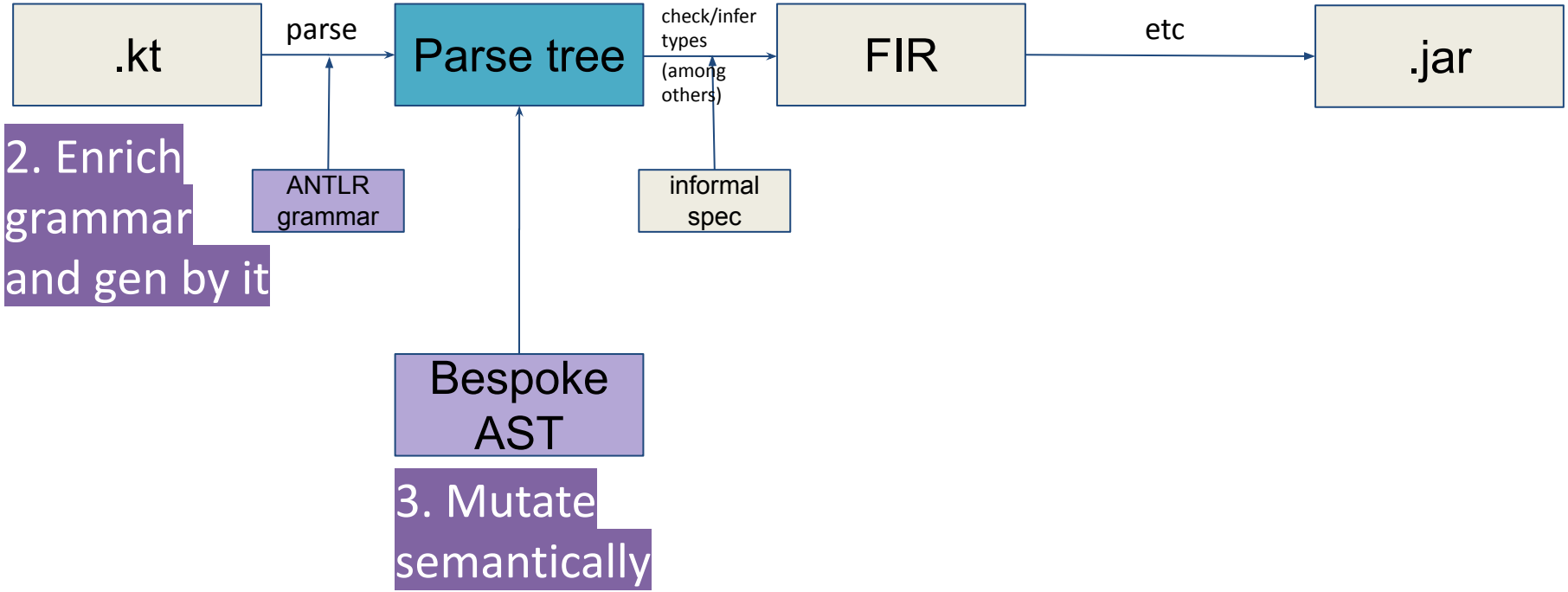

.kt
parse
Parse tree
parse
AST
check/infer types
(among others)
FIR
etc
.ja
ANTLR grammar
informal spec


.kt
parse
Parse tree
check/infer types
(among others)
FIR
etc
.jar
ANTLR grammar
informal spec


.kt
parse
Parse tree
check/infer types
(among others)
FIR
etc
.jar
ANTLR grammar
informal spec

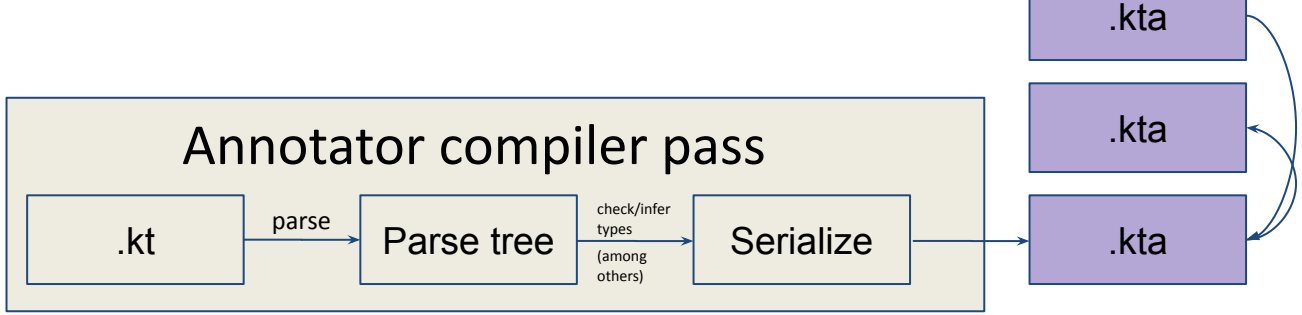


1. “AST” mix & match



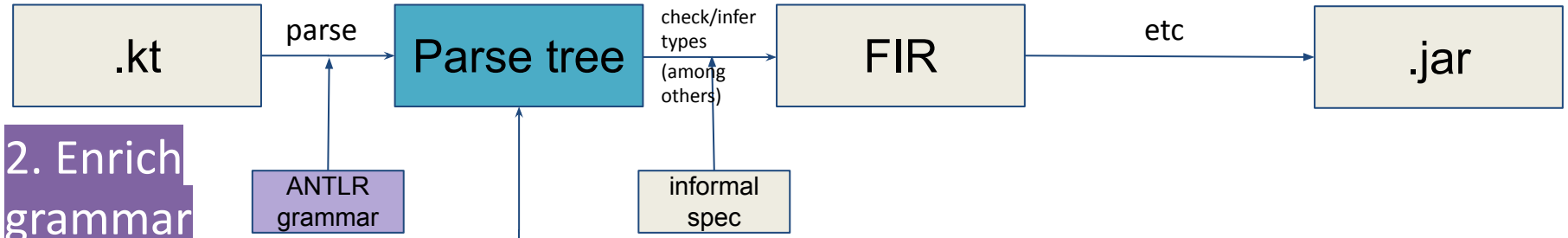
2. Enrich grammar and gen by it

3. Mutate semantically



1. “AST” mix & match

v1 almost there...



2. Enrich grammar and gen by it

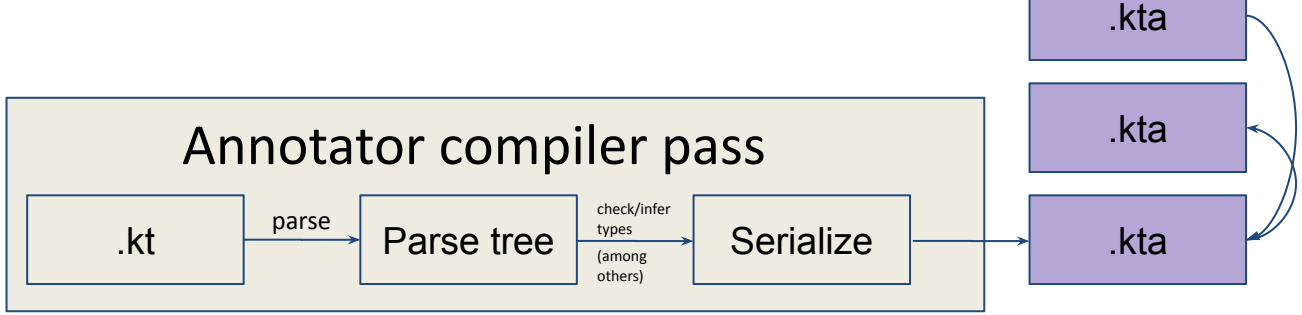
Heuristic+RL approach seems viable (RLcheck)

+DeepSmith experiment

Bespoke AST

3. Mutate semantically

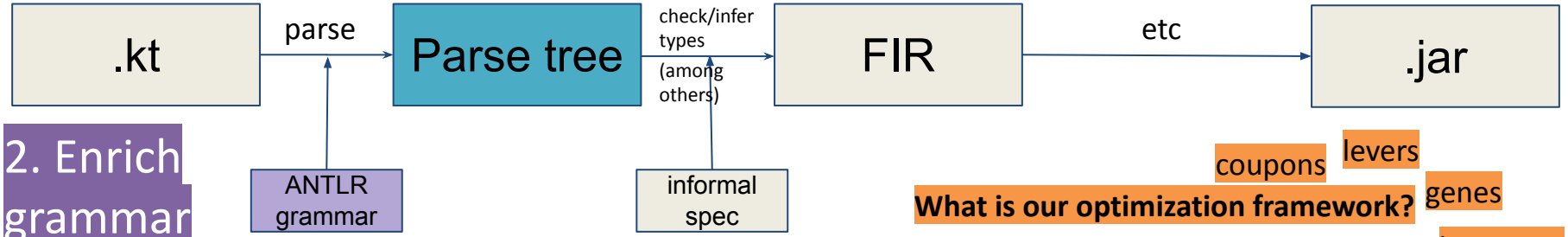
Out of scope this time



Is mixer complete & sound?

1. "AST" mix & match

v1 almost there...



2. Enrich grammar and gen by it

"gen"

Heuristic+RL approach seems viable (RLcheck)

+DeepSmith experiment

How much expert effort to enrich?

How do we schedule seeds?

How cheap is useful instrumentation?

How do we measure perf?

coupons

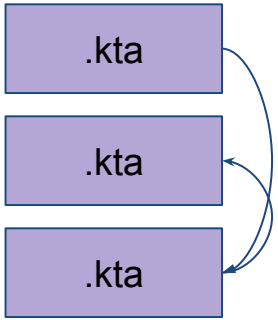
levers

What is our optimization framework?

genes

heuristics

bayesian updates



Novelty: semantic performance fuzzer +RL

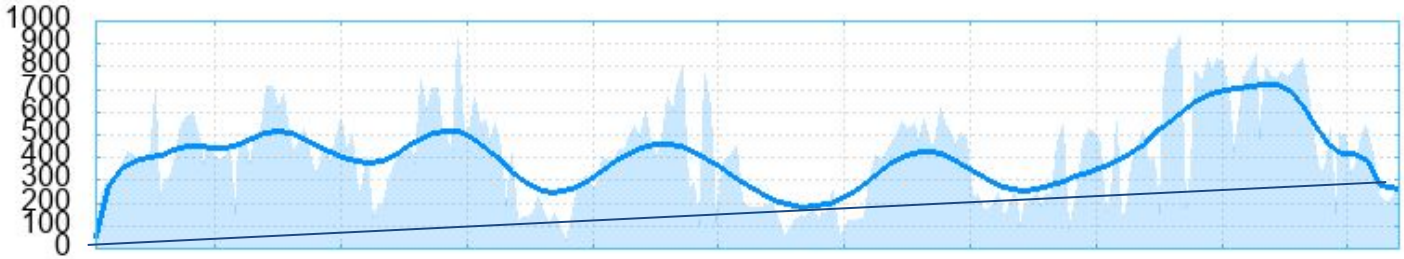
1. “AST” mix & match

v1 almost there...

TBD: name mangling, integrate upstream patch

8h

WIP report



Threats

- Kotlinc is *slow* (100 op/sec clear, 5 op/sec instrumented)
- Dataset (= compiler tests) incomplete
- Semi-blackbox model => useless rewrites
- Naive perf convergence assumptions
- No shrinking
- Unclear reusability

2. Enrich grammar and gen by it

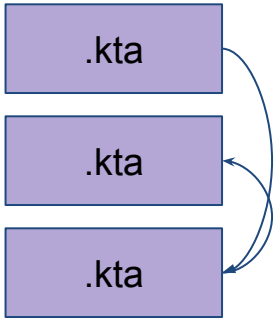
scale

parse github

heuristics

generalize

reuse prior JBR work



Novelty: semantic performance fuzzer +RL

1. “AST” mix & match

v1 almost there...

TBD: name mangling, integrate upstream patch

8h

WIP report

wldhx@wldhx.me, victor.petukhov@jetbrains.com

Threats

- Kotlinc is *slow* (100 op/sec clear, 5 op/sec instrumented)
- Dataset (= compiler tests) incomplete
- Semi-blackbox model => useless rewrites
- Naive perf convergence assumptions
- No shrinking
- Unclear reusability

2. Enrich grammar and gen by it

scale

parse github

heuristics

generalize

reuse prior JBR work