Accurate Interprocedural Null-Dereference Analysis for Java

Mangala Gowri Nanda and Saurabh Sinha

{mgowri,saurabhsinha}@in.ibm.com.

IBM India Research Lab

Interprocedural Null-Dereference

```
scanDir(File srcDir, File dest, JspMangler mangler, String files[]) {
  [499]  if (i < files.length)
  [500]  String filename = files[i];
  [501]  File srcFile = new File(srcDir, filename);
  [502]  javaFile = mapToJavaFile(mangler, srcFile, srcDir, dest);

  [509]  shouldCompile = isCompileNeeded(srcFile, javaFile);
}
mapToJavaFile(JspMangler mangler, File srcFile, File srcDir, File dest) {
  [565]  if (!srcFile.getName().endsWith(".jsp"))
  [566]  return null
}
isCompileNeeded(File srcFile, File javaFile) {
  [535]  javaFile.exists()
}
```

Interprocedural Null-Dereference

```
scanDir(File srcDir, File dest, JspMangler mangler, String files[]) {
  [499]  if (i < files.length)
  [500]  String filename = files[i];
  [501]  File srcFile = new File(srcDir, filename);
  [502]  javaFile = mapToJavaFile(mangler, srcFile, srcDir, dest);

  [509]  shouldCompile = isCompileNeeded(srcFile, javaFile);
}
mapToJavaFile(JspMangler mangler, File srcFile, File srcDir, File dest) {
  [565]  if (!srcFile.getName().endsWith(".jsp"))
  [566]  return null
}
isCompileNeeded(File srcFile, File javaFile) {
  [535]  javaFile.exists()
}
```

Interprocedural Null-Dereference

```
scanDir(File srcDir, File dest, JspMangler mangler, String files[]) {
  [499]  if (i < files.length)
  [500]  String filename = files[i];
  [501]  File srcFile = new File(srcDir, filename);
  [502]  javaFile = mapToJavaFile(mangler, srcFile, srcDir, dest);

  [509]  shouldCompile = isCompileNeeded(srcFile, javaFile);
}
mapToJavaFile(JspMangler mangler, File srcFile, File srcDir, File dest) {
  [565]  if (!srcFile.getName().endsWith(".jsp"))
  [566]  return null
}
isCompileNeeded(File srcFile, File javaFile) {
  [535]  javaFile.exists()
}
```

Interprocedural Null-Dereference

```
scanDir(File srcDir, File dest, JspMangler mangler, String files[]) {
  [499]  if (i < files.length)
  [500]  String filename = files[i];
  [501]  File srcFile = new File(srcDir, filename);
  [502]  javaFile = mapToJavaFile(mangler, srcFile, srcDir, dest);

  [509]  shouldCompile = isCompileNeeded(srcFile, javaFile);
}
mapToJavaFile(JspMangler mangler, File srcFile, File srcDir, File dest) {
  [565]  if (!srcFile.getName().endsWith(".jsp"))
  [566]  return null
}
isCompileNeeded(File srcFile, File javaFile) {
  [535]  javaFile.exists()
}
```

Interprocedural Null-Dereference

```
scanDir(File srcDir, File dest, JspMangler mangler, String files[]) {
  [499]  if (i < files.length)
  [500]  String filename = files[i];
  [501]  File srcFile = new File(srcDir, filename);
  [502]  javaFile = mapToJavaFile(mangler, srcFile, srcDir, dest);

  [509]  shouldCompile = isCompileNeeded(srcFile, javaFile );
}
mapToJavaFile(JspMangler mangler, File srcFile, File srcDir, File dest) {
  [565]  if (!srcFile.getName().endsWith(".jsp"))
  [566]  return null
}
isCompileNeeded(File srcFile, File javaFile ) {
  [535]  javaFile.exists()
}
```

Interprocedural Null-Dereference

```
scanDir(File srcDir, File dest, JspMangler mangler, String files[]) {
  [499]  if (i < files.length)
  [500]  String filename = files[i];
  [501]  File srcFile = new File(srcDir, filename);
  [502]  javaFile = mapToJavaFile(mangler, srcFile, srcDir, dest);

  [509]  shouldCompile = isCompileNeeded(srcFile, javaFile );
}
mapToJavaFile(JspMangler mangler, File srcFile, File srcDir, File dest) {
  [565]  if (!srcFile.getName().endsWith(".jsp"))
  [566]  return null
}
isCompileNeeded(File srcFile, File javaFile ) {
  [535]  javaFile.exists()
}
```

Motivation

Motivation

- **Null dereference is one of the most commonly occurring bug in Java programs**

Motivation

- **Null dereference is one of the most commonly occurring bug in Java programs**
- **Such bugs often involve interactions among multiple procedures**

Motivation

- **Null dereference is one of the most commonly occurring bug in Java programs**
- **Such bugs often involve interactions among multiple procedures**
- **Yet, widely used Java static-analysis tools, such as FindBugs, JLINT and ESC/JAVA perform limited interprocedural analysis**

Motivation

- Null dereference is one of the most commonly occurring bug in Java programs
- Such bugs often involve interactions among multiple procedures
- Yet, widely used Java static-analysis tools, such as FindBugs, JLINT and ESC/JAVA perform limited interprocedural analysis
- Salsa performs sound interprocedural analysis

Motivation

- Null dereference is one of the most commonly occurring bug in Java programs
- Such bugs often involve interactions among multiple procedures
- Yet, widely used Java static-analysis tools, such as FindBugs, JLINT and ESC/JAVA perform limited interprocedural analysis
- Salsa performs sound interprocedural analysis
- Tomb et al apply interprocedural analysis but limit the call depth explored

Contributions

- **A context- and path-sensitive interprocedural analysis for identifying potential null dereferences that is demand-driven and parametrized for cost-accuracy trade-offs**

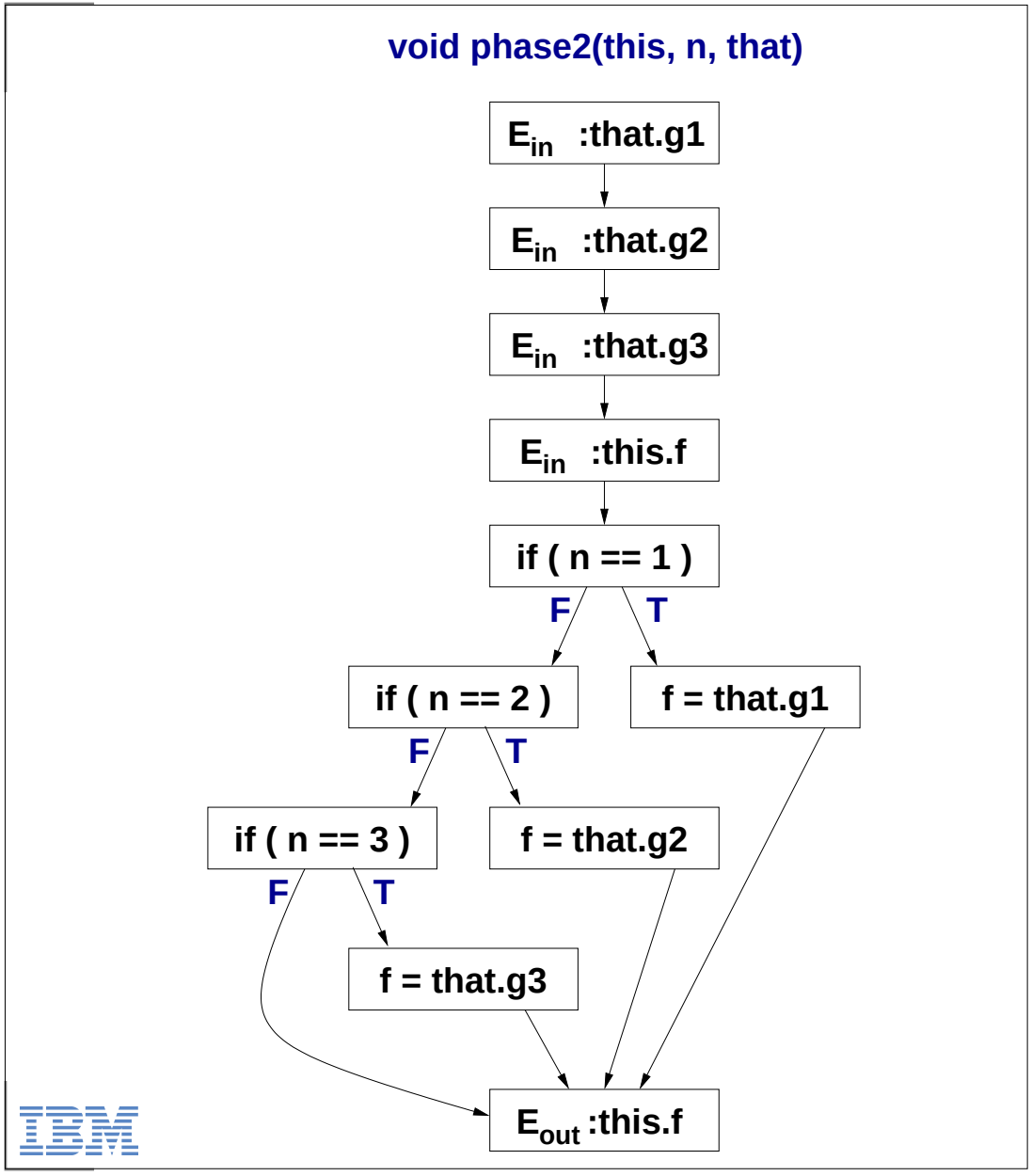
Contributions

- **A context- and path-sensitive interprocedural analysis for identifying potential null dereferences that is demand-driven and parametrized for cost-accuracy trade-offs**
- **Empirical studies, using large open-source and commercial products, that illustrate the effectiveness, efficiency, and usefulness of our approach.**

Outline

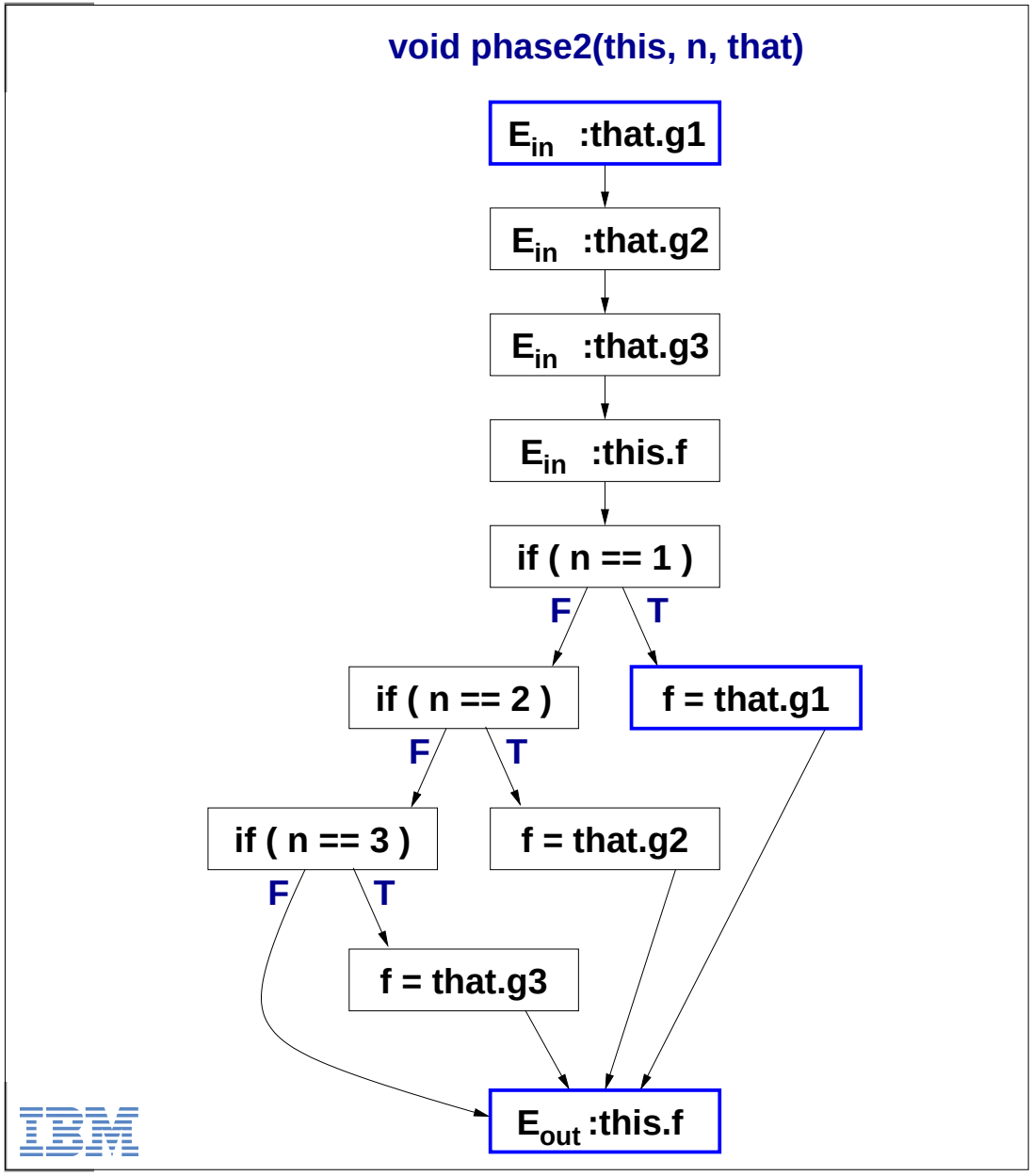
- Predicates - generation, transformation, states etc.
- Interprocedural analysis
- Scaling the analysis
  - Parameterized exploration
  - Simplified predicate handling
- Empirical evaluation
- Conclusion

Preliminaries



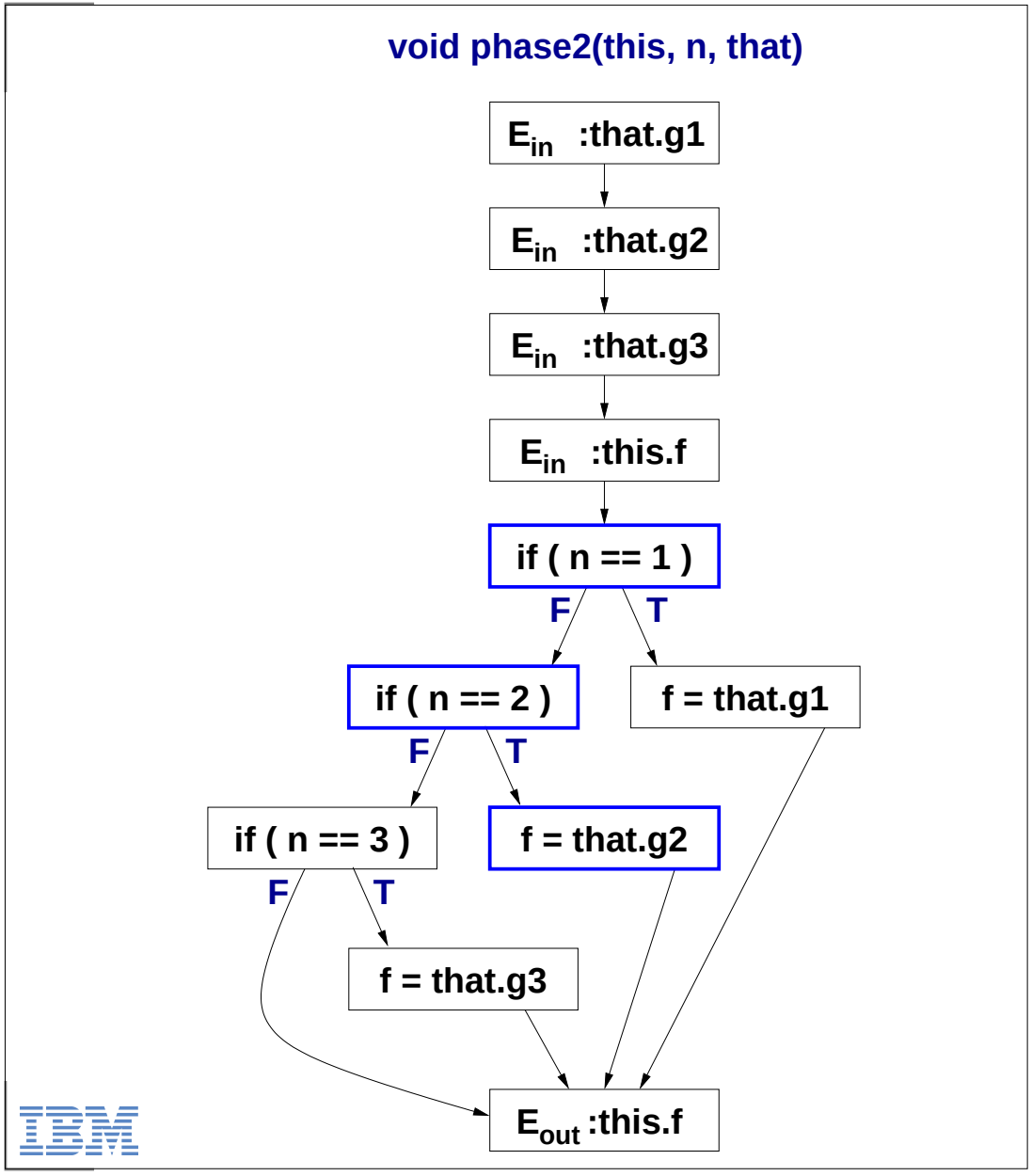
- **Control Flow Graph with Formalin and Formalout**
- **Escape and Pointer analysis**
- **Control dependence analysis**

Preliminaries



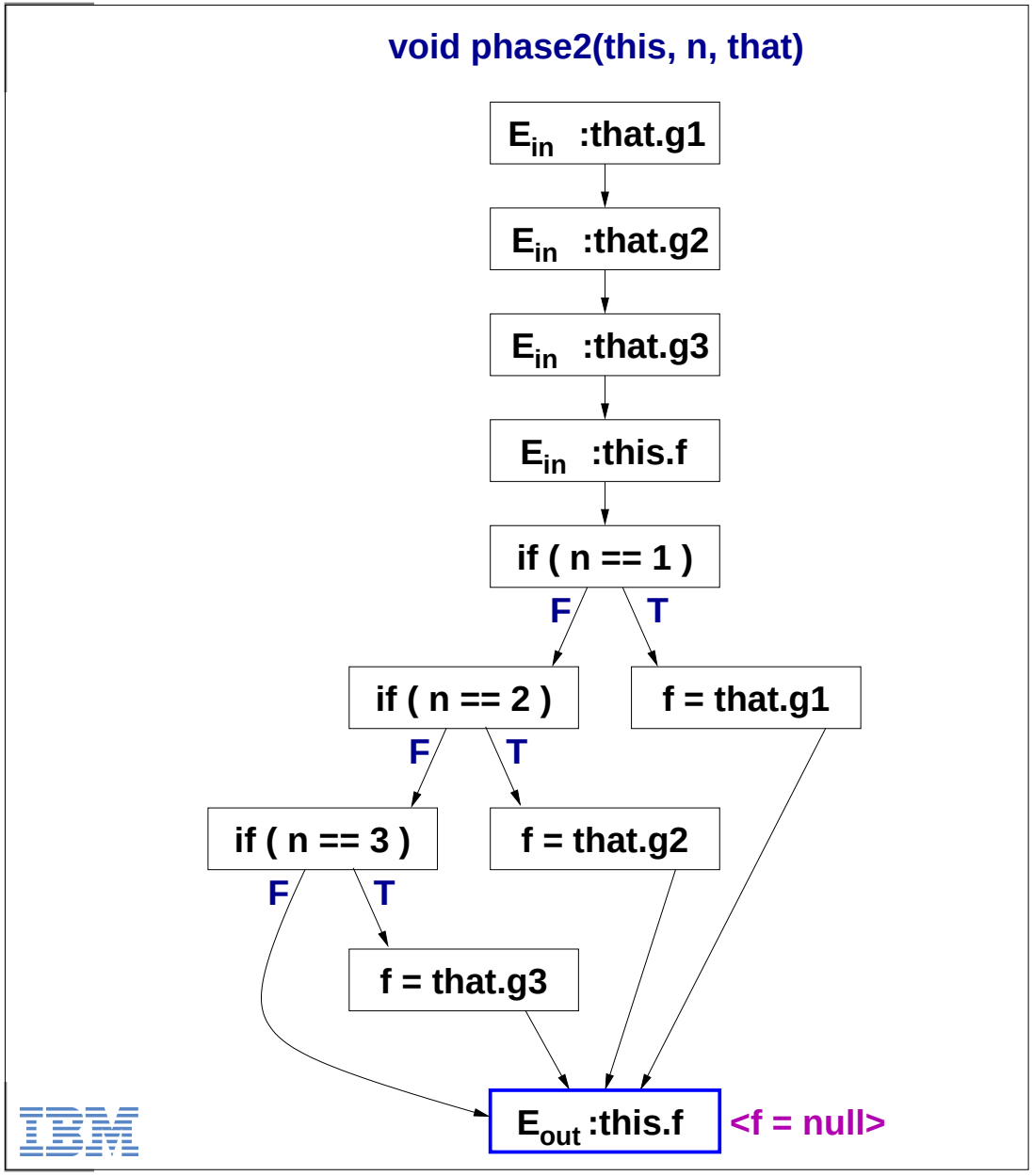
- Control Flow Graph with Formalin and Formalout
- **Escape and Pointer analysis**
- Control dependence analysis

Preliminaries



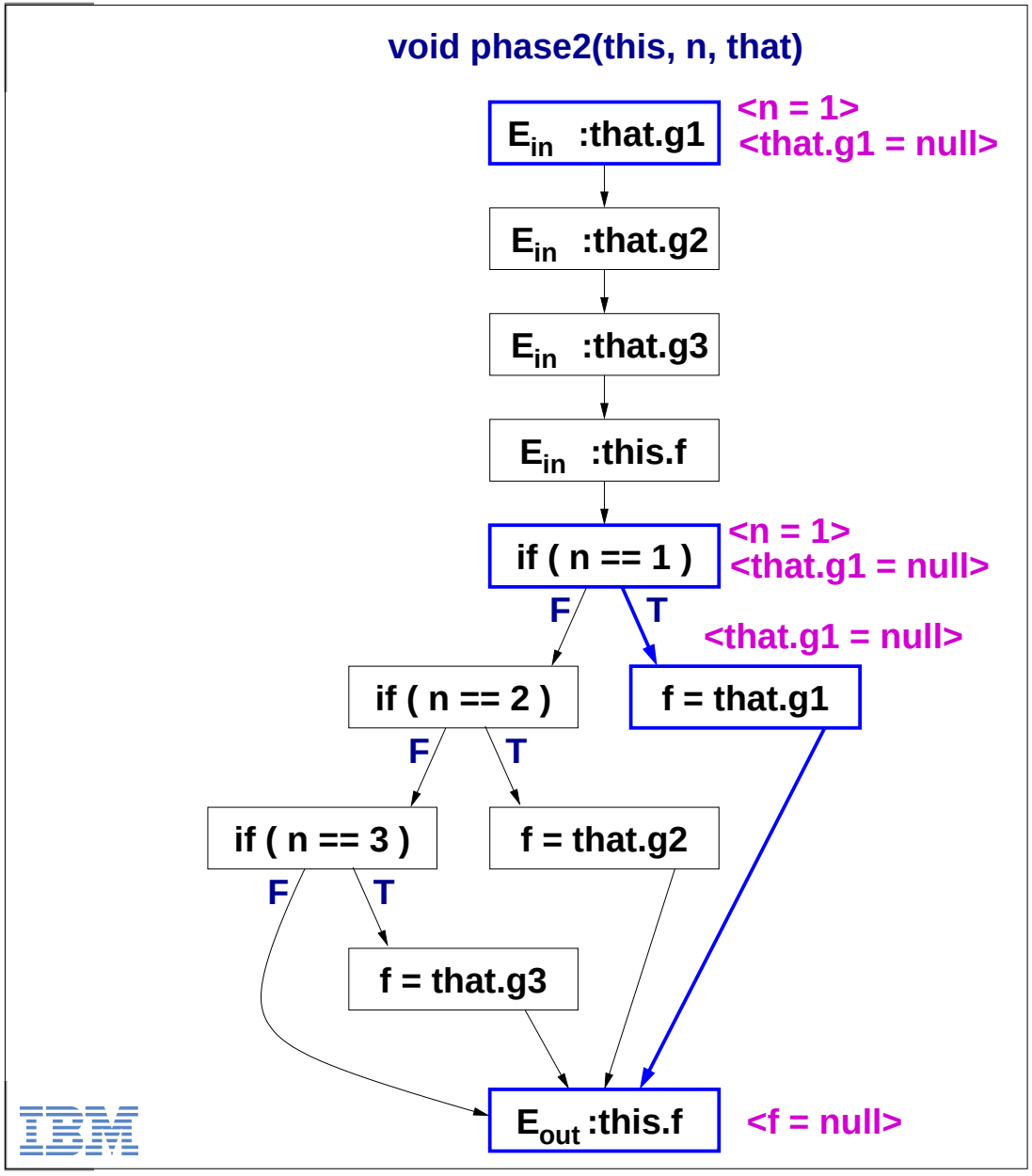
- Control Flow Graph with Formalin and Formalout
- Escape and Pointer analysis
- Control dependence analysis

Predicates, Transformations, States, and Consistency



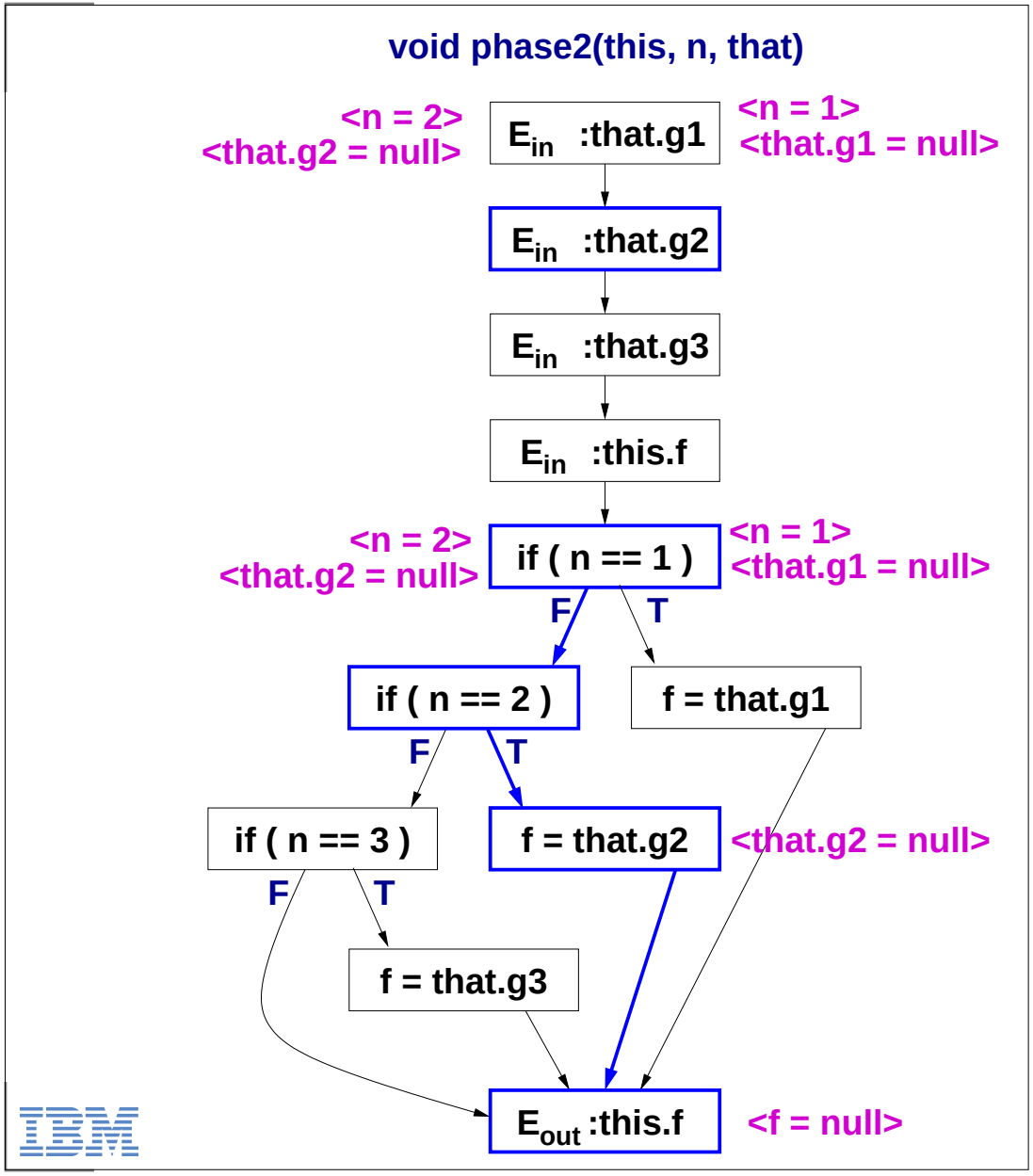
- **Root predicate**
- **Transformations**
- **States**
- **Consistency**

Predicates, Transformations, States, and Consistency



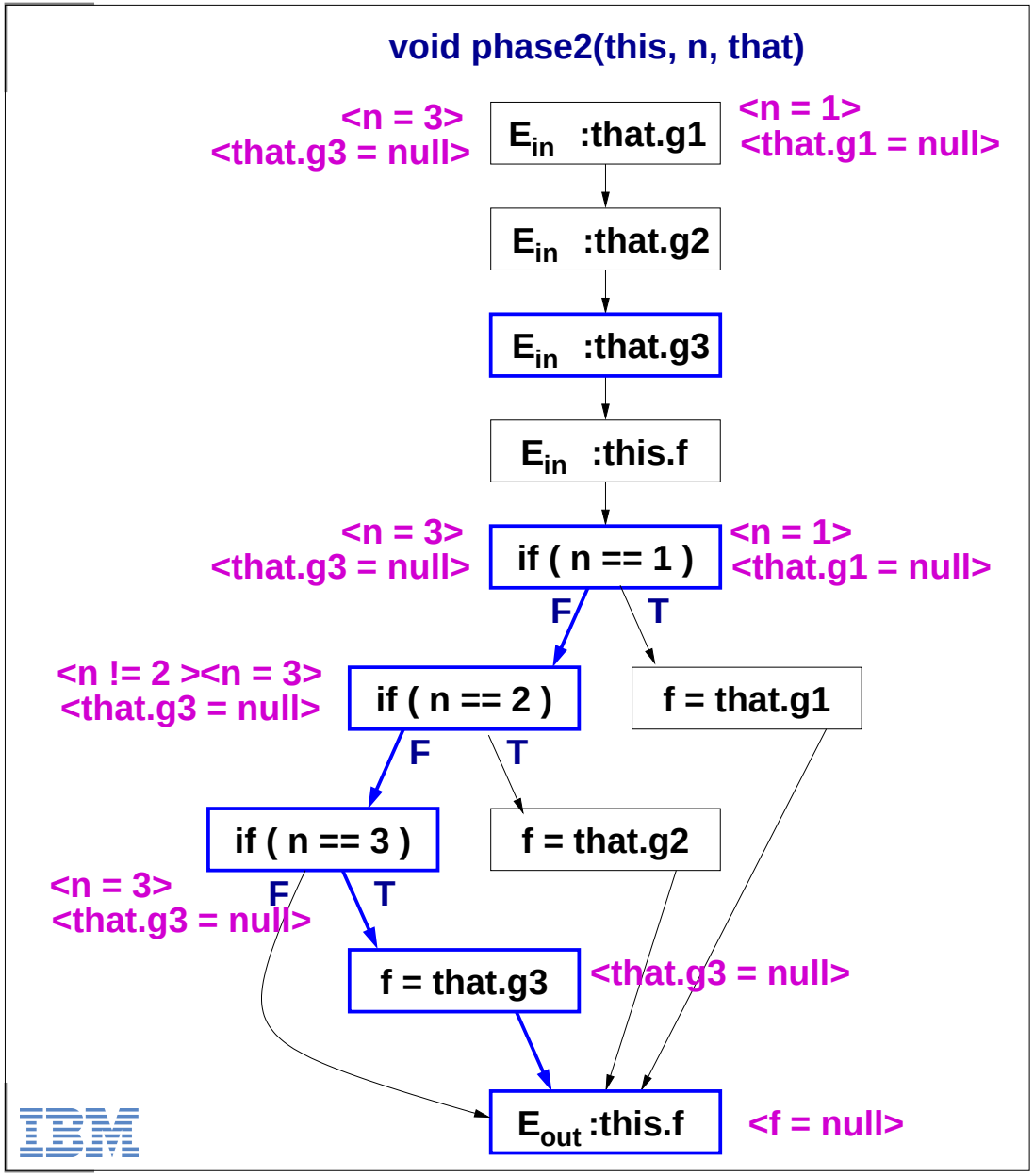
- Root predicate
- Transformations
- States
- Consistency

Predicates, Transformations, States, and Consistency



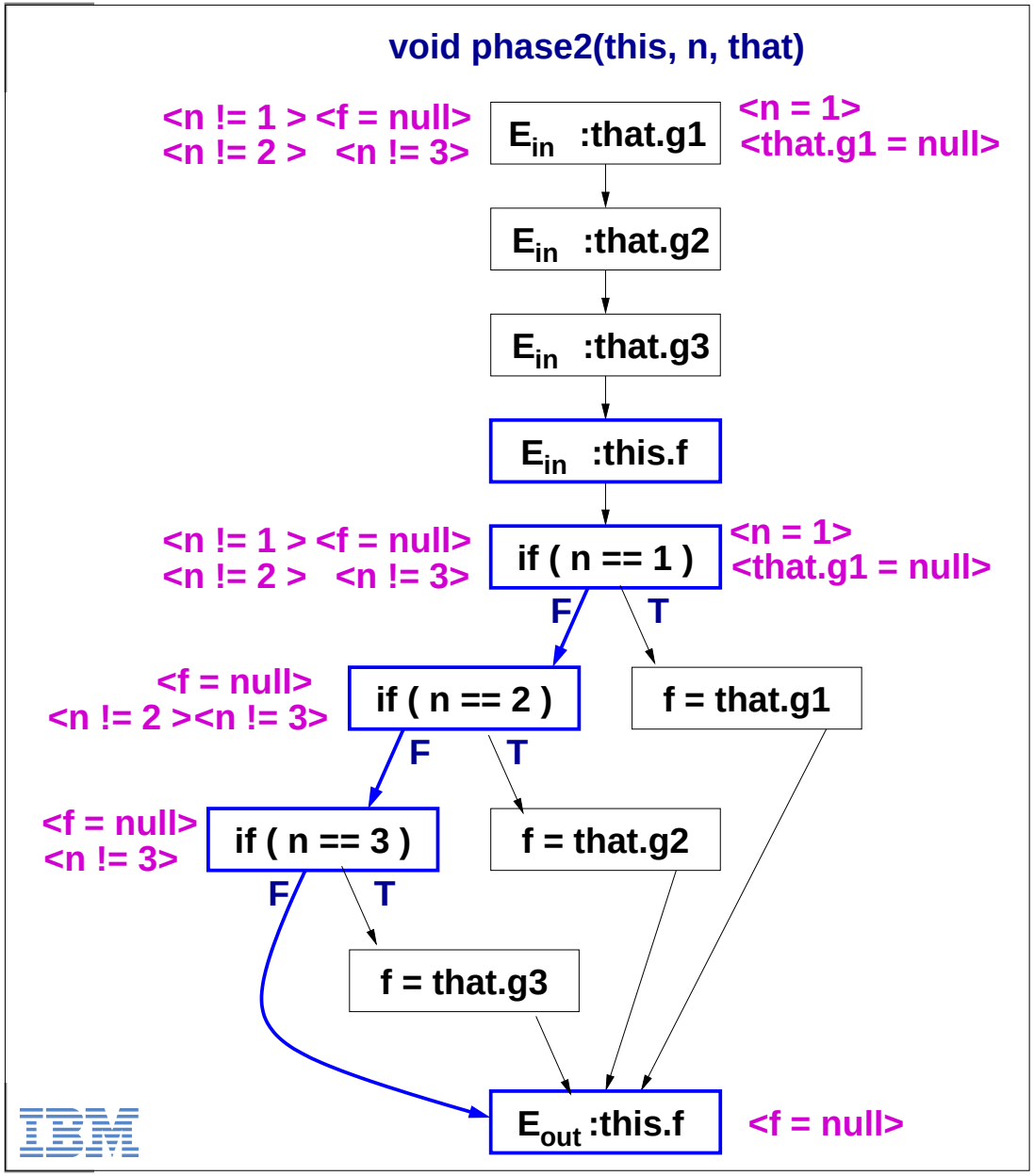
- Root predicate
- Transformations
- States
- Consistency

Predicates, Transformations, States, and Consistency



- Root predicate
- Transformations
- States
- Consistency

Method Summary



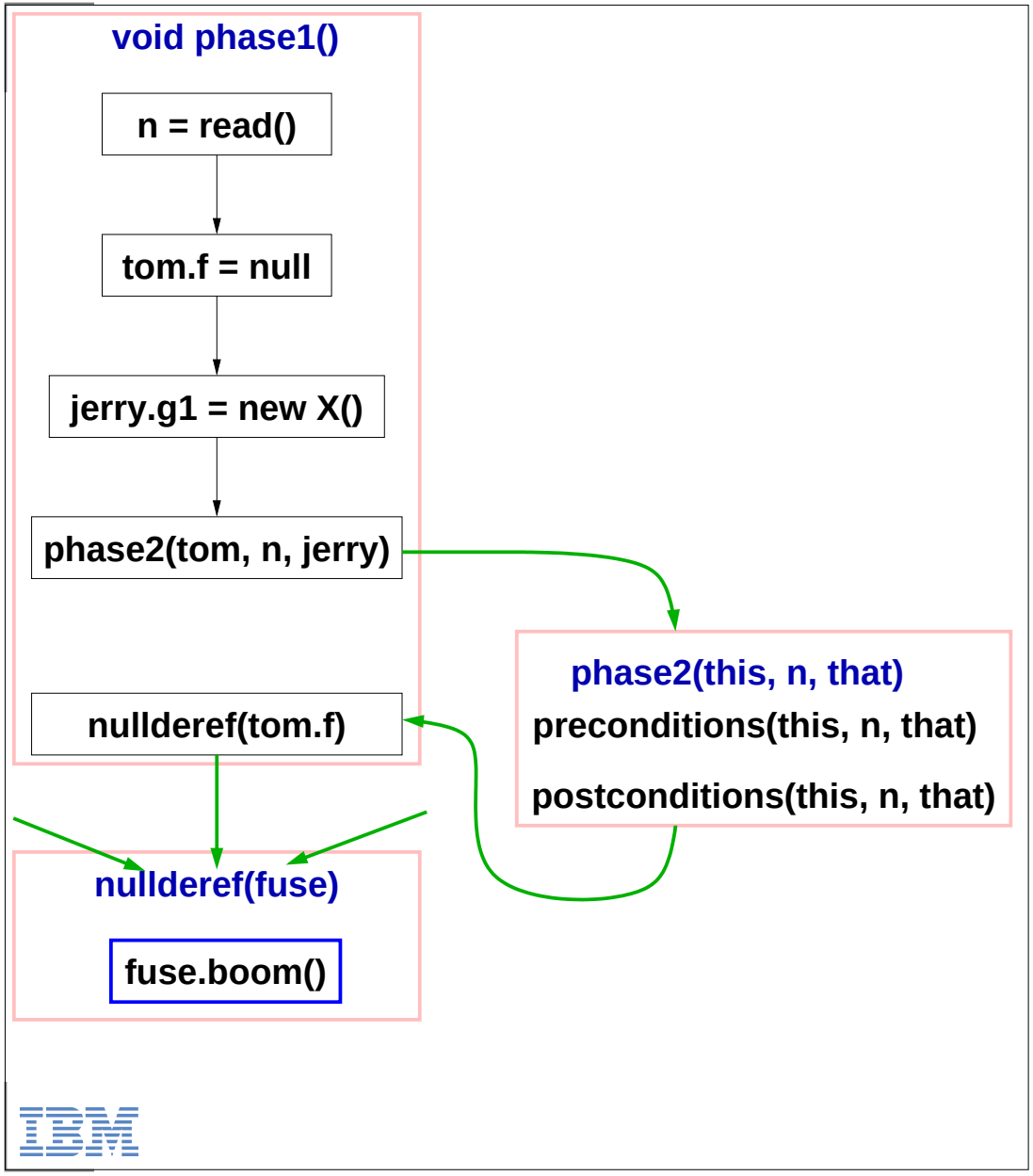
Postconditions

- ⟨ this.f = null ⟩

Preconditions

- ⟨ n = 1 ⟩ ⟨ that.g1 = null ⟩
- ⟨ n = 2 ⟩ ⟨ that.g2 = null ⟩
- ⟨ n = 3 ⟩ ⟨ that.g3 = null ⟩
- ⟨ n ≠ 3 ⟩ ⟨ n ≠ 2 ⟩ ⟨ n ≠ 1 ⟩ ⟨ this.f = null ⟩

Interprocedural Analysis and Method Summary



Method boundaries

- Phase1 - ascend into a calling method
- Phase2 - descend into a called method. Maintan stack for context-sensitivity
- Recursion

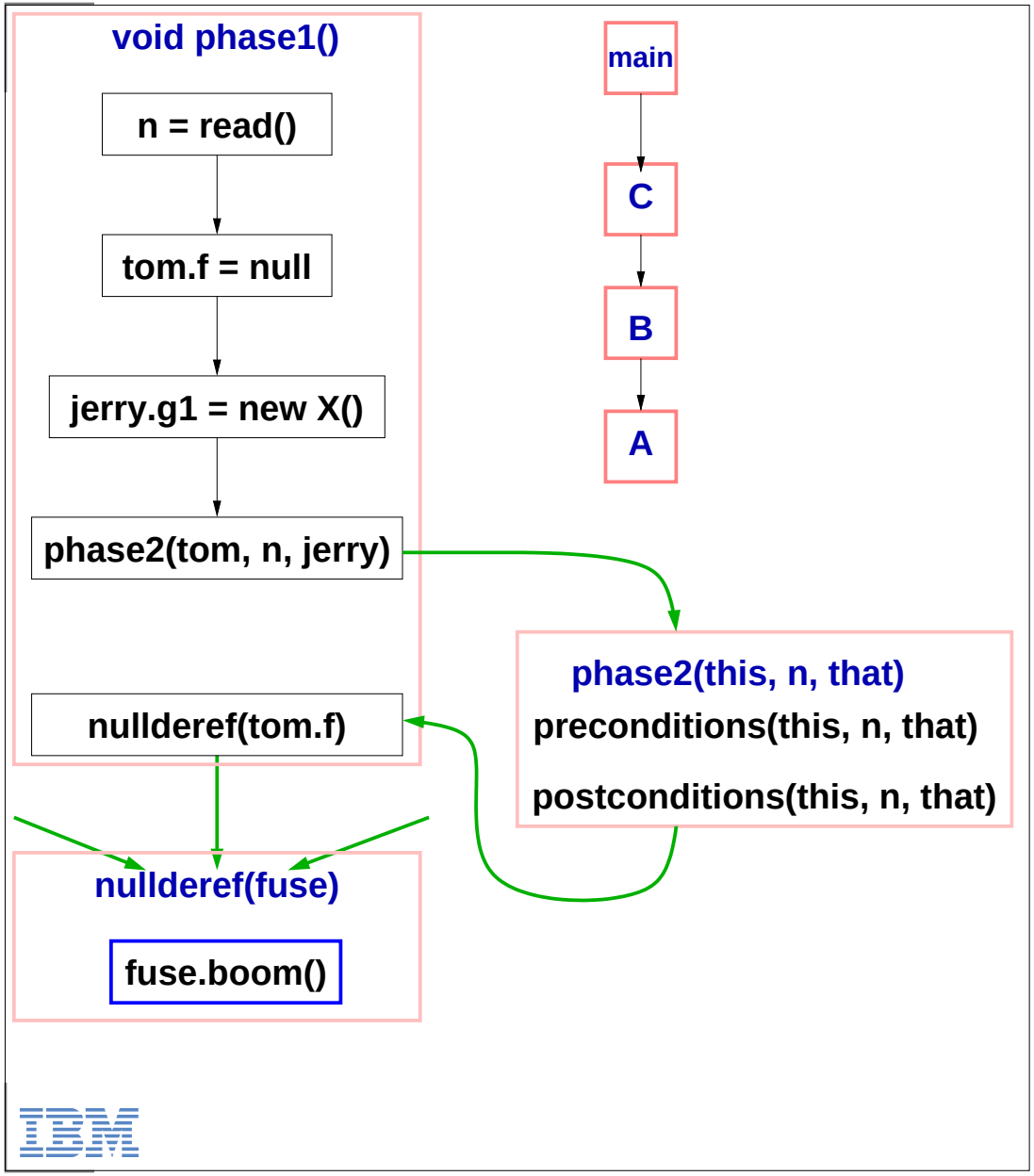
Postconditions

- $\langle$ this.f = null $\rangle$

Preconditions

- $\langle$ n = 1 $\rangle$ $\langle$ that.g1 = null $\rangle$
- $\langle$ n = 2 $\rangle$ $\langle$ that.g2 = null $\rangle$
- $\langle$ n = 3 $\rangle$ $\langle$ that.g3 = null $\rangle$
- $\langle$ n $\neq$ 3 $\rangle$ $\langle$ n $\neq$ 2 $\rangle$ $\langle$ n $\neq$ 1 $\rangle$ $\langle$ this.f = null $\rangle$

Interprocedural Analysis and Method Summary



Method boundaries

- **Phase1 - ascend into a calling method**
- Phase2 - descend into a called method. Maintan stack for context-sensitivity
- Recursion

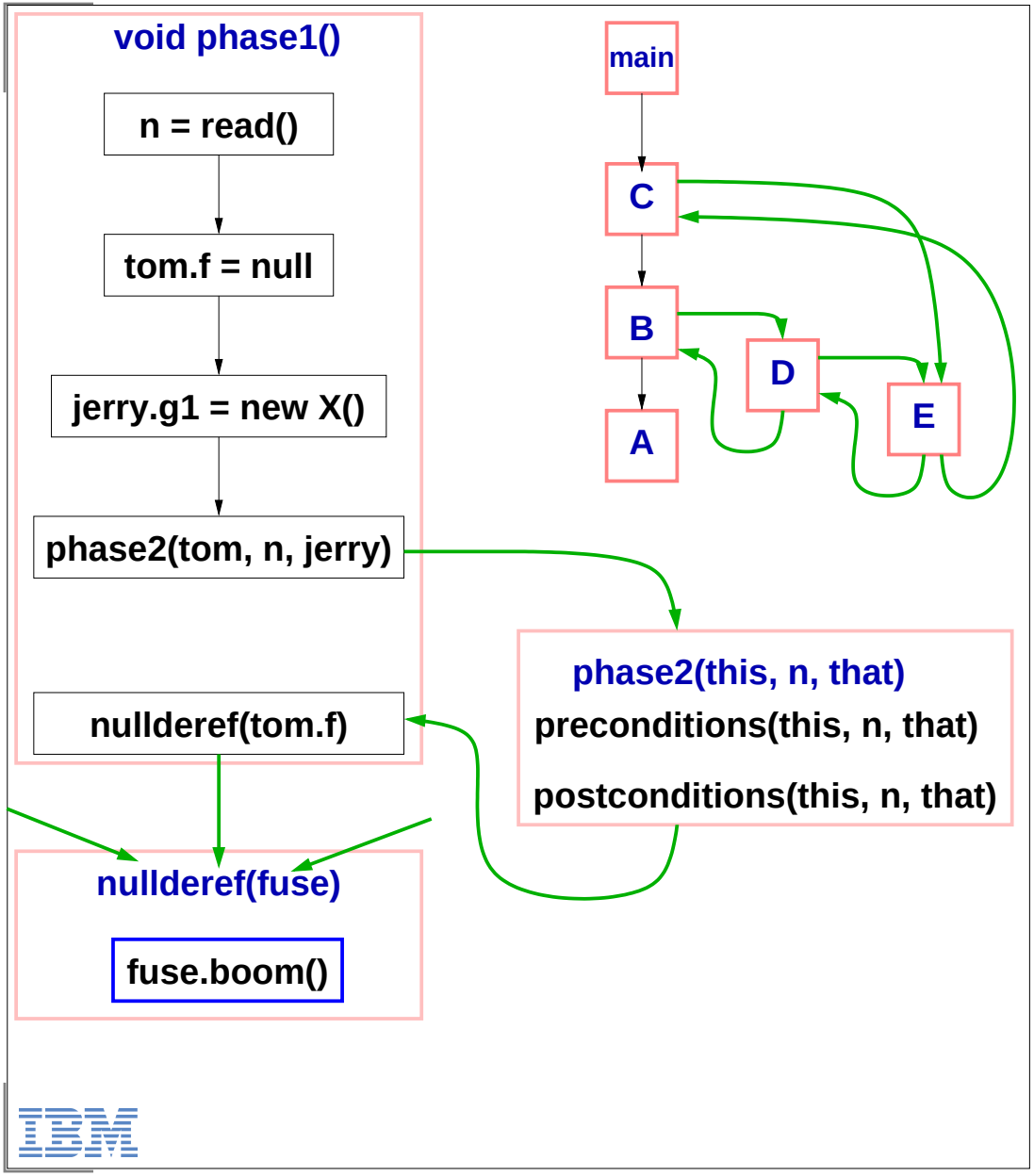
Postconditions

- $\langle$ this.f = null $\rangle$

Preconditions

- $\langle$ n = 1 $\rangle$ $\langle$ that.g1 = null $\rangle$
- $\langle$ n = 2 $\rangle$ $\langle$ that.g2 = null $\rangle$
- $\langle$ n = 3 $\rangle$ $\langle$ that.g3 = null $\rangle$
- $\langle$ n $\neq$ 3 $\rangle$ $\langle$ n $\neq$ 2 $\rangle$ $\langle$ n $\neq$ 1 $\rangle$ $\langle$ this.f = null $\rangle$

Interprocedural Analysis and Method Summary



Method boundaries

- Phase1 - ascend into a calling method
- Phase2 - descend into a called method. Maintan stack for context-sensitivity
- Recursion

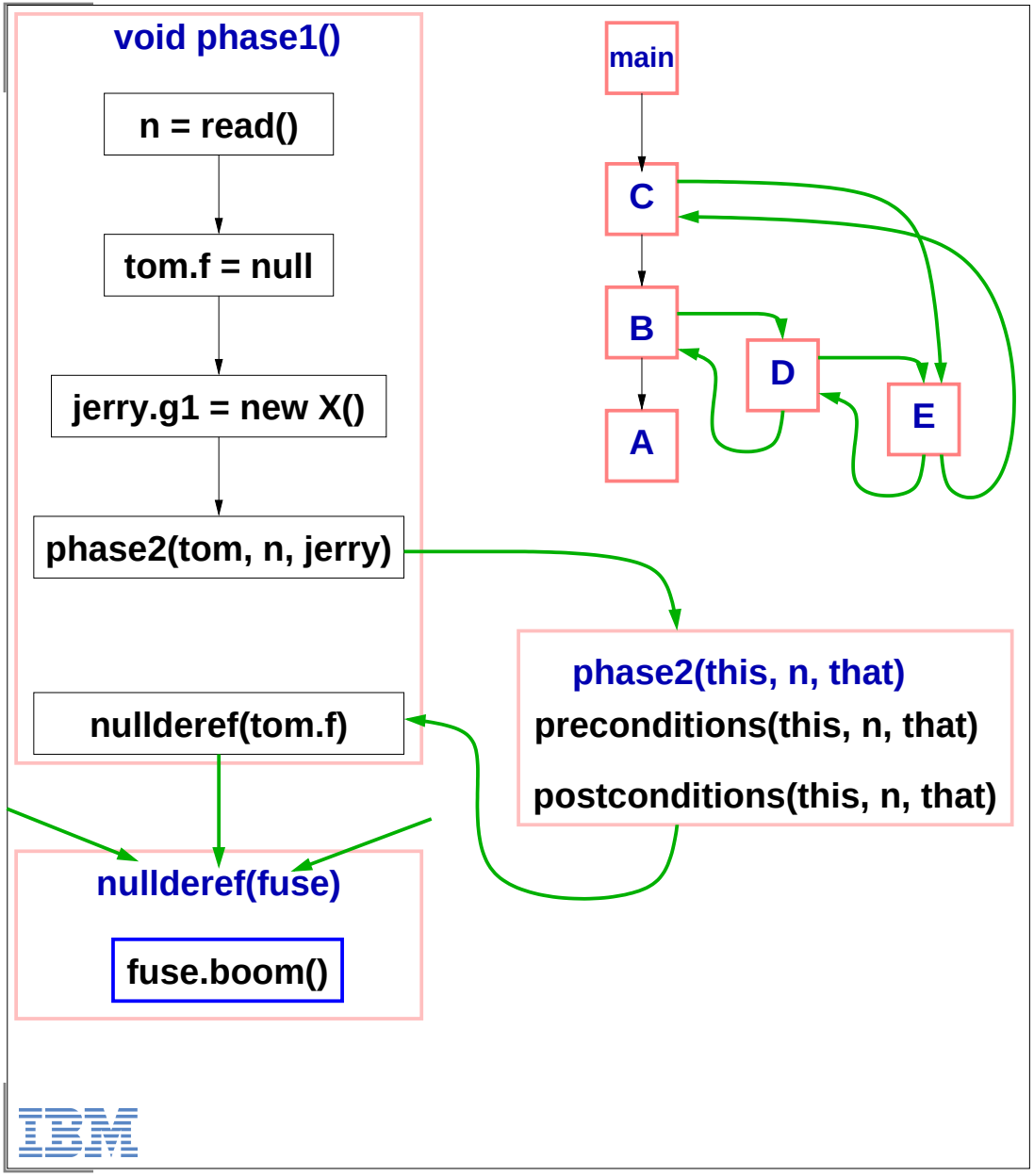
Postconditions

- $\langle$ this.f = null $\rangle$

Preconditions

- $\langle$ n = 1 $\rangle$ $\langle$ that.g1 = null $\rangle$
- $\langle$ n = 2 $\rangle$ $\langle$ that.g2 = null $\rangle$
- $\langle$ n = 3 $\rangle$ $\langle$ that.g3 = null $\rangle$
- $\langle$ n $\neq$ 3 $\rangle$ $\langle$ n $\neq$ 2 $\rangle$ $\langle$ n $\neq$ 1 $\rangle$ $\langle$ this.f = null $\rangle$

Interprocedural Analysis and Method Summary



Method boundaries

- Phase1 - ascend into a calling method
- Phase2 - descend into a called method. Maintan stack for context-sensitivity
- Recursion

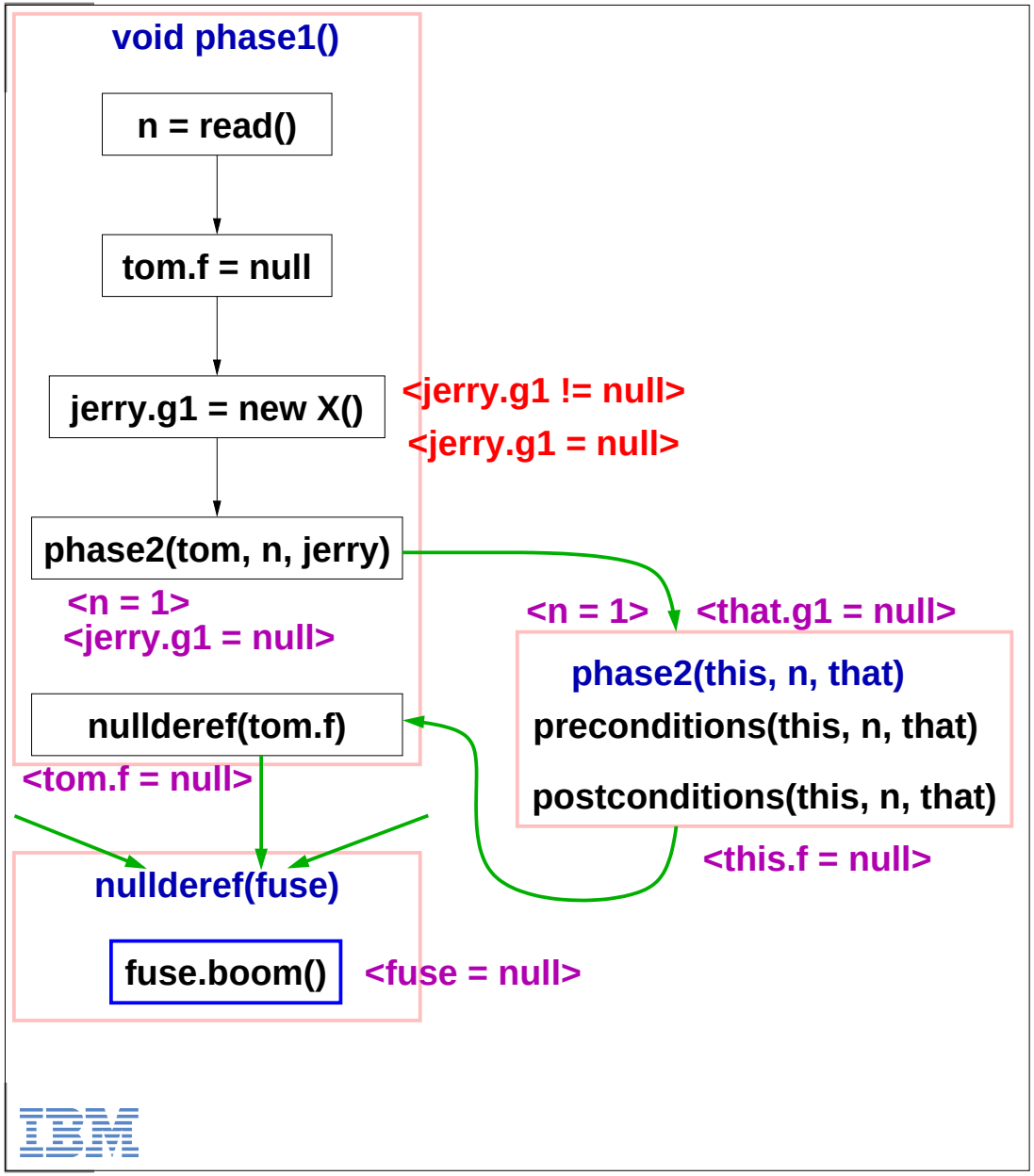
Postconditions

- $\langle$ this.f = null $\rangle$

Preconditions

- $\langle$ n = 1 $\rangle$ $\langle$ that.g1 = null $\rangle$
- $\langle$ n = 2 $\rangle$ $\langle$ that.g2 = null $\rangle$
- $\langle$ n = 3 $\rangle$ $\langle$ that.g3 = null $\rangle$
- $\langle$ n $\neq$ 3 $\rangle$ $\langle$ n $\neq$ 2 $\rangle$ $\langle$ n $\neq$ 1 $\rangle$ $\langle$ this.f = null $\rangle$

Interprocedural Analysis and Method Summary



Method boundaries

- Phase1 - ascend into a calling method
- Phase2 - descend into a called method. Maintan stack for context-sensitivity
- Recursion

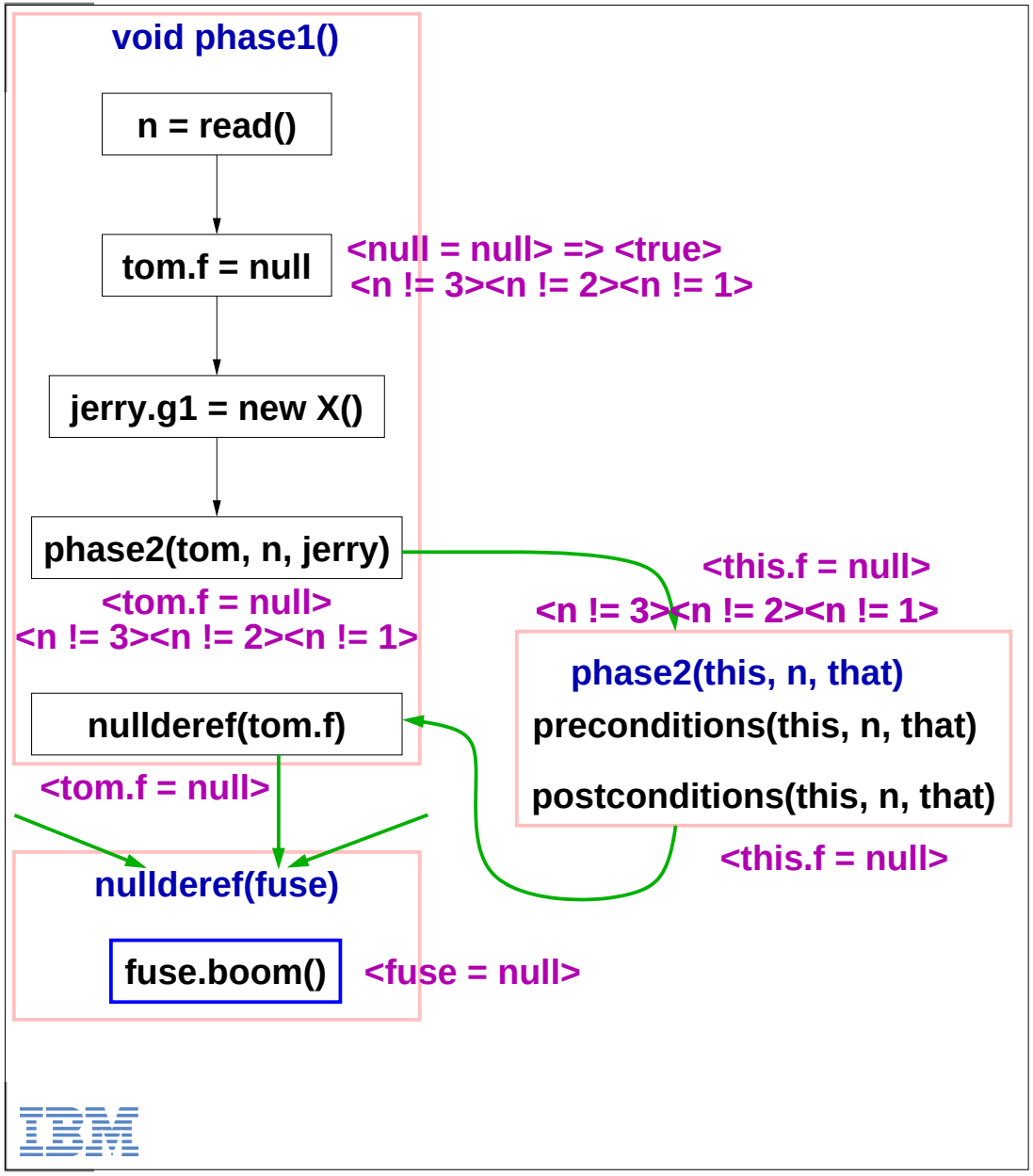
Postconditions

- $\langle$ this.f = null $\rangle$

Preconditions

- $\langle$ n = 1 $\rangle$ $\langle$ that.g1 = null $\rangle$
- $\langle$ n = 2 $\rangle$ $\langle$ that.g2 = null $\rangle$
- $\langle$ n = 3 $\rangle$ $\langle$ that.g3 = null $\rangle$
- $\langle$ n $\neq$ 3 $\rangle$ $\langle$ n $\neq$ 2 $\rangle$ $\langle$ n $\neq$ 1 $\rangle$ $\langle$ this.f = null $\rangle$

Interprocedural Analysis and Method Summary



Method boundaries

- Phase1 - ascend into a calling method
- Phase2 - descend into a called method. Maintan stack for context-sensitivity
- Recursion

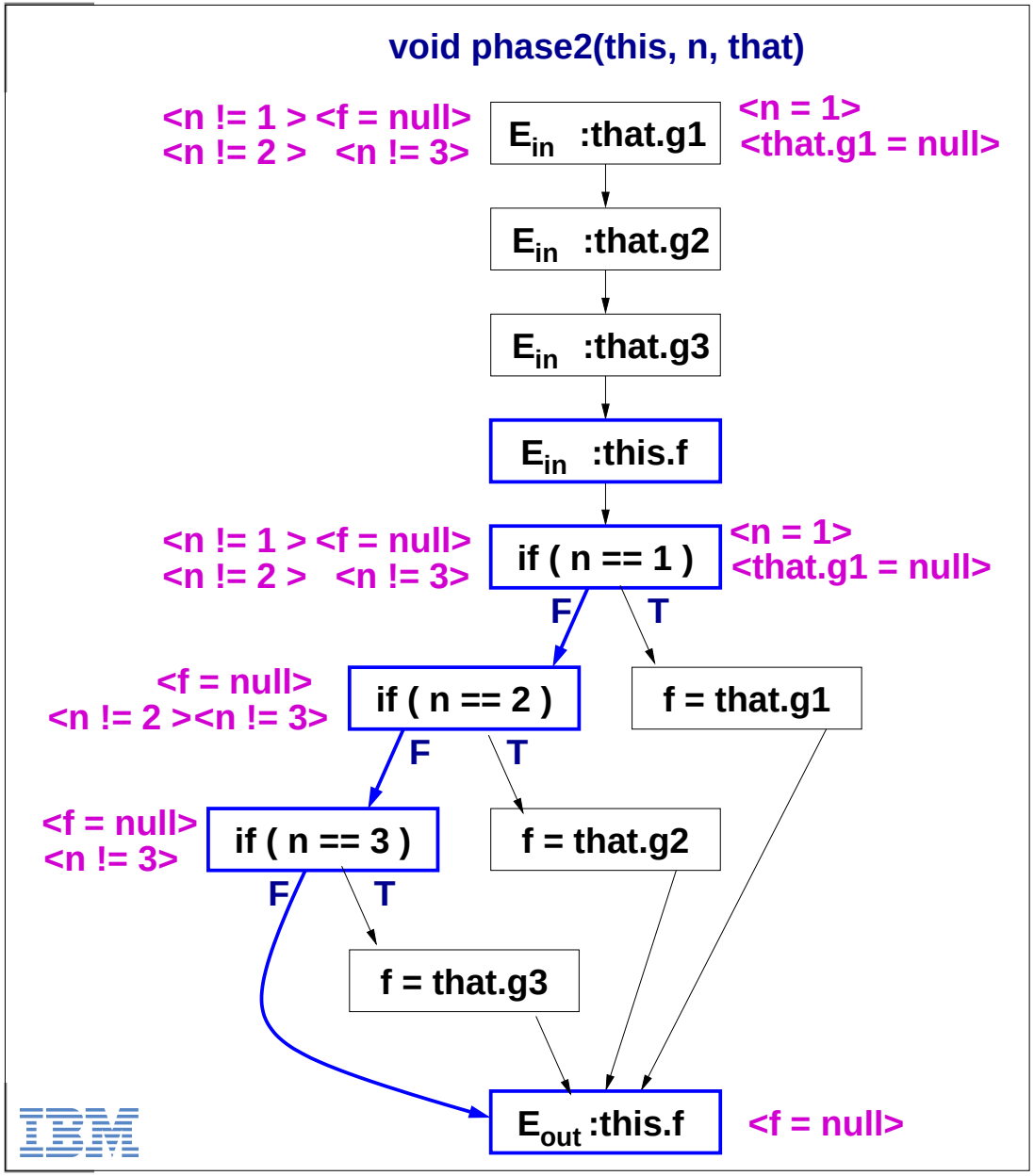
Postconditions

- $\langle$ this.f = null $\rangle$

Preconditions

- $\langle$ n = 1 $\rangle$ $\langle$ that.g1 = null $\rangle$
- $\langle$ n = 2 $\rangle$ $\langle$ that.g2 = null $\rangle$
- $\langle$ n = 3 $\rangle$ $\langle$ that.g3 = null $\rangle$
- $\langle$ n $\neq$ 3 $\rangle$ $\langle$ n $\neq$ 2 $\rangle$ $\langle$ n $\neq$ 1 $\rangle$ $\langle$ this.f = null $\rangle$

Scaling up the Analysis



- Parameterized exploration
  - Paths in a method: the number of postcondition to precondtion mappings per method (7)
  - States per node (80)
  - Time per traversal (2sec)
- Simplified predicate handling
- Postconditions
  - ⟨ this.f = null ⟩
- Preconditions
  - ⟨ n = 1 ⟩ ⟨ that.g1 = null ⟩
  - ⟨ n = 2 ⟩ ⟨ that.g2 = null ⟩
  - ⟨ n = 3 ⟩ ⟨ that.g3 = null ⟩
  - ⟨ n ≠ 3 ⟩ ⟨ n ≠ 2 ⟩ ⟨ n ≠ 1 ⟩ ⟨ this.f = null ⟩

Integer Arithmetic

P_1	P_2	test	T	F
$x = v_1$	$x = v_2$	$v_1 = v_2$	P_1	X
$x = v_1$	$x \neq v_2$	$v_1 \neq v_2$	P_1	X
$x = v_1$	$x < v_2$	$v_1 < v_2$	P_1	X
$x = v_1$	$x \leq v_2$	$v_1 \leq v_2$	P_1	X
$x = v_1$	$x > v_2$	$v_1 > v_2$	P_1	X
$x = v_1$	$x \geq v_2$	$v_1 \geq v_2$	P_1	X
$x \neq v_1$	$x = v_2$	$v_1 \neq v_2$	P_2	X
$x \neq v_1$	$x \neq v_2$	-	$P_{1,2}$	$P_{1,2}$
$x \neq v_1$	$x < v_2$	$v_1 \geq v_2$	P_2	P_1
$x \neq v_1$	$x \leq v_2$	$v_1 > v_2$	P_2	P_1
$x \neq v_1$	$x > v_2$	$v_1 \leq v_2$	P_2	P_1
$x \neq v_1$	$x \geq v_2$	$v_1 < v_2$	P_2	P_1

Integer Arithmetic

P_1	P_2	test	T	F
$x = v_1$	$x = v_2$	$v_1 = v_2$	P_1	X
$x = v_1$	$x \neq v_2$	$v_1 \neq v_2$	P_1	X
$x = v_1$	$x < v_2$	$v_1 < v_2$	P_1	X
$x = v_1$	$x \leq v_2$	$v_1 \leq v_2$	P_1	X
$x = v_1$	$x > v_2$	$v_1 > v_2$	P_1	X
$x = v_1$	$x \geq v_2$	$v_1 \geq v_2$	P_1	X
$x \neq v_1$	$x = v_2$	$v_1 \neq v_2$	P_2	X
$x \neq v_1$	$x \neq v_2$	-	$P_{1,2}$	$P_{1,2}$
$x \neq v_1$	$x < v_2$	$v_1 \geq v_2$	P_2	P_1
$x \neq v_1$	$x \leq v_2$	$v_1 > v_2$	P_2	P_1
$x \neq v_1$	$x > v_2$	$v_1 \leq v_2$	P_2	P_1
$x \neq v_1$	$x \geq v_2$	$v_1 < v_2$	P_2	P_1

$x = 10$ **and** $x < 15 \implies x = 10$

Integer Arithmetic

P_1	P_2	test	T	F
$x = v_1$	$x = v_2$	$v_1 = v_2$	P_1	X
$x = v_1$	$x \neq v_2$	$v_1 \neq v_2$	P_1	X
$x = v_1$	$x < v_2$	$v_1 < v_2$	P_1	X
$x = v_1$	$x \leq v_2$	$v_1 \leq v_2$	P_1	X
$x = v_1$	$x > v_2$	$v_1 > v_2$	P_1	X
$x = v_1$	$x \geq v_2$	$v_1 \geq v_2$	P_1	X
$x \neq v_1$	$x = v_2$	$v_1 \neq v_2$	P_2	X
$x \neq v_1$	$x \neq v_2$	-	$P_{1,2}$	$P_{1,2}$
$x \neq v_1$	$x < v_2$	$v_1 \geq v_2$	P_2	P_1
$x \neq v_1$	$x \leq v_2$	$v_1 > v_2$	P_2	P_1
$x \neq v_1$	$x > v_2$	$v_1 \leq v_2$	P_2	P_1
$x \neq v_1$	$x \geq v_2$	$v_1 < v_2$	P_2	P_1

$x = 10$ **and** $x < 15 \implies x = 10$

$x = 10$ **and** $x < 5 \implies X$

Integer Arithmetic

P_1	P_2	test	T	F
$x = v_1$	$x = v_2$	$v_1 = v_2$	P_1	X
$x = v_1$	$x \neq v_2$	$v_1 \neq v_2$	P_1	X
$x = v_1$	$x < v_2$	$v_1 < v_2$	P_1	X
$x = v_1$	$x \leq v_2$	$v_1 \leq v_2$	P_1	X
$x = v_1$	$x > v_2$	$v_1 > v_2$	P_1	X
$x = v_1$	$x \geq v_2$	$v_1 \geq v_2$	P_1	X
$x \neq v_1$	$x = v_2$	$v_1 \neq v_2$	P_2	X
$x \neq v_1$	$x \neq v_2$	-	$P_{1,2}$	$P_{1,2}$
$x \neq v_1$	$x < v_2$	$v_1 \geq v_2$	P_2	P_1
$x \neq v_1$	$x \leq v_2$	$v_1 > v_2$	P_2	P_1
$x \neq v_1$	$x > v_2$	$v_1 \leq v_2$	P_2	P_1
$x \neq v_1$	$x \geq v_2$	$v_1 < v_2$	P_2	P_1

$x \neq 10$ **and** $x < 8 \implies x < 8$

Integer Arithmetic

P_1	P_2	test	T	F
$x = v_1$	$x = v_2$	$v_1 = v_2$	P_1	X
$x = v_1$	$x \neq v_2$	$v_1 \neq v_2$	P_1	X
$x = v_1$	$x < v_2$	$v_1 < v_2$	P_1	X
$x = v_1$	$x \leq v_2$	$v_1 \leq v_2$	P_1	X
$x = v_1$	$x > v_2$	$v_1 > v_2$	P_1	X
$x = v_1$	$x \geq v_2$	$v_1 \geq v_2$	P_1	X
$x \neq v_1$	$x = v_2$	$v_1 \neq v_2$	P_2	X
$x \neq v_1$	$x \neq v_2$	-	$P_{1,2}$	$P_{1,2}$
$x \neq v_1$	$x < v_2$	$v_1 \geq v_2$	P_2	P_1
$x \neq v_1$	$x \leq v_2$	$v_1 > v_2$	P_2	P_1
$x \neq v_1$	$x > v_2$	$v_1 \leq v_2$	P_2	P_1
$x \neq v_1$	$x \geq v_2$	$v_1 < v_2$	P_2	P_1

$x \neq 10$ **and** $x < 8 \implies x < 8$

$x \neq 10$ **and** $x < 15 \implies$

$x \neq 10$ **and** $x < 15$

Integer Arithmetic

P_1	P_2	test	T	F
$x = v_1$	$x = v_2$	$v_1 = v_2$	P_1	X
$x = v_1$	$x \neq v_2$	$v_1 \neq v_2$	P_1	X
$x = v_1$	$x < v_2$	$v_1 < v_2$	P_1	X
$x = v_1$	$x \leq v_2$	$v_1 \leq v_2$	P_1	X
$x = v_1$	$x > v_2$	$v_1 > v_2$	P_1	X
$x = v_1$	$x \geq v_2$	$v_1 \geq v_2$	P_1	X
$x \neq v_1$	$x = v_2$	$v_1 \neq v_2$	P_2	X
$x \neq v_1$	$x \neq v_2$	-	$P_{1,2}$	$P_{1,2}$
$x \neq v_1$	$x < v_2$	$v_1 \geq v_2$	P_2	P_1
$x \neq v_1$	$x \leq v_2$	$v_1 > v_2$	P_2	P_1
$x \neq v_1$	$x > v_2$	$v_1 \leq v_2$	P_2	P_1
$x \neq v_1$	$x \geq v_2$	$v_1 < v_2$	P_2	P_1

```
[1] if (x == null)
[2]     y = 0;          ⟨ x = null ⟩ ⟨ 0 < 0 ⟩
[3] i = 0;              ⟨x = null⟩⟨0 < y⟩
[4] while(i < y)        ⟨x = null⟩⟨i < y⟩
[5]     x.foo();        ⟨x = null⟩
```

Integer Arithmetic

P_1	P_2	test	T	F
$x = v_1$	$x = v_2$	$v_1 = v_2$	P_1	X
$x = v_1$	$x \neq v_2$	$v_1 \neq v_2$	P_1	X
$x = v_1$	$x < v_2$	$v_1 < v_2$	P_1	X
$x = v_1$	$x \leq v_2$	$v_1 \leq v_2$	P_1	X
$x = v_1$	$x > v_2$	$v_1 > v_2$	P_1	X
$x = v_1$	$x \geq v_2$	$v_1 \geq v_2$	P_1	X
$x \neq v_1$	$x = v_2$	$v_1 \neq v_2$	P_2	X
$x \neq v_1$	$x \neq v_2$	-	$P_{1,2}$	$P_{1,2}$
$x \neq v_1$	$x < v_2$	$v_1 \geq v_2$	P_2	P_1
$x \neq v_1$	$x \leq v_2$	$v_1 > v_2$	P_2	P_1
$x \neq v_1$	$x > v_2$	$v_1 \leq v_2$	P_2	P_1
$x \neq v_1$	$x \geq v_2$	$v_1 < v_2$	P_2	P_1

```
[1]  if (x == null)
[2]      y = 0;          ⟨ x = null ⟩ ⟨ 0 < 0 ⟩
[3]  i = 0;              ⟨x = null⟩⟨0 < y⟩
[4]  while(i < y)        ⟨x = null⟩⟨i < y⟩
[5]      x.foo();        ⟨x = null⟩

[6]  if (x == null)      False Positive
[7]      y = 1;
[8]  else
[9]      y = z+1;
[10] i = 0;              ⟨x = null⟩
[11] t = y-1;            ⟨x = null⟩⟨true⟩
[12] while(i < t)        ⟨x = null⟩⟨i < t⟩
[13]     x.foo();        ⟨x = null⟩
```

Dual Variable Predicates

P_{double}	P_i	**gen**
$x_1 = x_2$	x_1 **op** $const$	x_2 **op** $const$
	x_2 **op** $const$	x_1 **op** $const$
$x_1 \neq x_2$	$x_1 = const$	$x_2 \neq const$
	$x_2 = const$	$x_1 \neq const$

Dual Variable Predicates

P_{double}	P_i	**gen**
$x_1 = x_2$	x_1 **op** $const$	x_2 **op** $const$
	x_2 **op** $const$	x_1 **op** $const$
$x_1 \neq x_2$	$x_1 = const$	$x_2 \neq const$
	$x_2 = const$	$x_1 \neq const$

[1] **b = x.foo();** $\langle$ **x** $\neq$ **null** $\rangle$ $\langle y = null \rangle$ $\langle$ **x** $=$ **null** $\rangle$

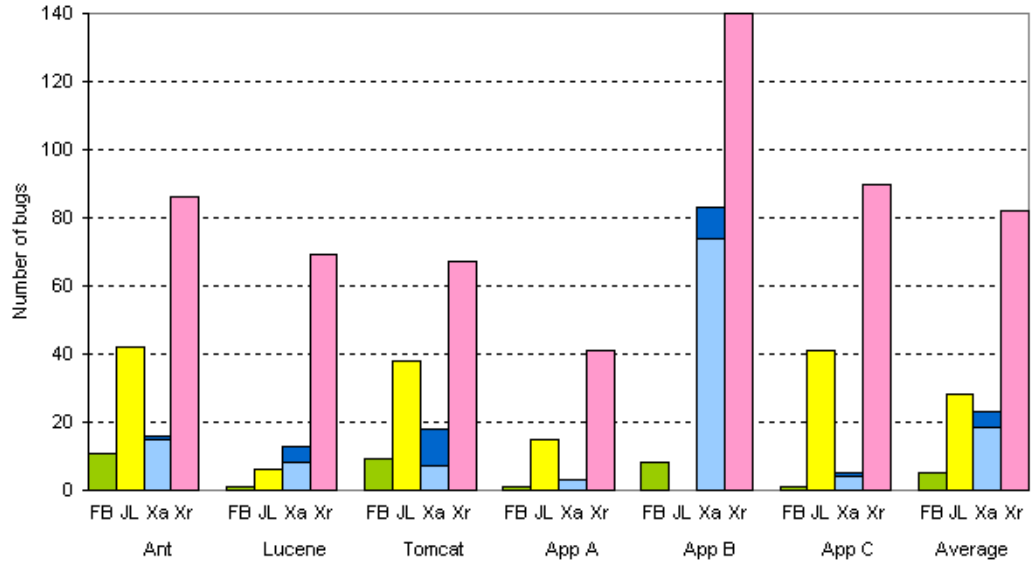
[2] **if (x == y)** $\langle y = null \rangle \langle x = null \rangle$

[3] **y.bar();** $\langle y = null \rangle$

Instanceof

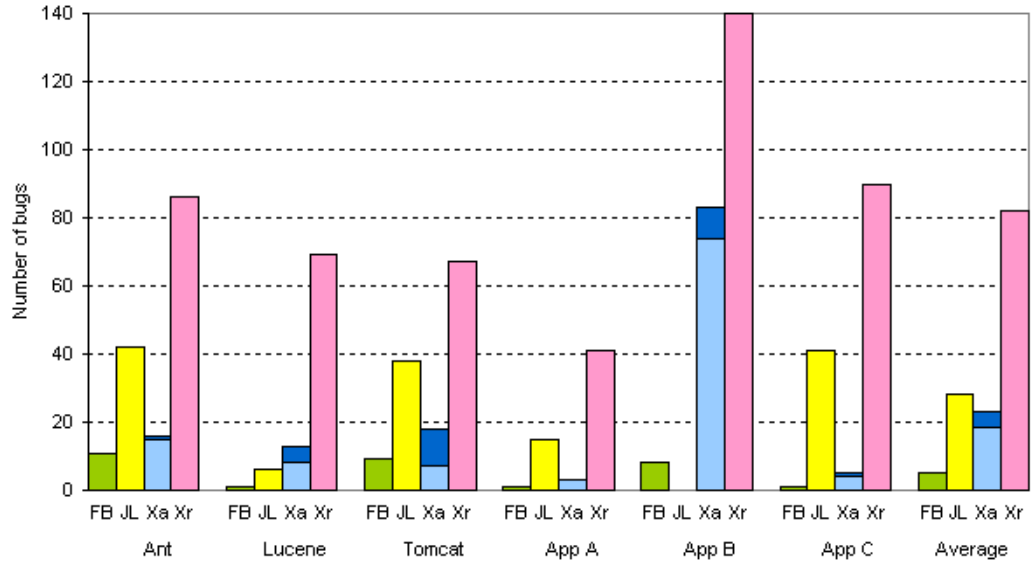
$[1]$	**if (el instanceof T)**	$\langle$ **el** $\neq$ **null** $\rangle$ $\langle$ **el** $=$ **null** $\rangle$
$[2]$	**T ta = (T) el;**	$\langle el = null \rangle$
$[3]$	**ta.get();**	$\langle ta = null \rangle$

Empirical Evaluation - Accuracy



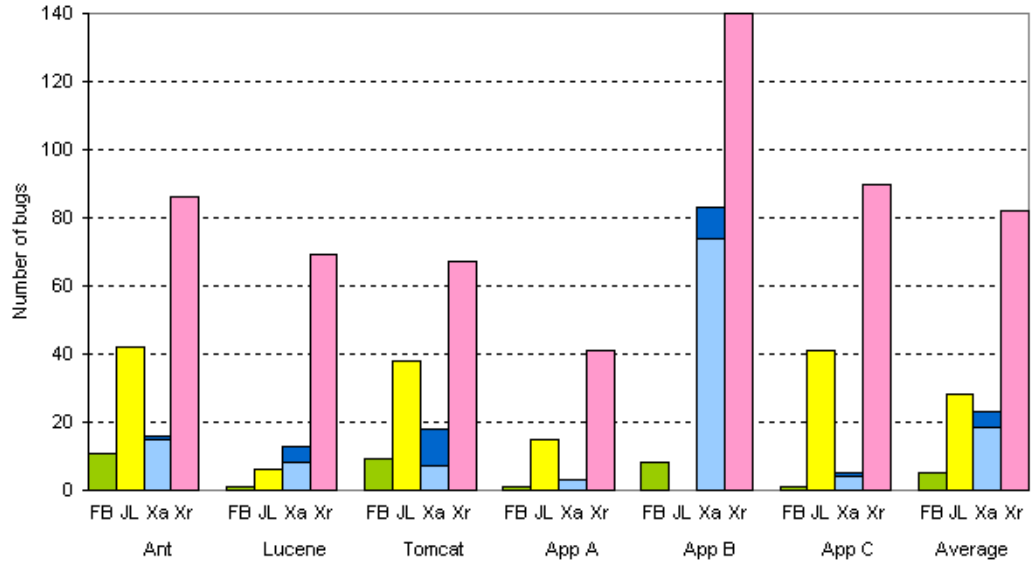
XYLEM**-inter detects 16 times as many bugs as** FINDBUGS **and 3 times as many bugs as** JLINT**.**

Empirical Evaluation - Accuracy



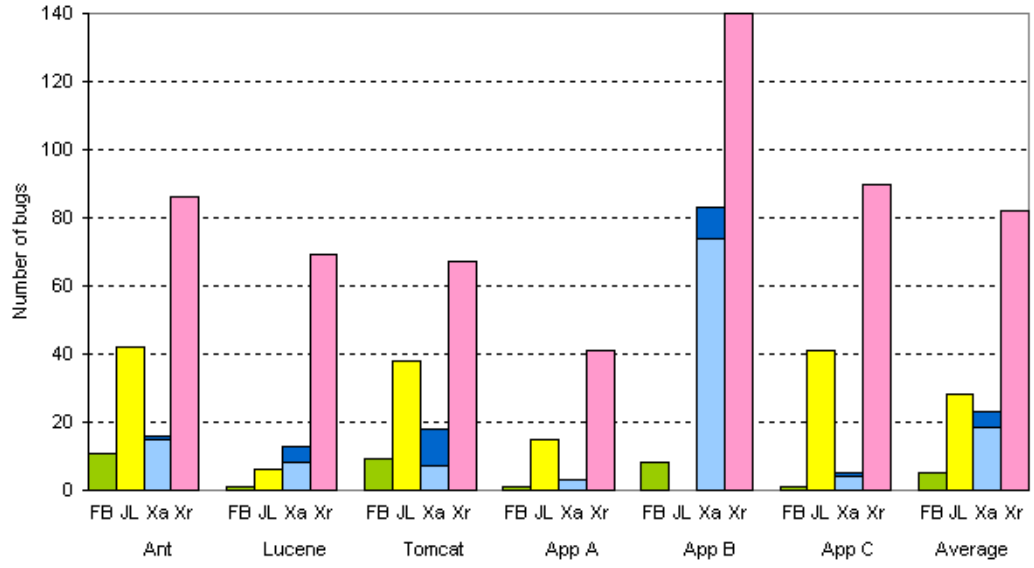
- XYLEM-inter detects 16 times as many bugs as FINDBUGS and 3 times as many bugs as JLINT.
- less than 5% of the intraprocedural bugs were invalidated by interprocedural analysis.

Empirical Evaluation - Accuracy



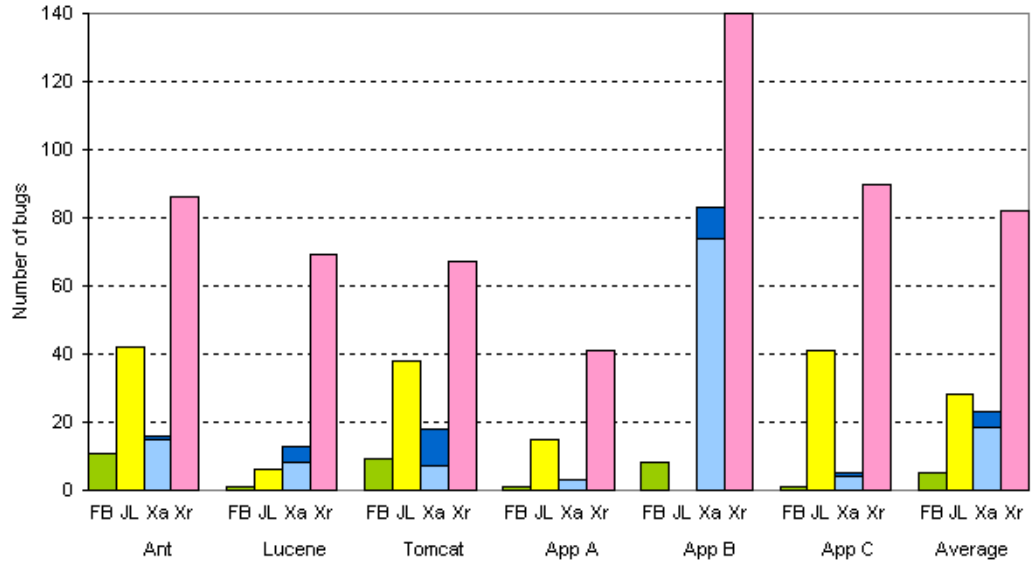
- XYLEM-inter detects 16 times as many bugs as FINDBUGS and 3 times as many bugs as JLINT.
- less than 5% of the intraprocedural bugs were invalidated by interprocedural analysis.
- For Ant JLINT reported 32 out of 42 (>75%) false positives. XYLEM-inter reported 4 out of 82 (<5%) false positives. FINDBUGS reported 1 out of 11 (10%) false positive.

Empirical Evaluation - Accuracy



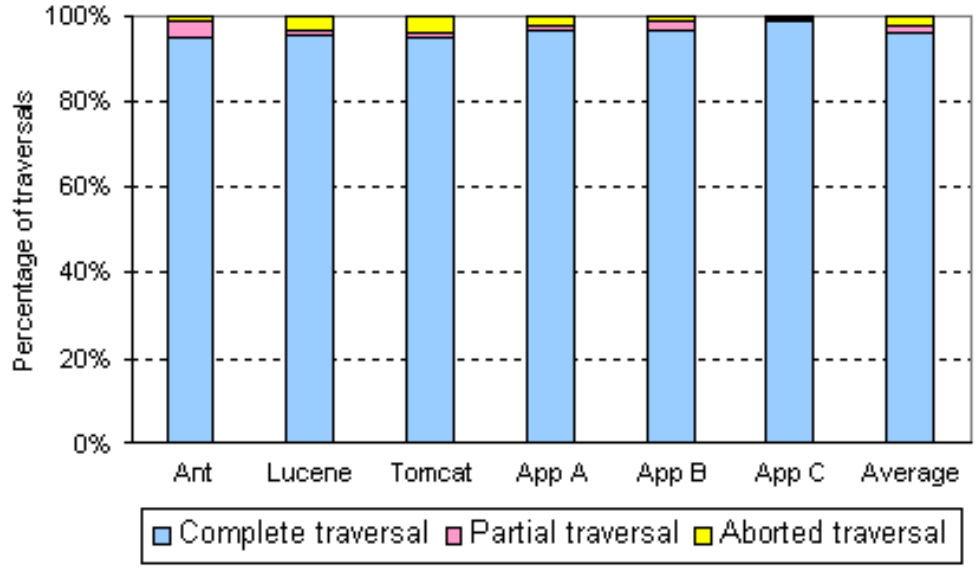
- XYLEM-inter detects 16 times as many bugs as FINDBUGS and 3 times as many bugs as JLINT.
- less than 5% of the intraprocedural bugs were invalidated by interprocedural analysis.
- For Ant JLINT reported 32 out of 42 (>75%) false positives. XYLEM-inter reported 4 out of 82 (<5%) false positives. FINDBUGS reported 1 out of 11 (10%) false positive.
- FINDBUGS reported one interprocedural bug. JLINT reported 7. XYLEM-inter reported 66.

Empirical Evaluation - Accuracy



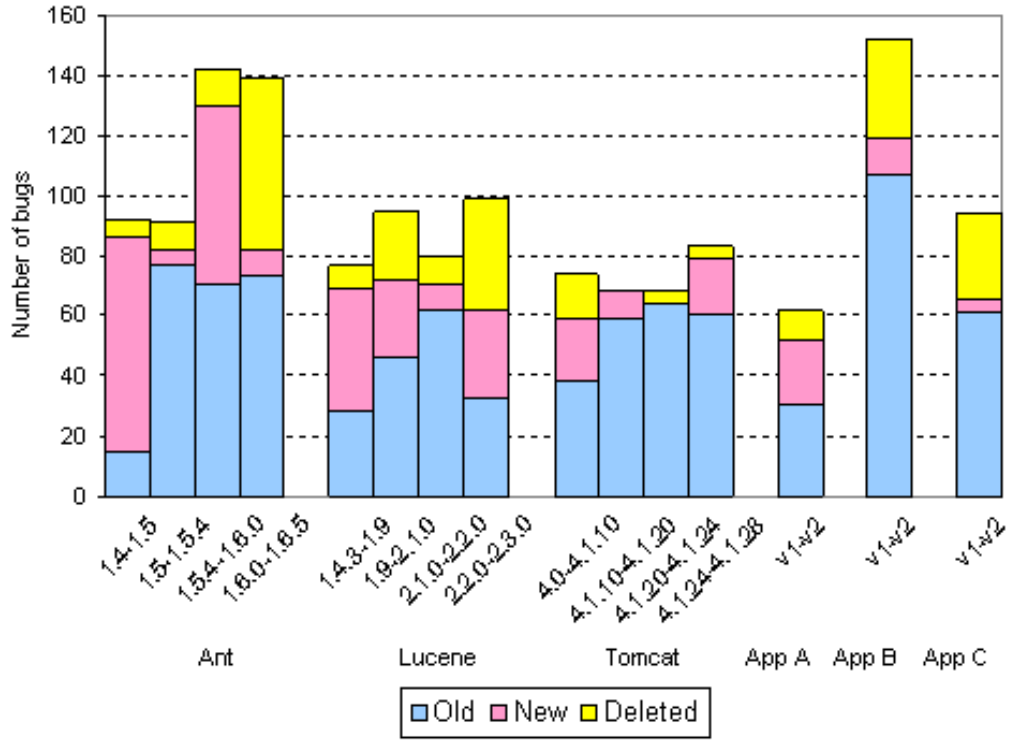
- XYLEM-inter detects 16 times as many bugs as FINDBUGS and 3 times as many bugs as JLINT.
- less than 5% of the intraprocedural bugs were invalidated by interprocedural analysis.
- For Ant JLINT reported 32 out of 42 (>75%) false positives. XYLEM-inter reported 4 out of 82 (<5%) false positives. FINDBUGS reported 1 out of 11 (10%) false positive.
- FINDBUGS reported one interprocedural bug. JLINT reported 7. XYLEM-inter reported 66.
- XYLEM-inter detected each true positive (intra or interprocedural) reported by either FINDBUGS or JLINT, and invalidated each false positive reported by FINDBUGS and JLINT.

Empirical Evaluation - Efficiency



- **The two-second time limit caused less than 1% of the traversals to abort.**
- **Partial traversals, caused by path and state limits, overall, were about 2%.**
- **For three subjects, more than 97% of the traversals were complete**
- **The largest application analyzed by XYLEM contains over 1,009,000 lines of code. The analysis completed in approximately 3 hours and 30 minutes.**

Empirical Evaluation - Relevance



- **The number of deleted bugs is an indicator of the relevance of the identified bugs.**
- FINDBUGS **reported bugs, 11 (10%) were deleted. For** XYLEM **257 (24%) bugs were deleted.**

Conclusions and Future Work

- We have presented an accurate analysis for detecting null-dereference bugs in Java programs.
- Our analysis detects bugs that many commonly used tools miss, and it eliminates the false positives that other tools report.
- Our studies illustrate the efficiency of the analysis and the relevance of the detected bugs.
- Our tool has been deployed with several product-development teams in IBM, and preliminary feedback has been positive

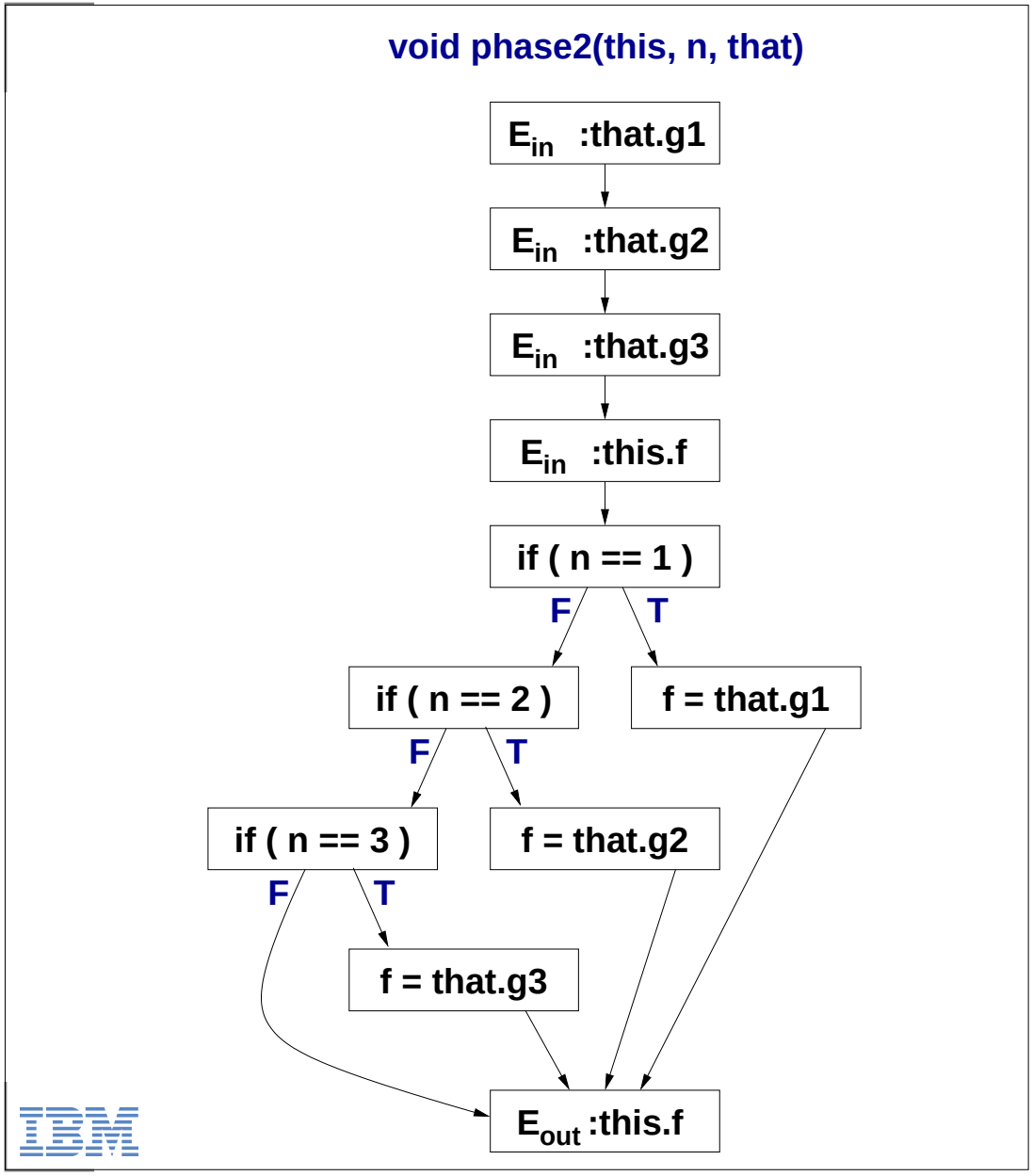
Future Work

- Comparison with sound analysis
- Prioritization of true positives
- Extension to other types of bugs, example resource-leak bugs

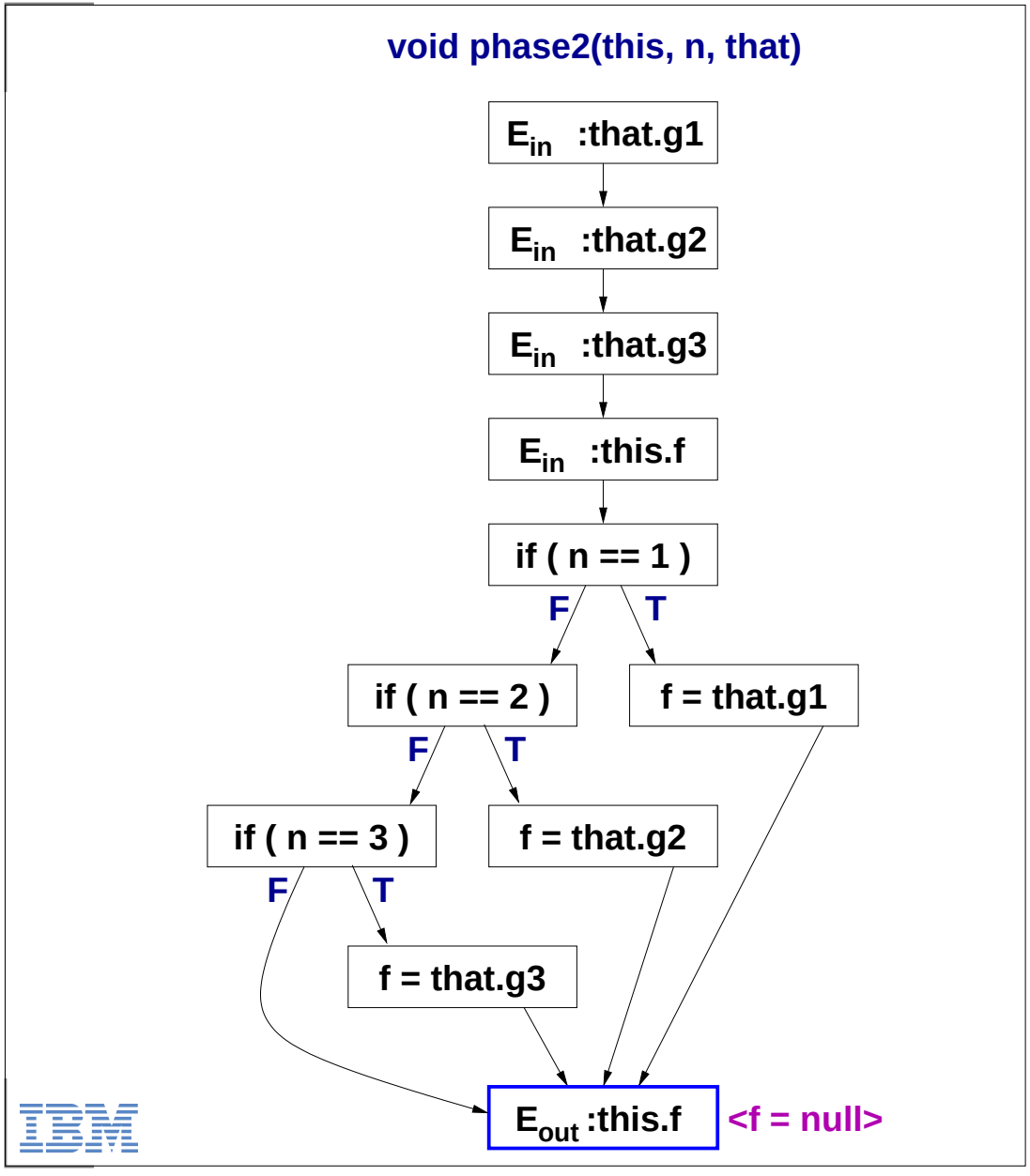
Genesis of the name Xylem

- XYLEM **is the transport tissue in plants whose basic function is to transport water from the root through the branches of the plant.**

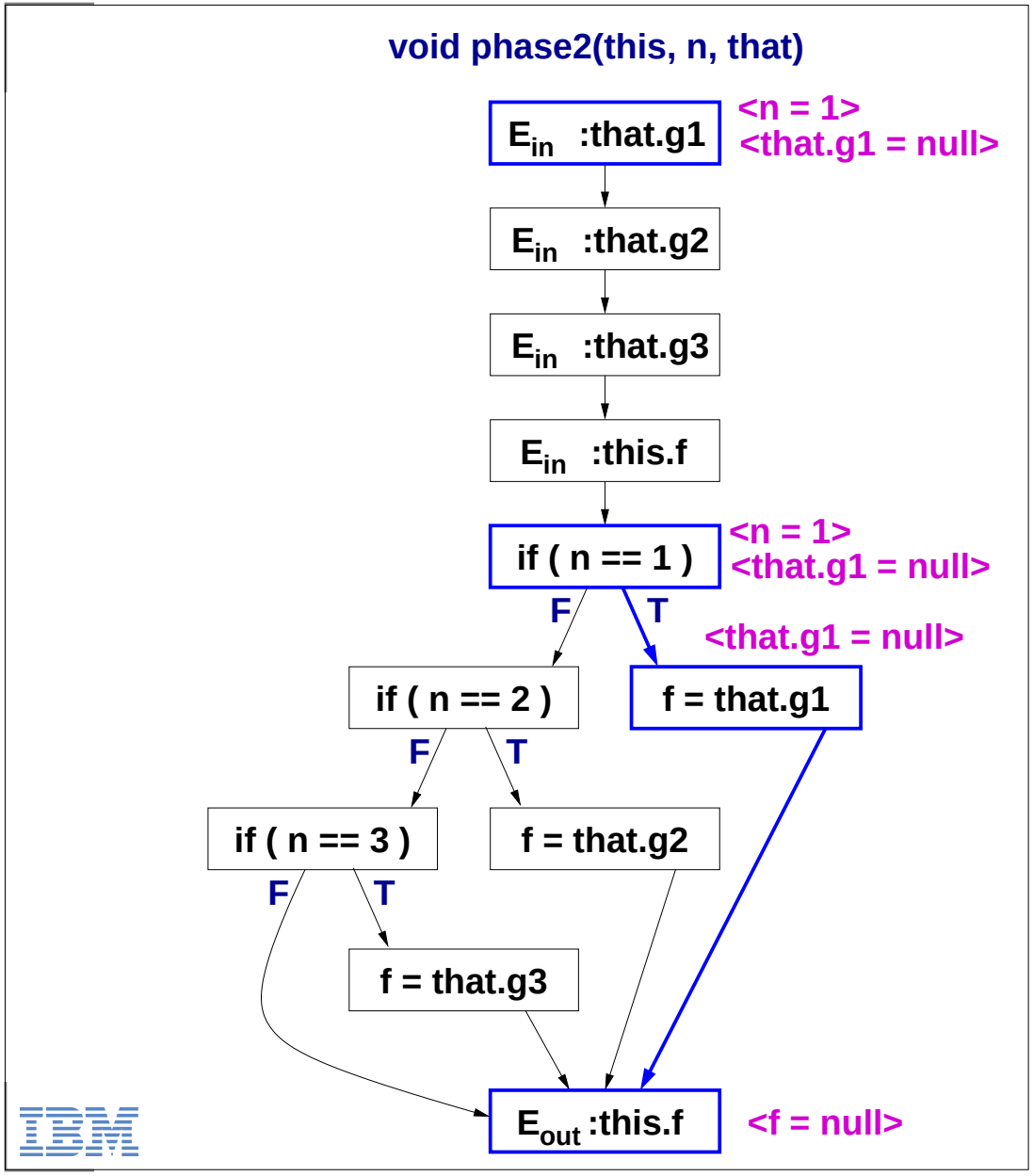
Genesis of the name Xylem



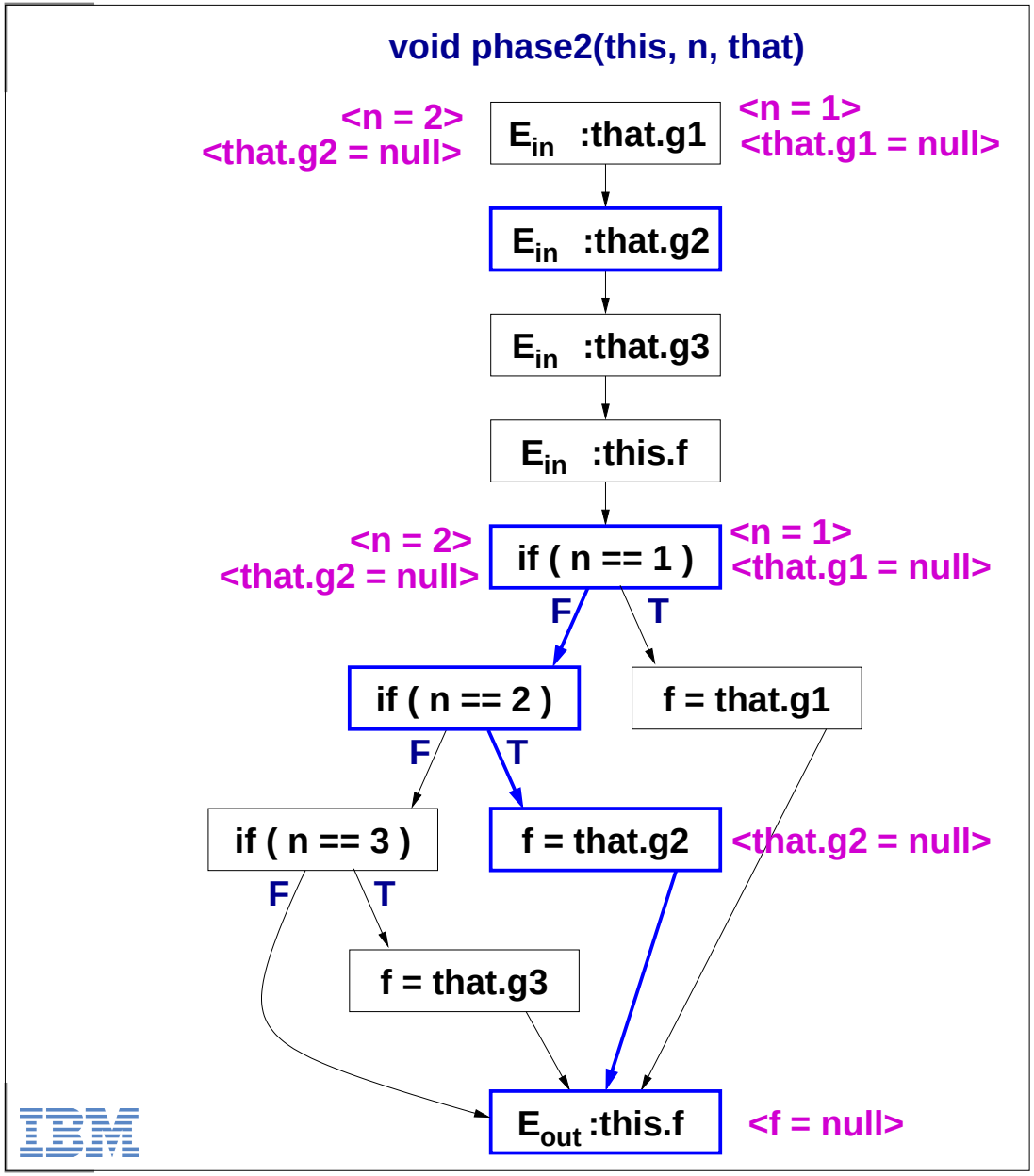
Genesis of the name Xylem



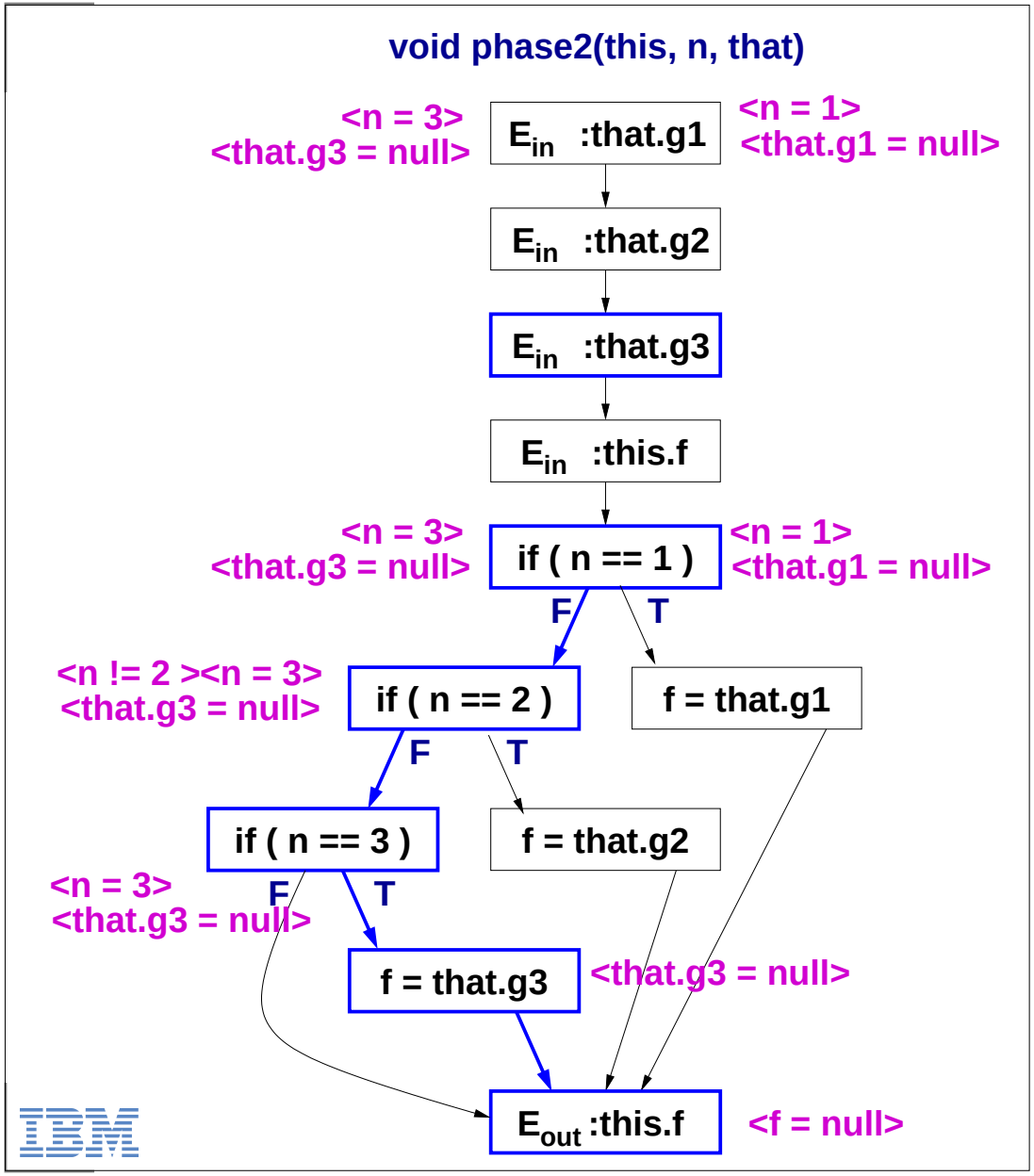
Genesis of the name Xylem



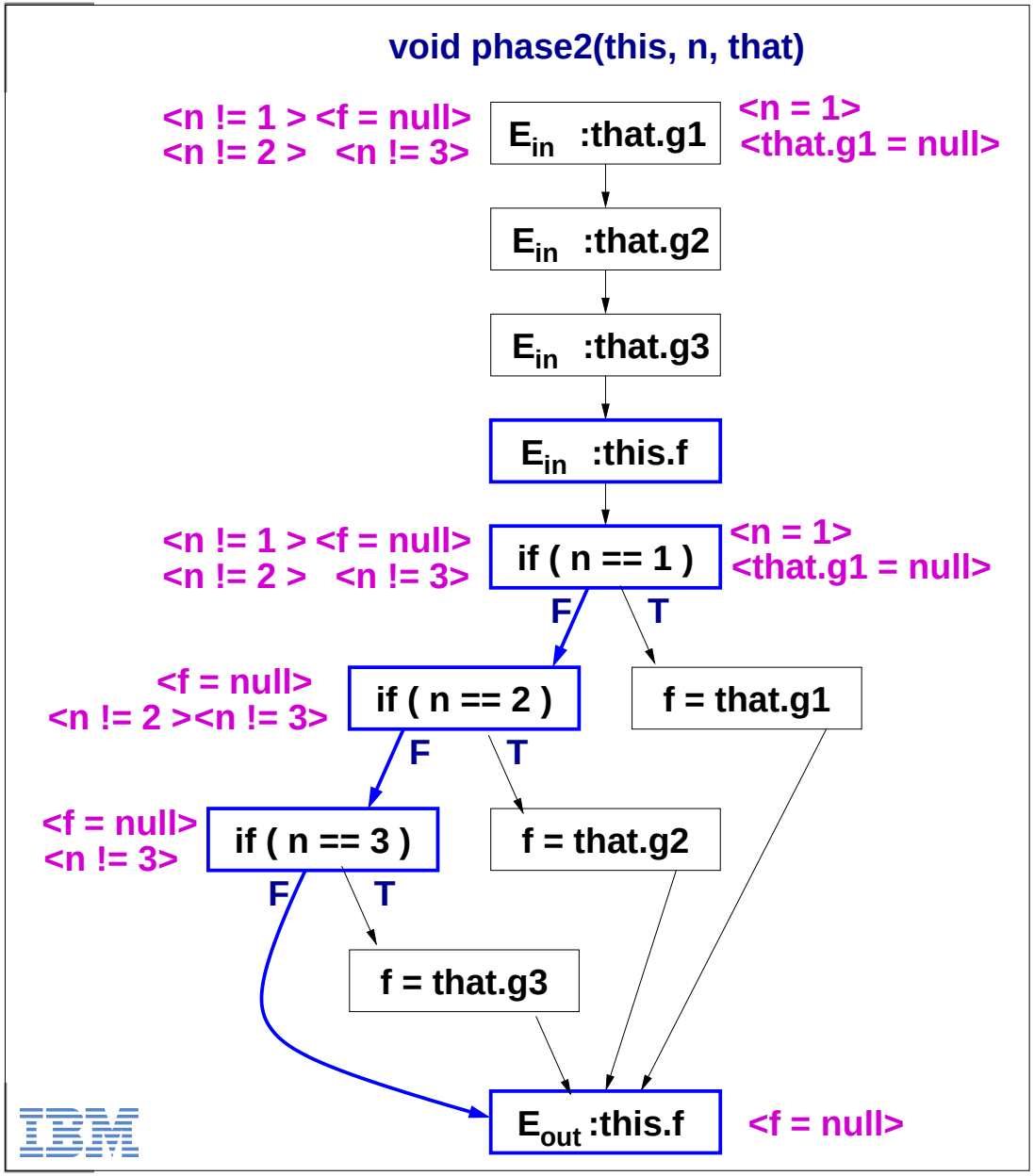
Genesis of the name Xylem



Genesis of the name Xylem



Genesis of the name Xylem



Questions