New Test Framework with IR Verification

Overview

Christian Hagedorn
Staff Meeting
February 2021

New Test Framework with IR Verification

- Why?
  - Cannot verify IR automatically

```
int iFld;

void test() {
  iFld = 2; // Removed?
  iFld = 3;
}
```

New Test Framework with IR Verification

- Why?
  - Cannot verify IR automatically
- Inspired by Valhalla's test framework
  - Found and prevented many bugs
  - New framework includes all features added over the years

```
int iFld;

void test() {
  iFld = 2; // Removed?
  iFld = 3;
}
```

New Test Framework with IR Verification

- Why?
  - Cannot verify IR automatically
- Inspired by Valhalla's test framework
  - Found and prevented many bugs
  - New framework includes all features added over the years
- Easy to use
  - Simple interface, no knowledge of internals required
  - Annotations
  - Informal error reporting (wrong test format, failed IR rules/assertions, …)

```
int iFld;

void test() {
  iFld = 2; // Removed?
  iFld = 3;
}
```

New Test Framework with IR Verification

- Why?
  - Cannot verify IR automatically
- Inspired by Valhalla's test framework
  - Found and prevented many bugs
  - New framework includes all features added over the years
- Easy to use
  - Simple interface, no knowledge of internals required
  - Annotations
  - Informal error reporting (wrong test format, failed IR rules/assertions, …)
- No changes to HotSpot code

```
int iFld;

void test() {
  iFld = 2; // Removed?
  iFld = 3;
}
```

How does it work?

VM1

Test.java

```
public static void main(String[] args) {
  TestFramework.run(Test.class);
}

int iFld;

@Test
@IR(counts = {IRNode.STORE, “1”})
public void test() {
  iFld = 2; // Must be removed!
  iFld = 3;
}
```

TestFramework.java

How does it work?

VM1

Test.java

```
public static void main(String[] args) {
  TestFramework.run(Test.class);
}

int iFld;

@Test
@IR(counts = {IRNode.STORE, “1”})
public void test() {
  iFld = 2; // Must be removed!
  iFld = 3;
}
```

TestFramework.java

- Create new VM instance with additional print flags (-XX:+PrintIdeal, -XX:+PrintOptoAssembly, …)

VM2

TestFramework.java

- Warm up test()
- Compile test()
- Invoke test() again

How does it work?

VM1

Test.java

```
public static void main(String[] args) {
  TestFramework.run(Test.class);
}

int iFld;

@Test
@IR(counts = {IRNode.STORE, "1"})
public void test() {
  iFld = 2; // Must be removed!
  iFld = 3;
}
```

TestFramework.java

- Create new VM instance with additional print flags (-XX:+PrintIdeal, -XX:+PrintOptoAssembly, ...)

VM2

TestFramework.java

- Warm up test()
- Compile test()
- Invoke test() again

- Parse **VM2** output to match @IR rule(s)
  - IR matching on PrintIdeal or PrintOptoAssembly output
  - Throw exception if not matched

How to Use the Framework – Different Tests

MyBaseTest.java

```
@Test
public void baseTest() { }


@Test
@Arguments(Argument.DEFAULT)
public void baseTestWithArgs(int x) { }


@Test
@Arguments(Argument.BOOLEAN_TOGGLE)
public void baseTestWithArgs2(boolean x) { }


@Test
@Warmup(0)
public void xcomp() { }


@Test(compLevel = CompLevel.C1_FULL_PROFILE)
public void c1Full() { }


@DontInline
@ForceCompile
public void helperMethod() { }
```

How to Use the Framework – Different Tests

MyBaseTest.java

```
@Test
public void baseTest() { }


@Test
@Arguments(Argument.DEFAULT)
public void baseTestWithArgs(int x) { }


@Test
@Arguments(Argument.BOOLEAN_TOGGLE)
public void baseTestWithArgs2(boolean x) { }


@Test
@Warmup(0)
public void xcomp() { }


@Test(compLevel = CompLevel.C1_FULL_PROFILE)
public void c1Full() { }


@DontInline
@ForceCompile
public void helperMethod() { }
```

MyCheckedTest.java

```
@Test
public int foo() { return 3; }


@Check(test = “foo”)
public void checkFoo(int returnValue) {
  if (returnValue != 3) { /* Throw error */ }
}
```

How to Use the Framework – Different Tests

MyBaseTest.java

```
@Test
public void baseTest() { }


@Test
@Arguments(Argument.DEFAULT)
public void baseTestWithArgs(int x) { }


@Test
@Arguments(Argument.BOOLEAN_TOGGLE)
public void baseTestWithArgs2(boolean x) { }


@Test
@Warmup(0)
public void xcomp() { }


@Test(compLevel = CompLevel.C1_FULL_PROFILE)
public void c1Full() { }


@DontInline
@ForceCompile
public void helperMethod() { }
```

MyCheckedTest.java

```
@Test
public int foo() { return 3; }


@Check(test = “foo”)
public void checkFoo(int returnValue) {
  if (returnValue != 3) { /* Throw error */ }
}
```

MyCustomRunTest.java

```
@Test
public int foo(MyObject myObject) {
  /* Do something with myObject */
}


@Run(test = “foo”)
public void runFoo(TestInfo info) {
  MyObject obj = init(info.isWarmup());
  int retValue = foo(obj);
  verify(retValue);
}
```

How to Use the Framework – IR Matching

- Multiple @IR rules/assertions allowed
- Regex matching (PrintIdeal/PrintOptoAssembly)
- FailOn: Fail if node found in IR
- Counts: Constraints on number of nodes
  - Allows comparison operators (=, !=, >, >=, <, <=)

MyIRTest.java

```
int iFld;

@Test
@IR(failOn = {IRNode.LOAD, IRNode.LOOP})
@IR(counts = {IRNode.STORE, “1”,
              IRNode.STORE_OF_FIELD, “iFld”, “1”})
@IR(counts = {IRNode.STORE, “< 2”})
public void goodTest() {
  iFld = 2; // Must be removed!
  iFld = 3;
}
```

How to Use the Framework – IR Matching

- Multiple @IR rules/assertions allowed
- Regex matching (PrintIdeal/PrintOptoAssembly)
- FailOn: Fail if node found in IR
- Counts: Constraints on number of nodes
  - Allows comparison operators (=, !=, >, >=, <, <=)
- Constraints when rule applied
  - Based on VM flags
  - applyIf, applyIfNot, applyIfAnd, applyIfOr
  - Allows comparison operators (=, !=, >, >=, <, <=)

MyIRTest.java

```
int iFld;

@Test
@IR(failOn = {IRNode.LOAD, IRNode.LOOP})
@IR(counts = {IRNode.STORE, “1”,
              IRNode.STORE_OF_FIELD, “iFld”, “1”})
@IR(counts = {IRNode.STORE, “< 2”})
public void goodTest() {
  iFld = 2; // Must be removed!
  iFld = 3;
}

@Test
@IR(applyIf    = {“LoopUnrollLimit”, “0”},
    counts     = {IRNode.STORE, “1”})
@IR(applyIfAnd = {“LoopUnrollLimit”, “< 8”,
                  “UseZGC”,          “true”},
    counts     = {IRNode.STORE, “1”})
public void goodTest2() {
  iFld = 2; // Must be removed!
  iFld = 3;
}
```

How to Use the Framework – IR Matching

- Multiple @IR rules/assertions allowed
- Regex matching (PrintIdeal/PrintOptoAssembly)
- FailOn: Fail if node found in IR
- Counts: Constraints on number of nodes
  - Allows comparison operators (=, !=, >, >=, <, <=)
- Constraints when rule applied
  - Based on VM flags
  - applyIf, applyIfNot, applyIfAnd, applyIfOr
  - Allows comparison operators (=, !=, >, >=, <, <=)

MyIRTest.java

```
int iFld;

@Test
@IR(failOn = {IRNode.LOAD, IRNode.LOOP})
@IR(counts = {IRNode.STORE, “1”,
              IRNode.STORE_OF_FIELD, “iFld”, “1”})
@IR(counts = {IRNode.STORE, “< 2”})
public void goodTest() {
  iFld = 2; // Must be removed!
  iFld = 3;
}

@Test
@IR(applyIf    = {“LoopUnrollLimit”, “0”},
    counts     = {IRNode.STORE, “1”})
@IR(applyIfAnd = {“LoopUnrollLimit”, “< 8”,
                  “UseZGC”,          “true”},
    counts     = {IRNode.STORE, “1”})
public void goodTest2() {
  iFld = 2; // Must be removed!
  iFld = 3;
}
```

Current State and Next Steps

- Support everything from Valhalla's test framework (almost done)
  - Some IR nodes still missing

Current State and Next Steps

- Support everything from Valhalla's test framework (almost done)
  - Some IR nodes still missing
- Convert Valhalla's tests (> 800) to use new framework
- More tests for testing the framework itself
  - Especially "bad tests" (e.g. wrong usage, failing @IR tests, …)
  - Example/reference tests for how to use the framework

Open Questions

- Which IR nodes should be supported by default?
- Documentation (example tests, Javadocs, Markdown, …)
- Jtreg support (e.g. @TestFramework)
- Add additional verification (e.g. check output of more flags, …)
- Where does it live?