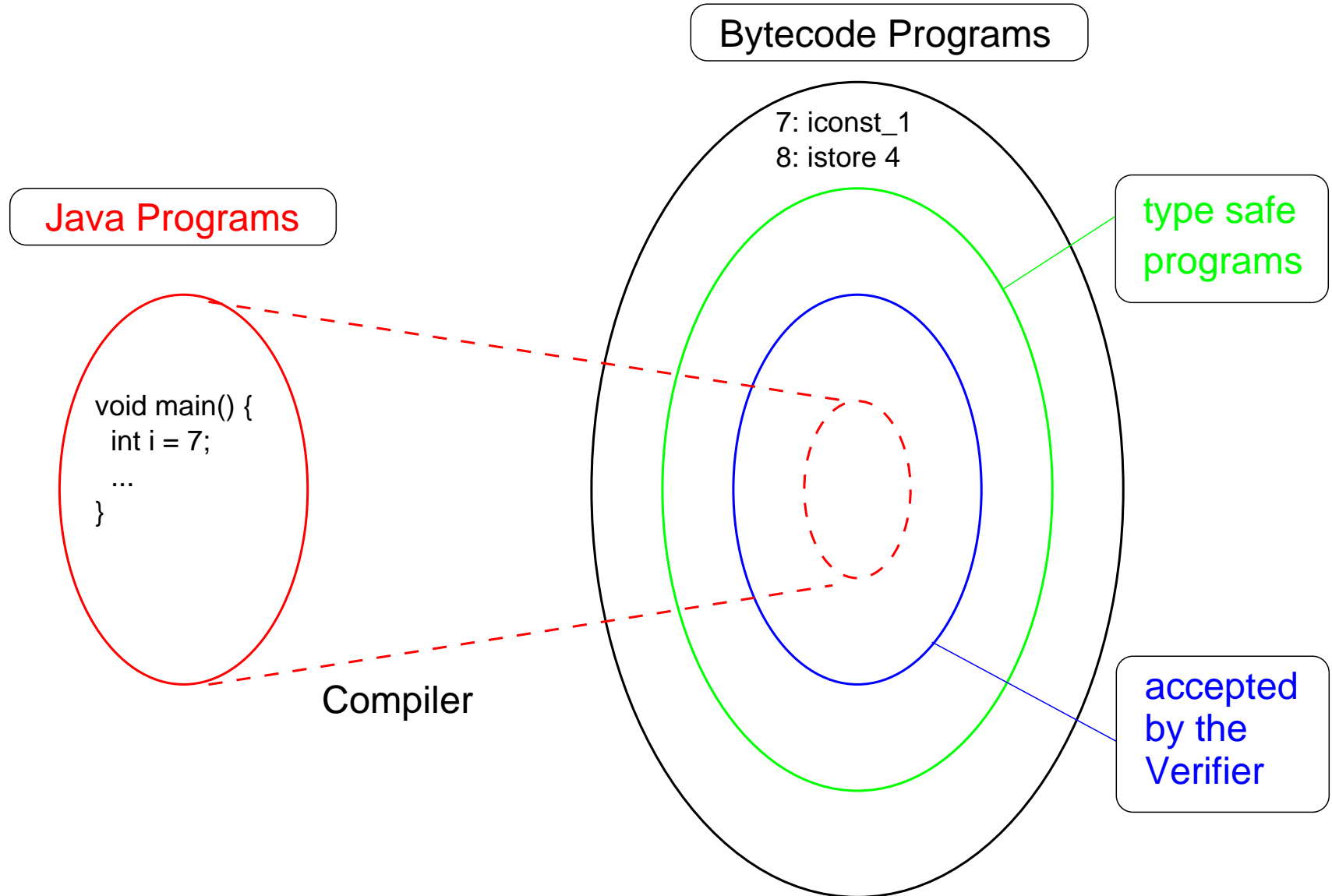
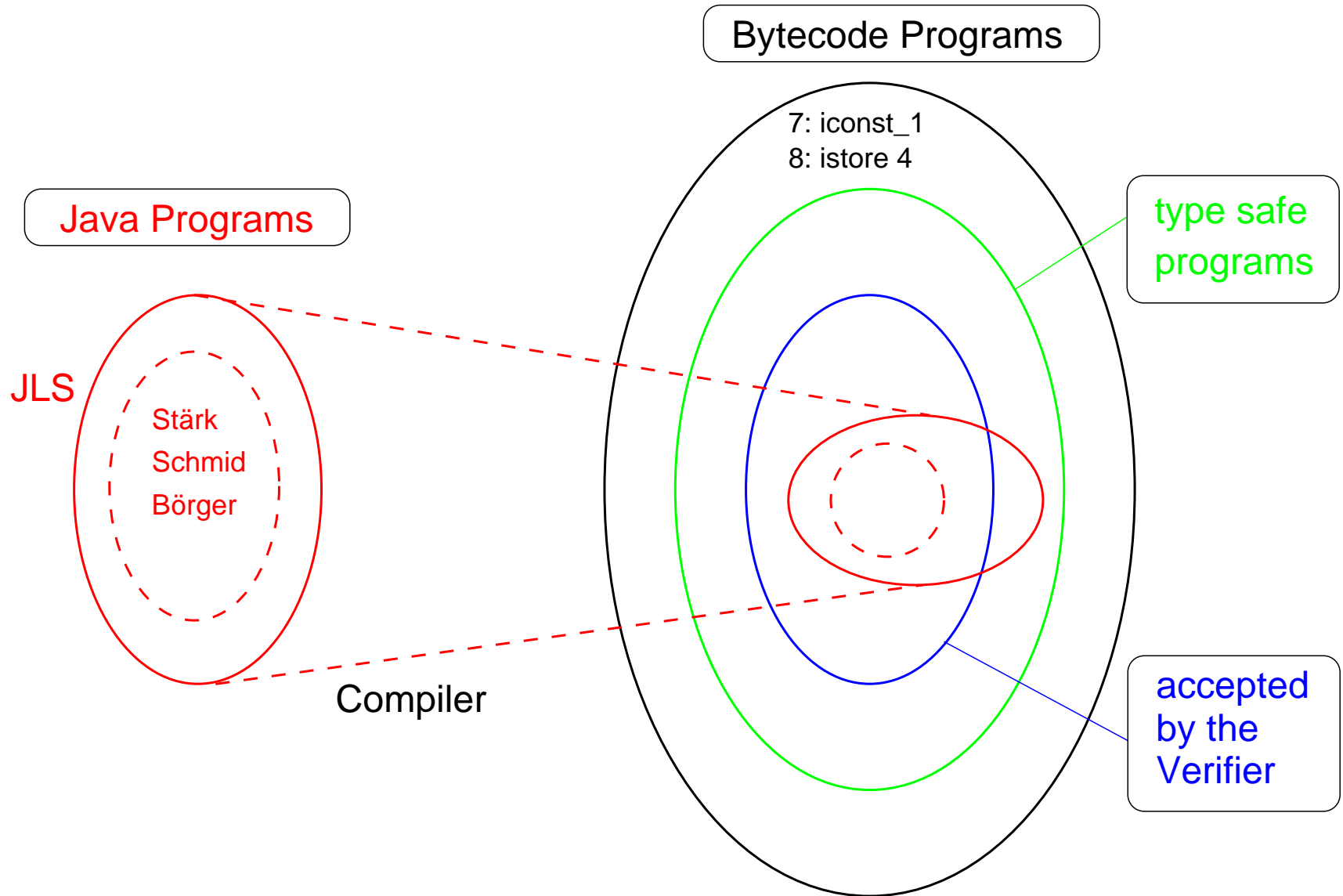


Java Bytecode Verification



Java Bytecode Verification (Reality)



Example 1: Legal Java program rejected by all verifiers

```
class Test1 {
  int test(boolean b) {
    int i;
    try {
      if (b) return 1;
      i = 2;
    } finally { if (b) i = 3; }
    return i;
  }
}
```

```
java version "1.3.0"
sun> javac Test1.java
sun> java Test1
```

java.lang.VerifyError: Register 2 contains wrong type

Kimera verifier: **Security flaw:** DFA_LOCVAR_WRONG_TYPE

Example 2: Legal Java program rejected by all verifiers

```
class Test2 {
  int test(boolean b) {
    int i;
    L: { try {
          if (b) return 1;
          i = 2;
          if (b) break L;
        } finally { if (b) i = 3; }
          i = 4;
        }
    return i;
  }
}
```

```
java version "1.3.0"
sun> javac Test2.java
sun> java Test2
```

java.lang.VerifyError: Register 2 contains wrong type

Problem: Subroutines are polymorphic

```
int test(int i) {  
    int j;  
    try {  
        if (i == 0)  
            return i * i;  
        j = i + i;  
    } finally { i = 0; }  
    return j + i;  
}
```

```
A: iload i  
   ifne B  
   iload i  
   iload i  
   imul  
   istore x  
   jsr S  
   iload x  
   ireturn
```

```
B: iload i  
   iload i  
   iadd  
   istore j
```

```
C: jsr S  
   goto E  
H: astore y  
   jsr S  
   aload y  
   athrow  
S: astore 4  
   iconst_0  
   istore i  
   ret 4  
E: iload j  
   iload i  
   iadd  
   ireturn
```

Exception table: catch Throwable from A to C using H

Remark: Subroutine S is polymorphic in x, j and y.

Breaking out of a subroutine to an enclosing subroutine

```
void test(boolean b) {
    try {
        return;
    } finally {
        while (b) {
            try {
                return;
            } finally {
                if (b) break;
            }
        }
    }
}

    jsr S1
    return
S1: astore r1
    goto W
A:   jsr S2
    return
S2: astore r2
    iload b
    ifne R1
    ret r2
W:   iload b
    ifne A
R1:  ret r1
```

Q: Does label **R1** belong to subroutine **S1** or **S2**?

Jumping out of a subroutine with an exception handler

```
void test(boolean b) {  
    try {  
        try {  
            return;  
        } finally {  
            if (b)  
                throw new E();  
        }  
    } catch (E x) {  
        return;  
    }  
}
```

```
A:  jsr S  
    return  
S:  astore r  
    iload b  
    ifeq B  
    new E  
    athrow  
B:  ret r  
H:  pop  
    return
```

catch **E** from A to H using **H**

Q: Does label **H** belong to subroutine **S**?

Which variables are modified by the subroutine?

```
void test(boolean b) {  
    while (true) {  
        try {  
            if (b) return;  
        } finally {  
            if (b) break;  
        }  
    }  
    b = true;  
}
```

```
A:  iload b  
    ifeq B  
    jsr S  
    return  
B:  jsr S  
    goto A  
S:  astore r  
    iload b  
    ifne E  
    ret r  
E:  iconst_1  
    istore b  
    return
```

Q: Is the variable **b** modified by the subroutine **S**?

Problem (Sun): Legal Java program rejected by the verifier

```
void test(boolean b) {  
    try {  
        try { if (b) return; }  
        finally {  
            try { if (b) return; }  
            finally { if (b) return; }  
        }  
    } finally { if (b) return; }  
}
```

```
sun> javac Test.java // JDK 1.3  
sun> java Test  
java.lang.VerifyError: Illegal return from subroutine
```

Remark: Flaw in Sun's bytecode verifier.

Bytecode Verification = Static Analysis + Type Inference

```
int test(boolean b) {      iload_1   ()      {1:int}
    int i;                 ifeq A    (int)    {1:int}
    try {                  iconst_1  ()      {1:int}
        if (b)             istore_3  (int)    {1:int}
            return 1;      jsr S     ()      {1:int,3:int}
            i = 2;         iload_3   ()      {1:int,3:int}
        } finally {       ireturn   (int)    {1:int,3:int}
            if (b)         A: iconst_2 ()      {1:int}
                i = 3;     istore_2  (int)    {1:int}
        }                 jsr S     ()      {1:int,2:int}
    return i;              goto C    ()      {1:int} // 2 modified by S
}                          S: astore 4  (ra(S)) {1:int}
                           iload_1   ()      {1:int,4:ra(S)}
                           ifeq B    (int)    {1:int,4:ra(S)}
                           iconst_3  ()      {1:int,4:ra(S)}
                           istore_2  (int)    {1:int,4:ra(S)}
                           B: ret 4    ()      {1:int,4:ra(S)}
                           C: iload_2  ()      {1:int}
                           ireturn   // 2 contains wrong type
```

Why sets of reference types?

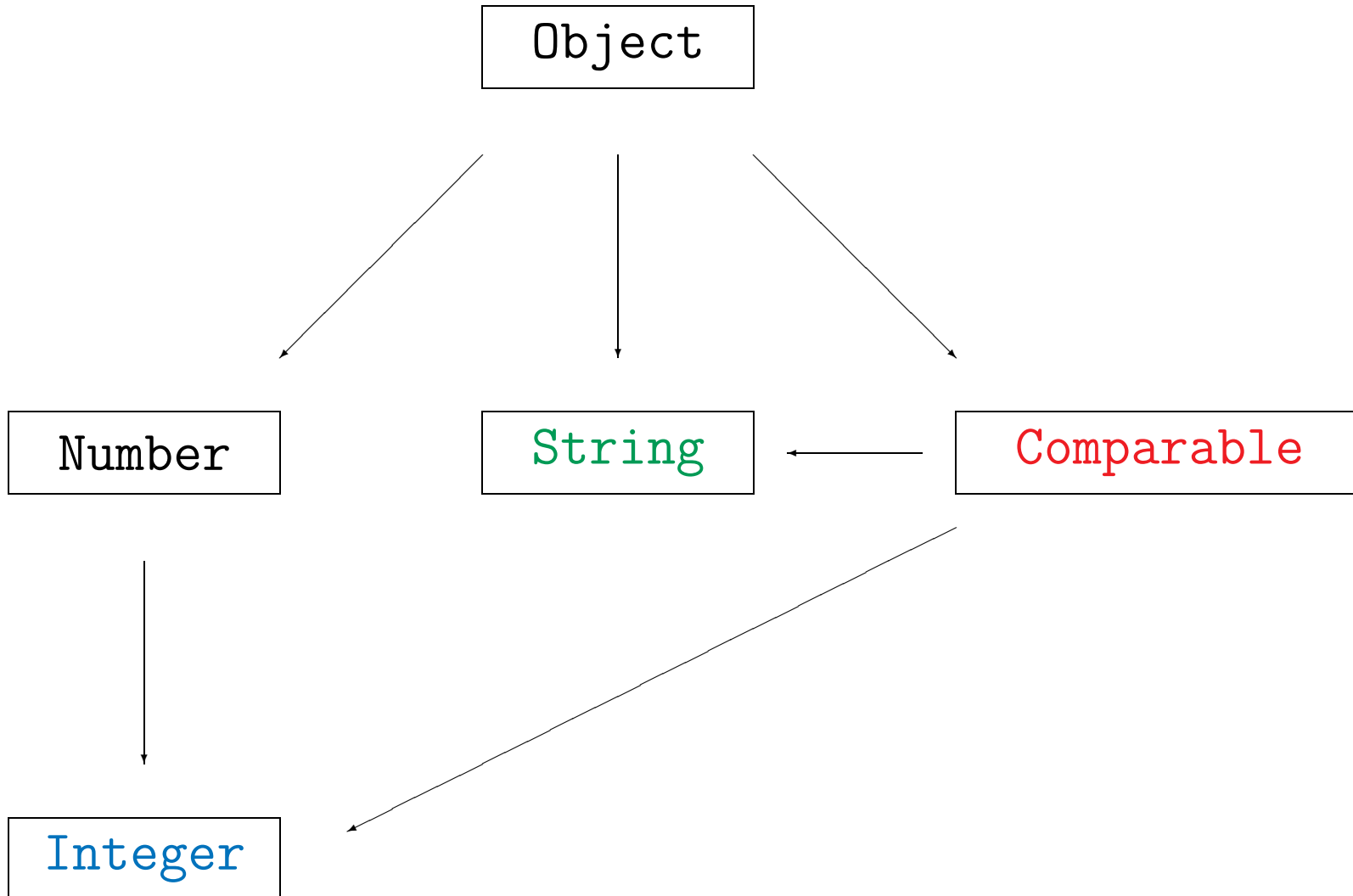
```
void m1(Integer i, String s) {
    Comparable x;
    if (i != null)
        x = i;
    else
        x = s;
    m2(x);
}

void m2(Comparable x) {}
```

```
    aload i
    ifnull A
    aload i
    astore x
    goto B
A:  aload s
    astore x
B:  aload_0
    aload x
    invoke m2(Comparable)
    return
```

Bytecode verifier: Type of `x` at B is `{Integer, String}`.

A fragment of the type hierarchy



Verify types

Refinement of verify types: finite sets of reference types

VerifyType = ...

| *Powerset*(Null | *Class* | *Interface* | *Array*)

Examples:

{Integer, String}, {Comparable}, {int[], float[]}

Refinement of \sqsubseteq for sets of reference types σ and τ :

$\sigma \sqsubseteq \tau$: \iff for each $A \in \sigma$ there exists a $B \in \tau$ such that $A \preceq B$.

Example:

{Integer, String} \sqsubseteq {Comparable}