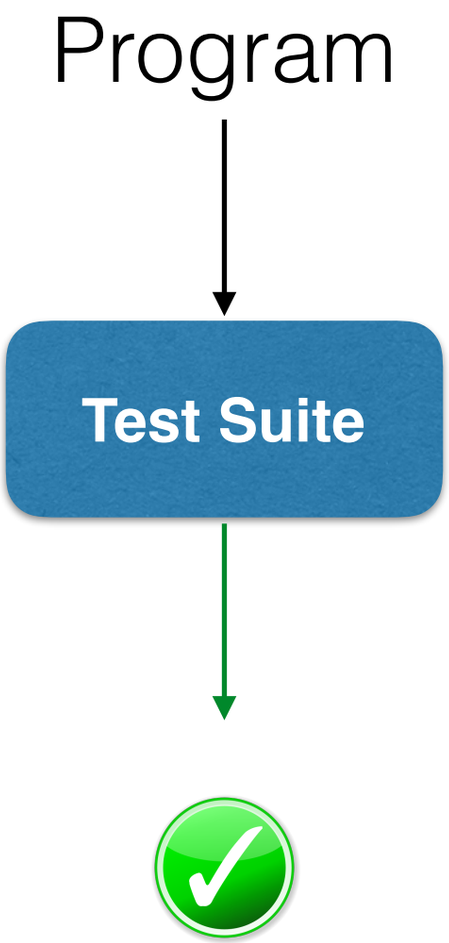
Verifying Determinism in Sequential Programs

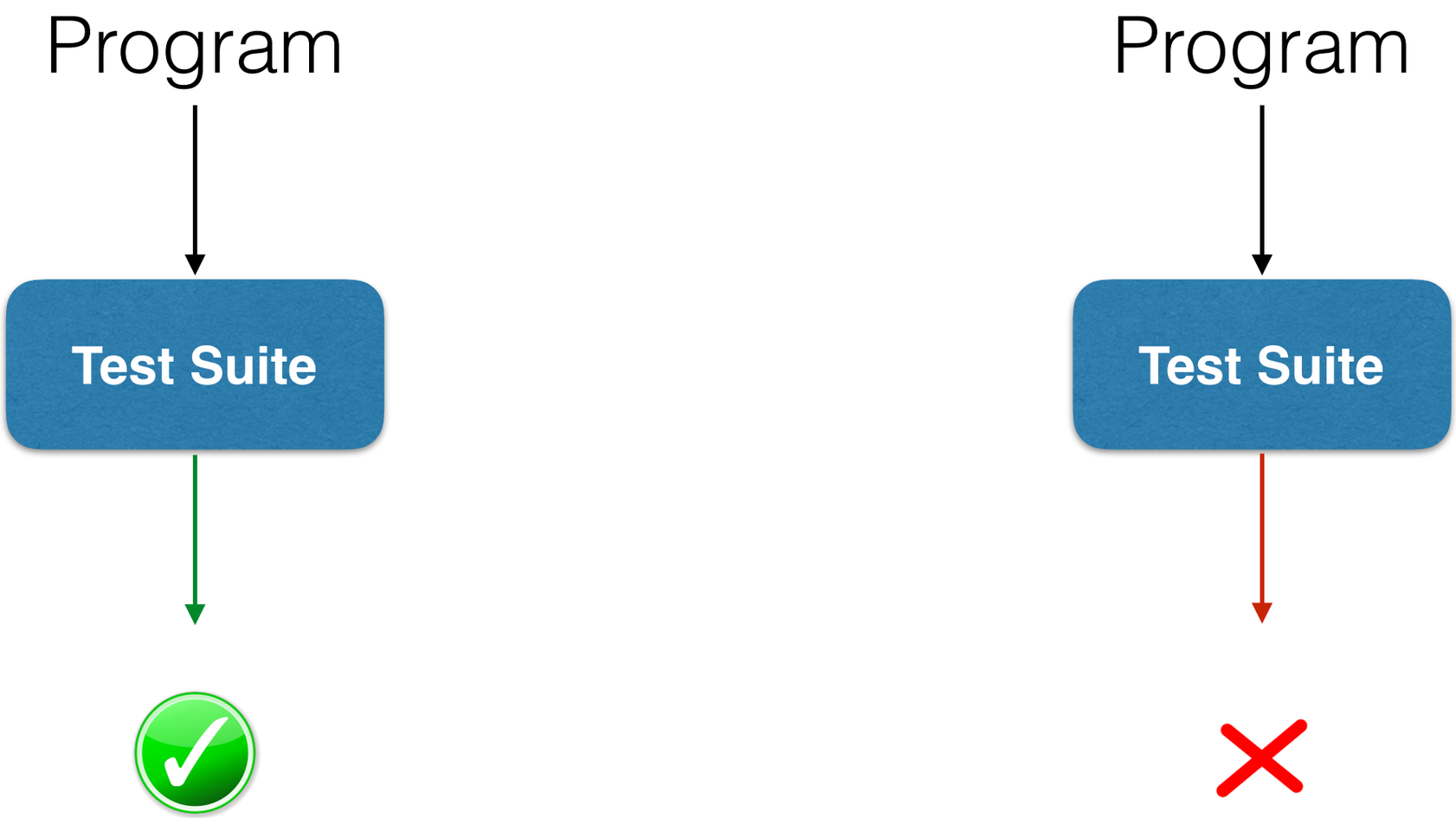
Rashmi Mudduluru, Jason Waataja, Suzanne Millstein, Michael Ernst

University of Washington

Testing Scenario

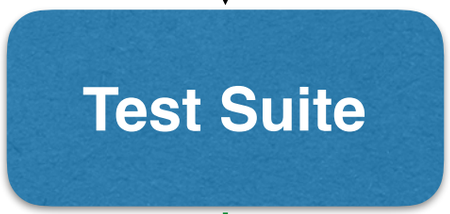


Testing Scenario



Testing Scenario

Program



Program

Sources of Nondeterminism in Sequential Programs

- Iterating over a hash table
- Date and time functions
- Accessing system properties (file system, env)
- Coin flipping

Example from Randoop

```
public List<TypeVariable> getTypeParameters() {
  Set<TypeVariable> parameters = new
          HashSet<>(super.getTypeParameters());
  …
  return new ArrayList<>(parameters);
}


public String toString(TypedClassOperation this) {
  return join(",", this.getTypeParameters());
}
```

Example from Randoop

```
public List<TypeVariable> getTypeParameters() {
  Set<TypeVariable> parameters = new
          HashSet<>(super.getTypeParameters());
  …
  return new ArrayList<>(parameters);
}
```

```
public String toString(TypedClassOperation this) {
  return join(",", this.getTypeParameters());
}
```

toString should be deterministic.

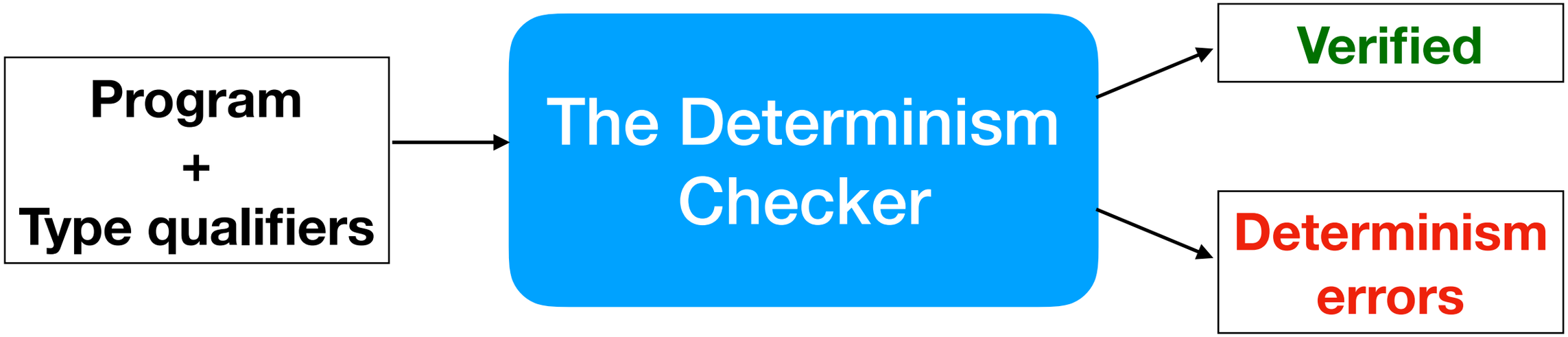
Goal

- ***Specify*** determinism properties in a program.
- ***Verify*** these properties statically and ***soundly***.
  - No warnings **—>** same output across executions.
- Permit nondeterminism where specification allows it.
  - Example: Use of hash tables in a method that tests membership.

Type System Based Solution

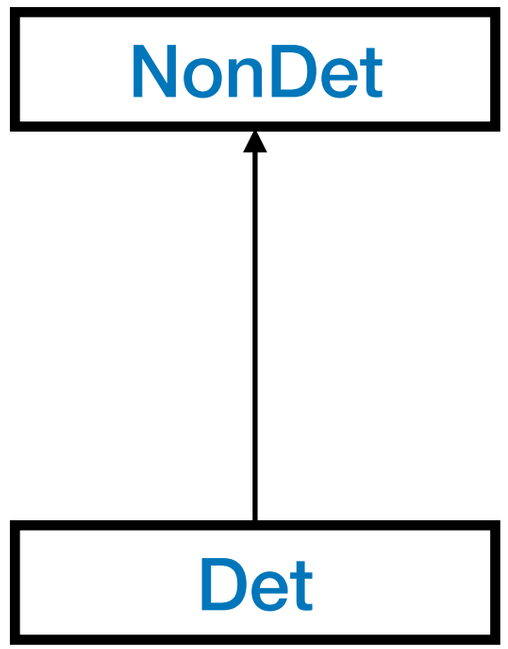
- ***Type qualifiers*** express determinism.
- Type ***well-formedness rules*** restrict Collection types.
- Enhancements to ***polymorphism*** improve precision.

Workflow


Program
+
Type qualifiers
The Determinism Checker
Verified
Determinism errors

Type Qualifiers

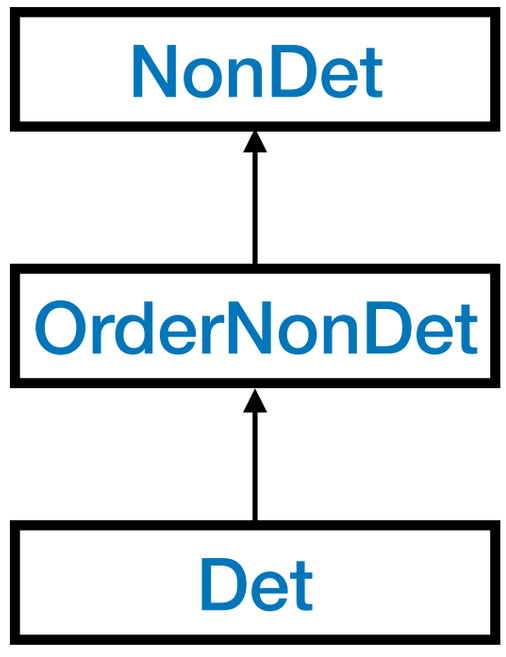
- **NonDet:** values that might differ across runs.



- **Det:** values that are the same across runs.

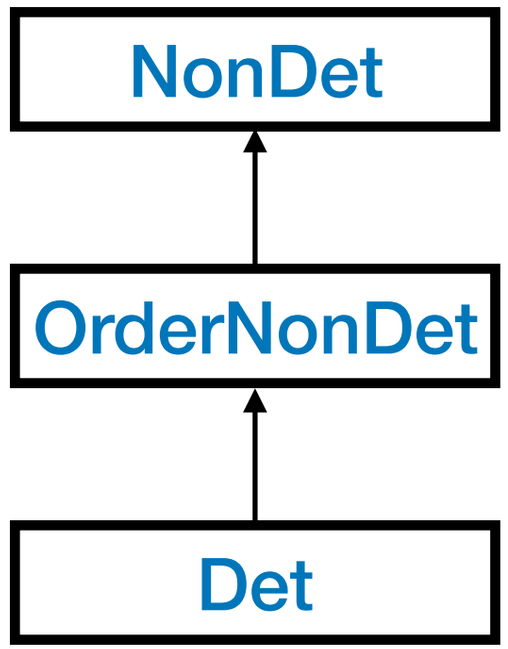
Type Qualifiers

- **NonDet:** values that might differ across runs.
- **OrderNonDet:** ***Collections*** containing
  - same elements across runs.
  - iteration order might differ.
- **Det:** values that are the same across runs.



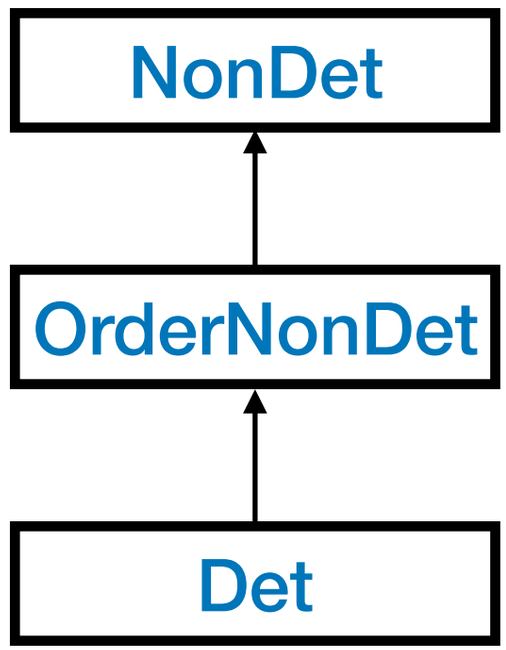
Collection Type Well-formedness

- Determinism is a ***deep*** property:
  - If a Collection is **NonDet**, so are its elements.



Collection Type Well-formedness

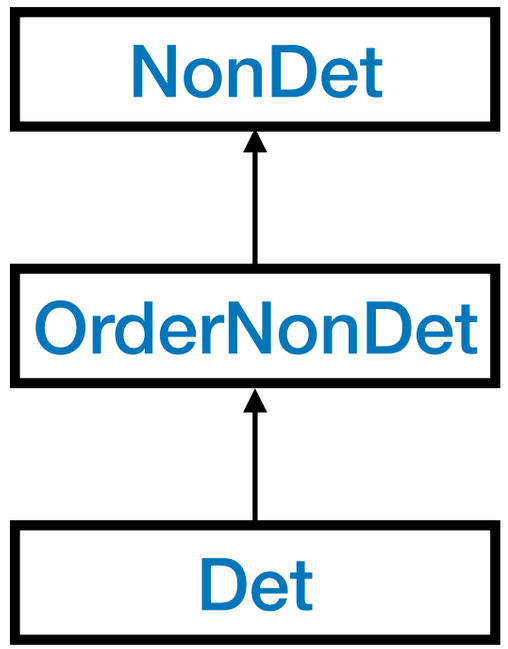
- Determinism is a ***deep*** property:
  - If a Collection is **NonDet**, so are its elements.



- Examples:
  - **OrderNonDet** List<**Det** String>
  - **Det** List<**NonDet** String>

Polymorphism

A polymorphic type expresses multiple overloaded definitions.

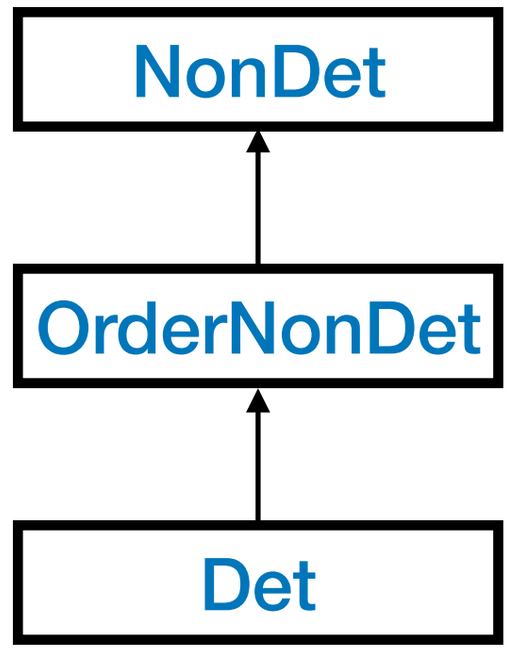


Enhancements to Polymorphism

A polymorphic type expresses multiple overloaded definitions.

Example: Type of List.get method:

- **NonDet** List<E> × **NonDet** int ⟶ **NonDet** E



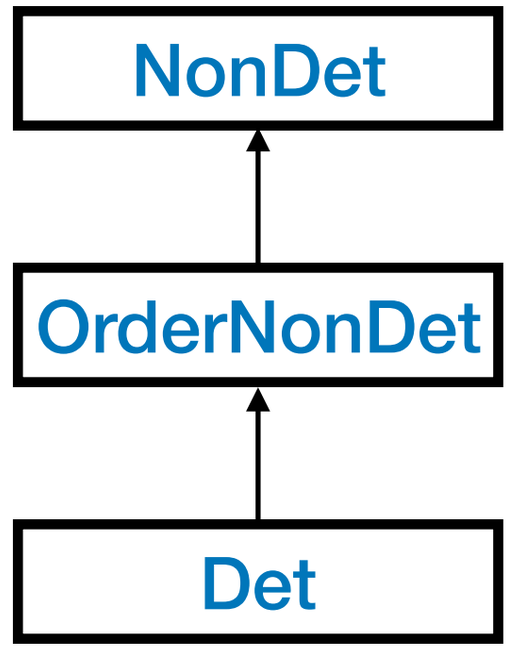
Enhancements to Polymorphism

A polymorphic type expresses multiple overloaded definitions.

Example: Type of List.get method:

- **NonDet** List<E> × **NonDet** int ⟶ **NonDet** E

- **Det** List<E> × **Det** int ⟶ **Det** E

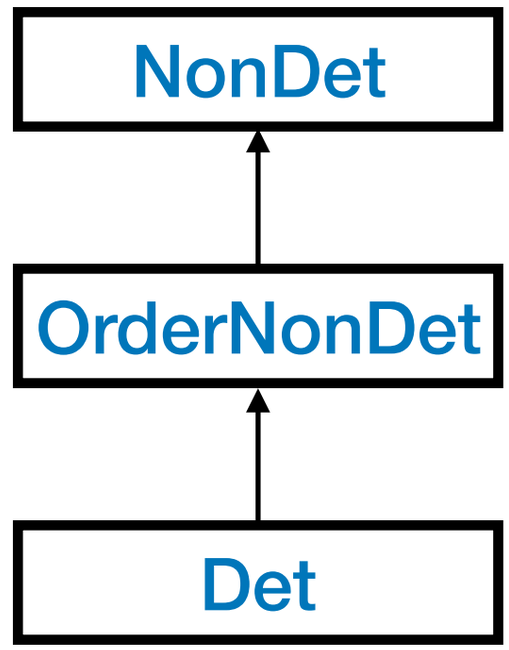


Enhancements to Polymorphism

A polymorphic type expresses multiple overloaded definitions.

Example: Type of List.get method:

- **NonDet** List<E> × **NonDet** int ⟶ **NonDet** E



- **Det** List<E> × **Det** int ⟶ **Det** E

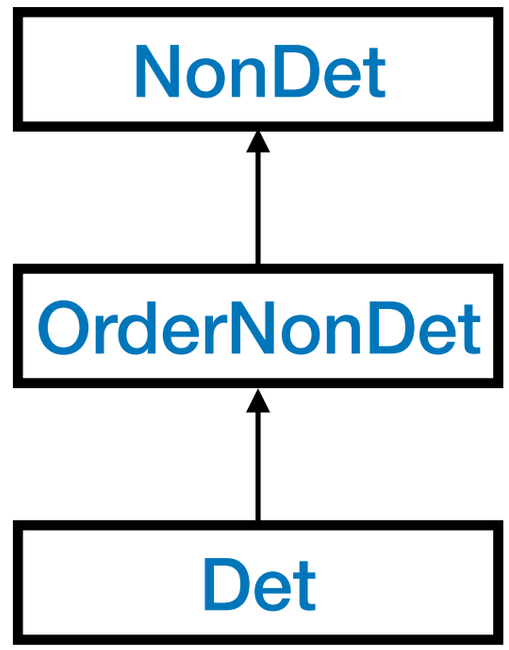
PolyDet E get(**PolyDet** List<E> this, **PolyDet** int index)

Enhancements to Polymorphism

A polymorphic type expresses multiple overloaded definitions.

Example: Type of List.get method:

- **NonDet** List<E> × **NonDet** int ⟶ **NonDet** E
- **OrderNonDet** List<E> ×**Det** int ⟶**NonDet** E
- **Det** List<E> × **Det** int ⟶ **Det** E



PolyDet(up) E get(**PolyDet** List<E> this, **PolyDet** int index)

Treats **OrderNonDet** as **NonDet**

Experiments: Case Studies

Project	LoC
Randoop	23849
Checkstyle	36182
CF dataflow	13235
Plume lib	15220
Total	**88486**

Experiments: Case Studies

Project	LoC	Bugs
Randoop	23849	15
Checkstyle	36182	13
CF dataflow	13235	43
Plume lib	15220	16
Total	**88486**	**87**

Fixed by developers.

Experiments: Case Studies

Project	LoC	Bugs	#Warning Suppressions
Randoop	23849	15	136
Checkstyle	36182	13	84
CF dataflow	13235	43	70
Plume lib	15220	16	463
Total	**88486**	**87**	**753**

Order insensitive algorithms.

```
@SuppressWarnings("determinism:return.type.incompatible") // process is order insensitive
private static Det Field fieldForName(Class<?> type, String fieldName) {
  for (NonDet Field f : type.getDeclaredFields()) {
    if (fieldName.equals(f.getName())) {
      return f;
    }
  }
  return null;
}
```

Experiments: Case Studies

Project	LoC	Bugs	#Warning Suppressions	#Annotations
Randoop	23849	15	136	3103
Checkstyle	36182	13	84	513
CF dataflow	13235	43	70	1277
Plume lib	15220	16	463	1285
Total	**88486**	**87**	**753**	**6178**

Annotations: ~1 per 17 LoC.
Serve as machine-checked documentation.
Found issues that developers did not.

Related Work

- NonDex [Shi et al, ICST 2016]:
  - Authors built models for 25 nondeterministic JDK methods.
  - Requires modifications to JVM.

Related Work

- NonDex [Shi et al, ICST 2016]:
  - Authors built models for 25 nondeterministic JDK methods.
  - Requires modifications to JVM.
- Deflaker [Bell et al, ICSE 2018]:
  - Depends on version control history.
  - Uses test suite, code coverage to detect flaky tests.

Related Work

- NonDex [Shi et al, ICST 2016]:
  - Authors built models for 25 nondeterministic JDK methods.
  - Requires modifications to JVM.
- Deflaker [Bell et al, ICSE 2018]:
  - Depends on version control history.
  - Uses test suite, code coverage to detect flaky tests.
- These run-time approaches are ***unsound***.

Comparison to Previous Work

- Determinism Checker on case studies of NonDex and DeFlaker:
  - Found ***all*** non-concurrency bugs.
  - Found 13 ***additional*** bugs in Checkstyle.
- NonDex and DeFlaker on our case studies.
  - Found ***none*** of the bugs.

Research Contributions

- The ***first sound compile-time*** analysis of nondeterminism.
- A type system for ***expressing*** and ***verifying*** determinism properties.
- Enhanced ***polymorphism*** to improve ***expressivity***.
- Found ***87*** determinism bugs in ***real-world projects***.
- Tool: https://github.com/t-rasmud/checker-framework/tree/nondet-checker

Nondeterminism due to Concurrency

- Previous work has addressed concurrency nondeterminism.
  - DeFlaker: 4/7 (57%) are concurrency bugs.
  - Verify hyperproperties - properties about multiple program executions [Eilers et al, ACM Trans Program Lang Sys, 2019],
  - Nondeterminism in MapReduce programs [Xiao et. al, ICSE 2014]
- Our work is complimentary and addresses previously. overlooked problem.