Kex in 2021: Ups and Downs

Azat Abdullin

December 17, 2021

Kex

- white box fuzzer for JVM bytecode
- based on symbolic execution
- test generation for Java and Kotlin

What happened in 2021

- participation in SBST Java tool competition 2021[1]
- improving Java standard library support
- evaluation of Reanimator

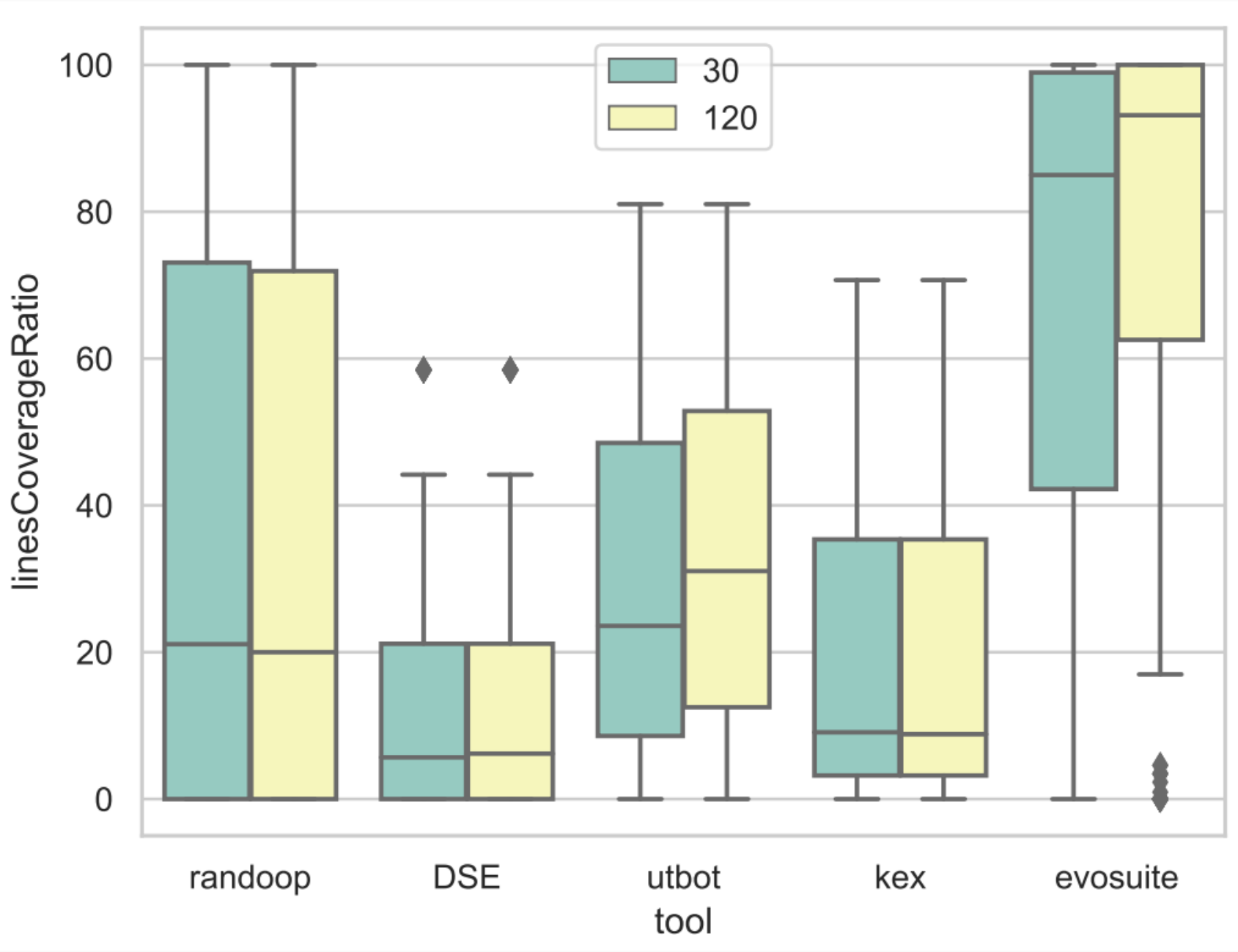
[1] Panichella S. et al. Sbst tool competition 2021 //2021 IEEE/ACM 14th International Workshop on Search-Based Software Testing (SBST). – IEEE, 2021. – C. 20-27.

SBST 2021

Automatic test case generation competition

- 6 test projects with 98 benchmarks
- 30 and 120 second time budgets
- 4 participaiting tools and 1 baseline

SBST 2021 results



SBST 2021: Kex results[2]

- Kex was ranked fifth with score of 44.21
- Kex achieved any coverage only on one project
    - average coverage of ~20%
- Kex failed on 5 out of 6 projects
    - 1 project failed because of unhandled ASM error
    - 2 projects failed because Kfg encoutered some unexpected bytecode
    - 2 projects failed because Kex required too much RAM
- Kex (and Reanimator) failed on some of the more complex language features (abstract classes, inner classes, etc.)
- Kex required too much of disk space

Main teakeaway: Kex had a low level of maturity

[2] Abdullin A., Akhin M., Belyaev M. Kex at the 2021 SBST Tool Competition //2021 IEEE/ACM 14th International Workshop on Search-Based Software Testing (SBST). – IEEE, 2021. – C. 32-33.

SBST 2021: implications

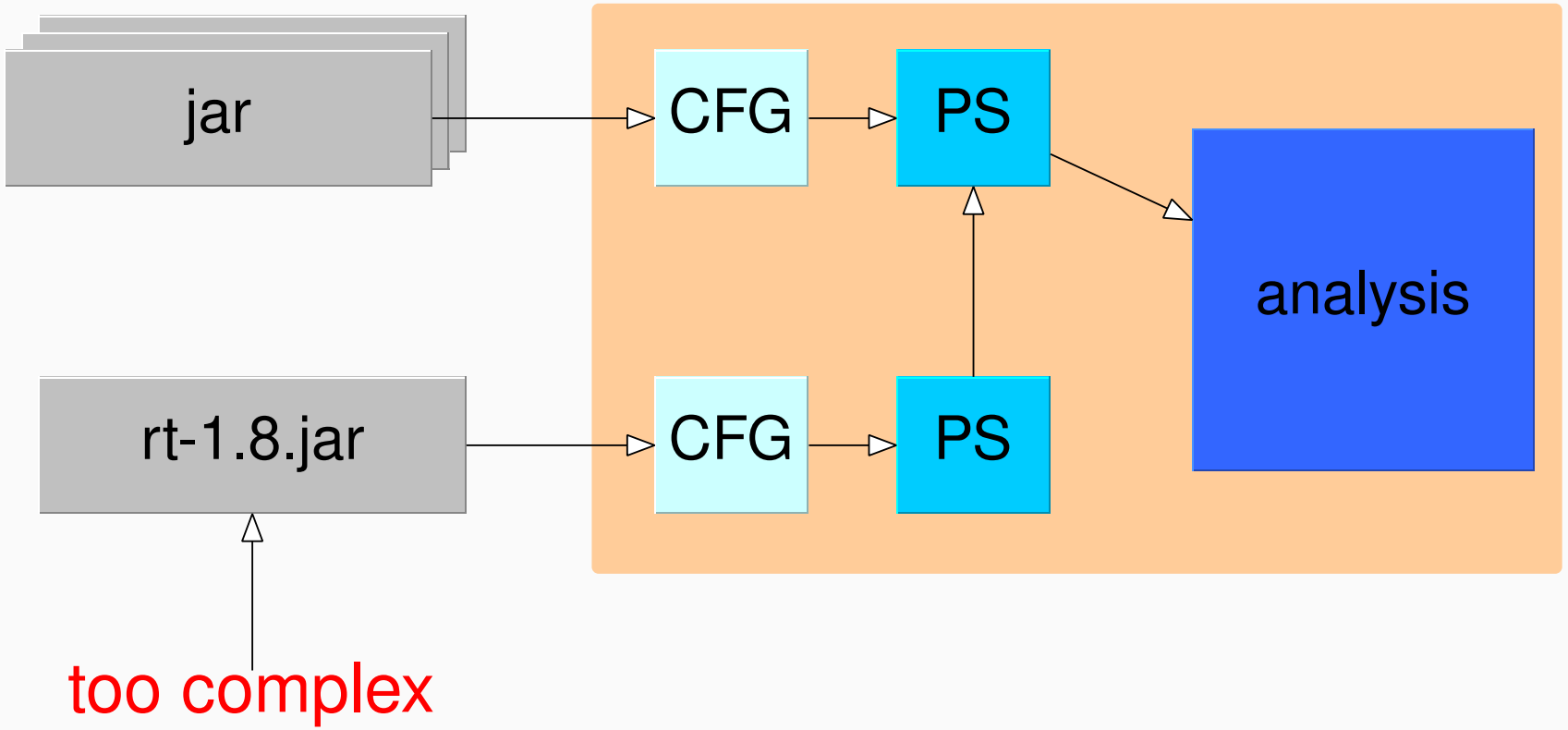
- Kfg and Kex were optimized w.r.t. RAM usage:
  - Kfg currently uses ~2 times less RAM
  - Kex uses only one copy of each classes of PUT
- Kex and Reanimator were extended to support some new language features

	30s	**120s**
`line coverage`	22.31%	26.70%
`branch coverage`	14.69%	17.95%

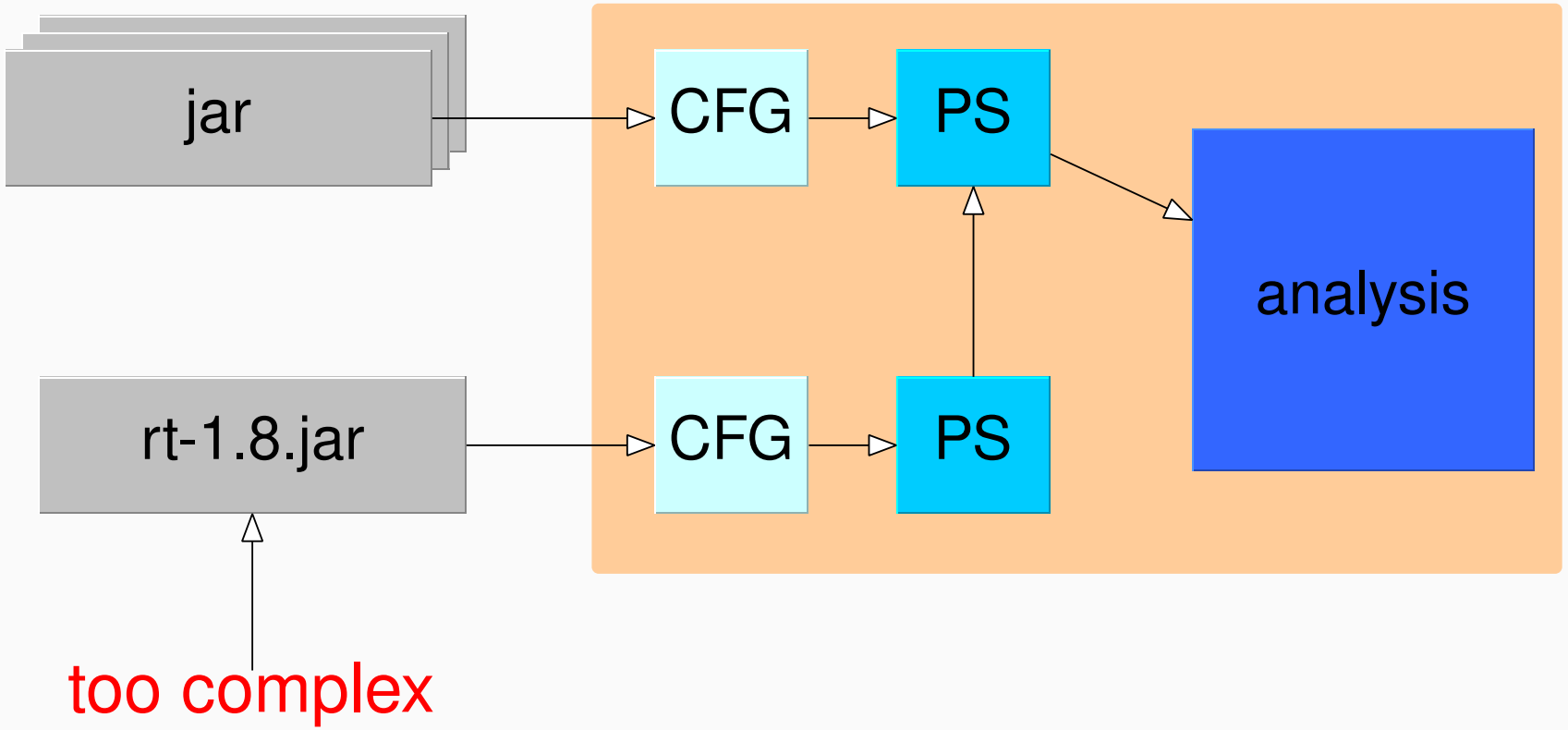
Applied for SBST 2022 competition


CHANGE
AHEAD

Java standard library support at the end of 2020



Java standard library support at the end of 2020



Many of the standard library methods and classes can be approximated

Java rt-1.8 approximarion

Prototype implementation based on `kex-intrinsics`[3]:

- wrappers for primitive types
- string builders
- some collections (based on `ArrayList` approximation)
- utility methods from `Arrays` and `System` classes

Kex substitutes all Java runtime operations with approximated analogs if they are available

[3] https://github.com/vorpal-research/kex-intrinsics

Exmaple of `ArrayList::add` method

```java
@Override
public boolean add(E e) {
  AssertIntrinsics.kexNotNull(elementData);
  int oldLength = elementData.length;
  elementData = CollectionIntrinsics.generateObjectArray(
    oldLength + 1,
    index -> {
      if (index < oldLength) return elementData[index];
      else return null;
  });
  elementData[oldLength] = e;
  return true;
}
```

Support of intrinsics

Predicate State level support:

- Kfg extended to support `invokedynamic`
- PS extended to support lambdas
  - lambdas can't change program state

SMT level support:

- slightly reworked memory model
  - arrays are now represented as SMT arrays
- $\exists$ and $\forall$ quantors for array operations
- λ expressions for array generation
- experimented with SMT string theory (unsuccessfully)

Java standard library support: current state

- prototype implementation
  - limited in expressivness
  - limited number of supported classes
- no thorough evaluation
  - experiments show a small increase in coverage


CHANGE
AHEAD

Reanimator

- an approach to generate valid code snippets using only public API
  - can’t produce invalid objects
- works in reasonable time
- applicable in any automatic test generation tool
- can be used in any programming language

Reanimator at the end of 2020

- working prototype
- evaluation:
  - testing with Kex on open source projects from github
  - testing on random objects
- can successfully and efficiently generate 70% of target objects on average

Problem: evaluation is not representative enough

New evaluation

- integrated Reanimator as an optional test generation module in Tardis[4]
  - extended Tardis model computation
- compared with its default test generator — Evosuite[5]
- used SBST 2021 benchmark for evaluation, but extended time budgets

---

[4] Braione P., Denaro G. SUSHI and TARDIS at the SBST2019 Tool Competition //2019 IEEE/ACM 12th International Workshop on Search-Based Software Testing (SBST). – IEEE, 2019. – C. 25-28.

[5] Fraser G., Arcuri A. Evosuite: automatic test suite generation for object-oriented software //Proceedings of the 19th ACM SIGSOFT symposium. – 2011. – C. 416-419.

Reanimator: current state

	60s	120s	300s	600s
tardis + evosuite	13.96%	15.71%	18.50%	19.60%
tardis + reanimator	13.84%	15.99%	17.84%	19.30%
kex + reanimator	24.57%	25.29%	25.43%	27.61%

Reanimator: what to do

- implement Tardis approach in Kex
- compare both Tardis and Reanimator on random object generation
- improve Reanimator (duh)
- try to publish as is

Plans for the (nearest) future

- SBST 2022
- finish work on Reanimator
  - evaluation and publication
  - integrate Reanimator with concolic mode
- extend standard library support
  - support of state chages in lambdas
  - more classes

Contact information

abdullin@kspt.icc.spbstu.ru

azat.abdullin@jetbrains.com

POLYTECH
Peter the Great
St. Petersburg Polytechnic
University