

Software Model Checking Using Bogor - a Modular and Extensible Model Checking Framework

3rd Estonian Summer School in
Computer and System Science (ESSCaSS'04)

Slide Set 05: Representing Java in BIR

<http://bogor.projects.cis.ksu.edu>
<http://www.cis.ksu.edu/~hatcliff/ESSCaSS04>

John Hatcliff

Matthew B. Dwyer

Robby

SANToS Laboratory, Kansas State University, USA

Support

US Army Research Office (ARO)
US National Science Foundation (NSF)
US Department of Defense
Advanced Research Projects Agency (DARPA)

Boeing
Honeywell Technology Center
IBM
Intel

Lockheed Martin
NASA Langley
Rockwell-Collins ATC
Sun Microsystems

Motivation and Goals

- BIR/Bogor provides direct support for representing Java (and e.g., C#)
 - methods, exceptions, virtual method tables, references
- We will take a quick look at how Java is mapped to BIR.

Modeling Java Programs

Java classes and methods

```
public class A {
  public static int N;

  public int x;
  protected int y;

  public A() {...}

  public void foo() {...}
}

class B extends A {}

class C extends B {
  public void foo() {...}
}
```

```
record ([A])
  extends ([java.lang.Object])
  { int /|A.x|\; int /|A.y|\; }
  int \|A.N|/;

function {|A.<init>()|}
  (([A]) [|this|]) --

function {|A.foo()|} (([A]) [|this|]) --

record ([B]) extends ([A]) {}
record ([C]) extends ([B]) {}

function {|C.foo()|} (([C]) [|this|]) --

virtual +|A.foo()|+ {
  ([A]) -> {|A.foo()|}
  ([B]) -> {|A.foo()|}
  ([C]) -> {|C.foo()|}
}
```

Modeling Java Programs

Additional BIR Identifiers

```
public class A {
  public static int N;

  public int x;
  protected int y;

  public A() {...}

  public void foo() {...}
}

class B extends A {}

class C extends B {
  public void foo() {...}
}
```

```
record ([A])
  extends ([java.lang.Object])
  { int /|A.x|\; int /|A.y|\; }
  int \|A.N|/;

function {|A.<init>()|}
  (([A]) [|this|]) --

function {|A.foo()|} (([A]) [|this|]) --

record ([B]) extends ([A]) {}
record ([C]) extends ([B]) {}

function {|C.foo()|} (([C]) [|this|]) --

virtual +|A.foo()|+ {
  ([A]) -> {|A.foo()|}
  ([B]) -> {|A.foo()|}
  ([C]) -> {|C.foo()|}
}
```

$([...])$, $/|...|\backslash$, $\backslash|...|/$, $\{|...|\}$,
and $+|...|+$ are all BIR identifiers
used to avoid name clashes

Modeling Java Programs

Records for Java classes

records are used to model
Java classes (inheritance)

```
public class A {
  public static int N;

  public int x;
  protected int y;

  public A() {...}

  public void foo() {...}
}

class B extends A {}

class C extends B {
  public void foo() {...}
}
```

```
record ([A])
  extends ([java.lang.Object])
  { int /|A.x|\; int /|A.y|\; }
  int \|A.N|/;

function {|A.<init>()|}
  (([A]) [|this|]) --

function {|A.foo()|} (([A]) [|this|]) --

record ([B]) extends ([A]) {}
record ([C]) extends ([B]) {}

function {|C.foo()|} (([C]) [|this|]) --

virtual +|A.foo()|+ {
  ([A]) -> {|A.foo()|}
  ([B]) -> {|A.foo()|}
  ([C]) -> {|C.foo()|}
}
```

Modeling Java Programs

Static fields

static fields are modeled
as global variables

```
public class A {
  public static int N;

  public int x;
  protected int y;

  public A() {...}

  public void foo() {...}
}

class B extends A {}

class C extends B {
  public void foo() {...}
}
```

```
record ([A])
  extends ([java.lang.Object])
  { int /|A.x|\; int /|A.y|\; }
  int \|A.N|/;

function {|A.<init>()|}
  (([A]) [|this|]) --

function {|A.foo()|} (([A]) [|this|]) --

record ([B]) extends ([A]) {}
record ([C]) extends ([B]) {}

function {|C.foo()|} (([C]) [|this|]) --

virtual +|A.foo()|+ {
  ([A]) -> {|A.foo()|}
  ([B]) -> {|A.foo()|}
  ([C]) -> {|C.foo()|}
}
```

Modeling Java Programs

Java methods as functions

```
public class A {
    public static int N;
    public int x;
    protected int y;
    public A() {...}
    public void foo() {...}
}
class B extends A {}
class C extends B {
    public void foo() {...}
}
```

Java methods are modeled as functions

```
record ([A])
    extends ([java.lang.Object])
    { int /|A.x|\; int /|A.y|\; }
    int \|A.N|/;
    function {[A.<init>()]}
        (([A]) [|this|]) ...
    function {[A.foo()]} (([A]) [|this|]) ...
    record ([B]) extends ([A]) {}
    record ([C]) extends ([B]) {}
    function {[C.foo()]} (([C]) [|this|]) ...
    virtual +|[A.foo()]|+ {
        ([A]) -> {[A.foo()]}
        ([B]) -> {[A.foo()]}
        ([C]) -> {[C.foo()]}
    }
```

Modeling Java Programs

Dynamic dispatch of methods

```
public class A {
    public static int N;
    public int x;
    protected int y;
    public A() {...}
    public void foo() {...}
}
class B extends A {}
class C extends B {
    public void foo() {...}
}
```

```
record ([A])
    extends ([java.lang.Object])
    { int /|A.x|\; int /|A.y|\; }
    int \|A.N|/;
    function {[A.<init>()]}
        (([A]) [|this|]) ...
    function {[A.foo()]} (([A]) [|this|]) ...
    record ([B]) extends ([A]) {}
    record ([C]) extends ([B]) {}
    function {[C.foo()]} (([C]) [|this|]) ...
    virtual +|[A.foo()]|+ {
        ([A]) -> {[A.foo()]}
        ([B]) -> {[A.foo()]}
        ([C]) -> {[C.foo()]}
    }
```

Virtual tables are used to resolve dynamic dispatch of methods

Modeling Java Programs

Dynamic dispatch of methods

```
public class A {
    public static int N;
    public int x;
    protected int y;
    public A() {...}
    public void foo() {...}
}
class B extends A {}
class C extends B {
    public void foo() {...}
}
```

```
record ([A])
    extends ([java.lang.Object])
    { int /|A.x|\; int /|A.y|\; }
    int \|A.N|/;
    function {[A.<init>()]}
        (([A]) [|this|]) ...
    function {[A.foo()]} (([A]) [|this|]) ...
    record ([B]) extends ([A]) {}
    record ([C]) extends ([B]) {}
    function {[C.foo()]} (([C]) [|this|]) ...
    virtual +|[A.foo()]|+ {
        ([A]) -> {[A.foo()]}
        ([B]) -> {[A.foo()]}
        ([C]) -> {[C.foo()]}
    }
```

Modeling Java Programs

Dynamic dispatch of methods

```
function bar(([A]) a) {
    loc loc0; // invokespecial
    invoke {[A.foo()]} (a)
    goto loc0;
    loc loc1; // invokevirtual
    invoke virtual
    +|[A.foo()]|+ (a)
    goto loc1;
}
```

```
record ([A])
    extends ([java.lang.Object])
    { int /|A.x|\; int /|A.y|\; }
    int \|A.N|/;
    function {[A.<init>()]}
        (([A]) [|this|]) ...
    record ([B]) extends ([A]) {}
    record ([C]) extends ([B]) {}
    function {[C.foo()]} (([C]) [|this|]) ...
    virtual +|[A.foo()]|+ {
        ([A]) -> {[A.foo()]}
        ([B]) -> {[A.foo()]}
        ([C]) -> {[C.foo()]}
    }
```

BIR function invocations models Java method invocations (static, special, virtual, or interface)

Modeling Java Programs

Exceptions

```
public static void baz() {
    try {
        ...
    } catch (Throwable t) {
        ...
    }
}
```

Catch tables are used to keep track try-catch regions (order of declarations)

Throwable record

```
throwable -
    record ([java.lang.Throwable])
    extends ([java.lang.Object]) {...}
    function {[|baz|]}() {
        ([|java.lang.Throwable|] [|t|]);
        loc loc0; do { ... } goto loc1;
        loc loc1; do { ... } return;
        loc loc2; do { ... } goto loc1;
        catch ([|java.lang.Throwable|] [|t|])
            at loc0 goto loc2;
    }
```