Kex in 2021: Ups and Downs

Azat Abdullin

December 13, 2021

Kex

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

What happened in 2021

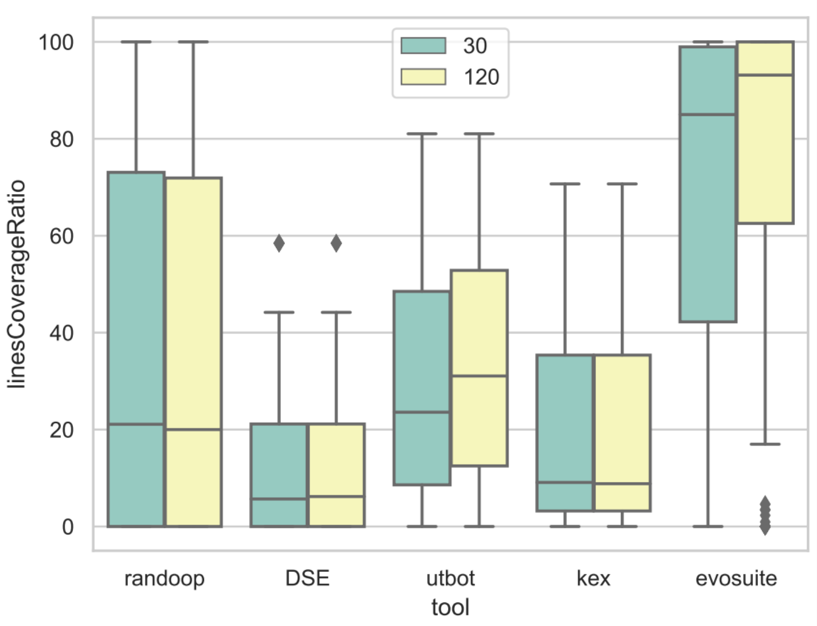
1. Participation in SBST Java tool competition 2021[1]
2. Work towards better Java standard library support
3. Reanimator evaluation

[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
- evaluation on 6 projects with 98 benchmarks
- each tool evaluated on 30 and 120 second time budgets

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
- applied for SBST 2022 Java tool competition

	30s	**120s**
line coverage	21.70%	25.29%
branch coverage	14.69%	17.95%

CHANGE
AHEAD

Java standard library support

- Java standard library is used almost everywhere
- despite having access to standard library sources, Kex struggles to simulate it
- many of the standard library methods and classes can be approximated in SMT

Intrinsics library[3]

Intrinsics for basic operations and checks:

- `assertions` and `assumes`
- `unknown` values with no constraints
- array operations:
  - `contains` checks
  - array generation methods
- etc.

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

kex-rt[4]

Proof-of-concept implementation:

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

Kex substitutes all Java runtime operations with kex-rt analogs if they are available

[4] https://github.com/AbdullinAM/kex-rt

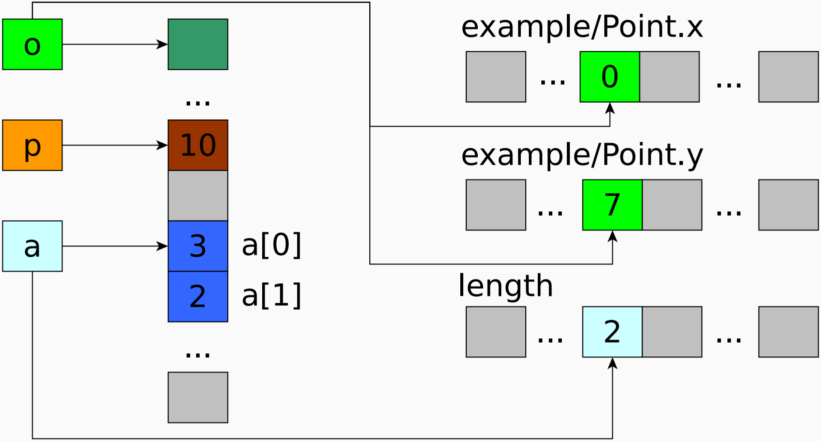
Exmaple of `ArrayList::add` method

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

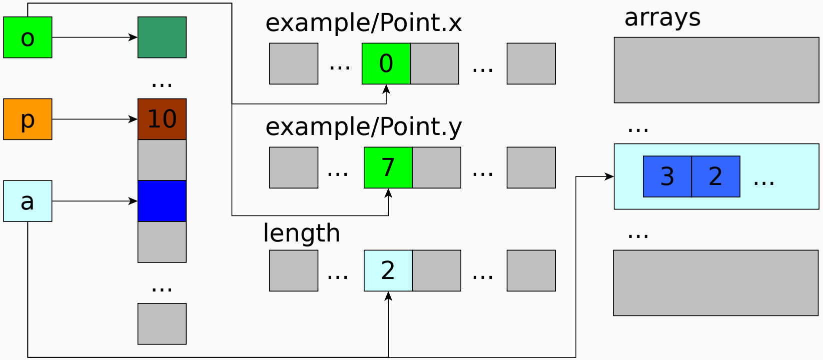
SMT support of intrinsics

- 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)

SMT support of intrinsics



SMT support of intrinsics



Java standard library support: takeaways

- prototype implementation
    - limited in expressivness
    - limited number of supported classes
- no thorough evaluation

CHANGE
AHEAD

Reanimator

Contact information

abdullin@kspt.icc.spbstu.ru

JET BRAINS RESEARCH

POLYTECH
Peter the Great
St. Petersburg Polytechnic
University