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 04: Bogor Extensions

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

Customization Mechanisms

Bogor -- Extensible Modeling Language

Threads, Objects, Methods, Exceptions, etc. + Domain-Specific Abstractions

Core Modeling Language

Bogor -- Customizable Checking Engine Modules

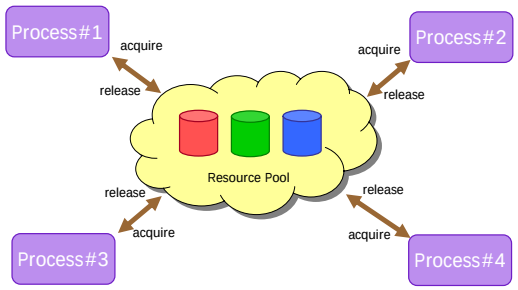
Scheduling Strategy | State-space Exploration | State Representation

Core Checker Modules

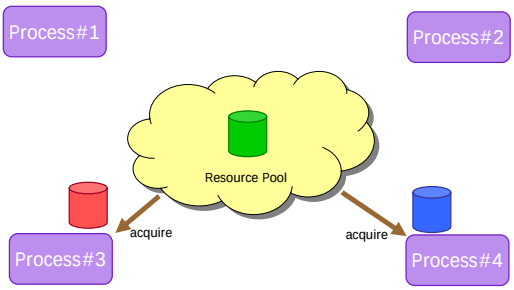
...existing modules... | Domain-Specific Scheduler | Domain-Specific Search | Domain-Specific State Rep.

Customized Checker Modules

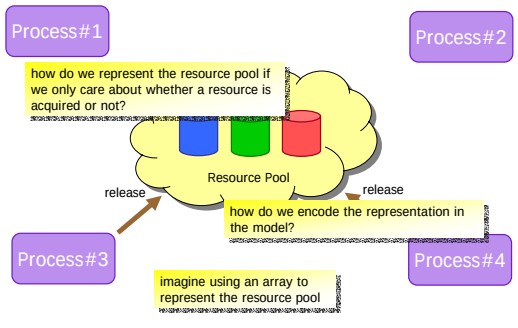
Extensible Input Language – Resource Contention Example



Extensible Input Language – Resource Contention Example



Extensible Input Language – Resource Contention Example



Domain-Specific Modeling

Bogor -- Extensible Modeling Language

Threads, Objects, Methods, Exceptions, etc. + **Set.type<'a>**

create()
choose()
+ **add()**
⋮
forAll()

```
class SetValue implements IValue {
  byte[] linearize(…) {
    Bogor API calls…
  } –
}
```

```
IValue choose(…) {
  Bogor API calls…
}
```

⋮

```
IValue forAll(…) {
  Bogor API calls…
}
```

Core Modeling Language

New Bogor types and primitives

Java implementation of new values and new primitives inside model-checker

Modeling Language Extensions

Bogor allows definitions of new abstract types and abstract operations as first-class constructs

```
extension Set for SetModule
{
  typedef type<'a>;

  expdef Set.type<'a> create<'a>('a ...);

  expdef 'a choose<'a>(Set.type<'a>);

  actiondef add<'a>(Set.type<'a>, 'a);

  actiondef remove<'a>(Set.type<'a>, 'a);

  expdef boolean forAll<'a>('a -> boolean, Set.type<'a>);
}
```

A new *type* to represent polymorphic symmetric sets

Modeling Language Extensions

Bogor allows definitions of new abstract types and abstract operations as first-class constructs

```
extension Set for SetModule
{
  typedef type<'a>;

  expdef Set.type<'a> create<'a>('a ...);

  expdef 'a choose<'a>(Set.type<'a>);

  actiondef add<'a>(Set.type<'a>, 'a);

  actiondef remove<'a>(Set.type<'a>, 'a);

  expdef boolean forAll<'a>('a -> boolean, Set.type<'a>);
}
```

Variable arity function for creating symmetric sets

Modeling Language Extensions

Bogor allows definitions of new abstract types and abstract operations as first-class constructs

```
extension Set for SetModule
{
  typedef type<'a>;

  expdef Set.type<'a> create<'a>('a ...);

  expdef 'a choose<'a>(Set.type<'a>);

  actiondef add<'a>(Set.type<'a>, 'a);

  actiondef remove<'a>(Set.type<'a>, 'a);

  expdef boolean forAll<'a>('a -> boolean, Set.type<'a>);
}
```

Non-deterministically pick an element of the set to return

Modeling Language Extensions

Bogor allows definitions of new abstract types and abstract operations as first-class constructs

```
extension Set for SetModule
{
  typedef type<'a>;

  expdef Set.type<'a> create<'a>('a ...);

  expdef 'a choose<'a>(Set.type<'a>);

  actiondef add<'a>(Set.type<'a>, 'a);

  actiondef remove<'a>(Set.type<'a>, 'a);

  expdef boolean forAll<'a>('a -> boolean, Set.type<'a>);
}
```

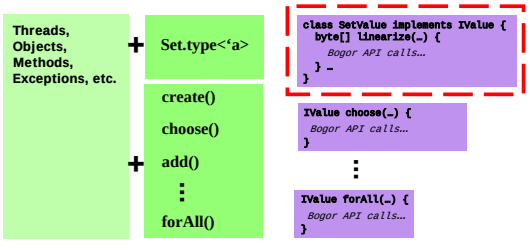
Higher-order function implements quantification over set elements

Predicate on set element

Set value to interate over

Domain-Specific Modeling

Bogor -- Extensible Modeling Language



Core Modeling Language

New Bogor types and primitives

Java implementation of new values and new primitives inside model-checker

Extension Implementation

Implementing the set value for the set type

- operations on the value
  - *e.g.*, add, remove, isEmpty, *etc.*
- visitor pattern for the value
  - used for linearization, *etc.*
- linearization for state storage
  - fine-grained control over the value representation
- XML externalization for counter-example display

Extension Implementation

Implementing the set value for the set type

```
public interface ISetValue
       extends INonPrimitiveExtValue {

  void add(IValue v);

  boolean contains(IValue v);

  IValue[] elements();

  boolean isEmpty();

  void remove(IValue v);
}
```

Create an interface for all implementations of set values

Extension Implementation

Implementing the set value for the set type

```
public class ReferenceElementSetValue
       implements ISetValue {

  protected HashSet set = new HashSet();

  public void add(IValue v) { set.add(v); }

  public boolean contains(IValue v) { retu

  public boolean isEmpty() { return set.size() == 0; }

  public void remove(IValue v) { set.remove(v); }

  public IValue[] elements() {
    Object[] elements = set.toArray();
    orderValues(elements);

    IValue[] result = new IValue[elements.length];
    System.arraycopy(elements, 0, result, 0, elements.length);

    return result;
  }
  …
```

A set value implementation where its elements are of reference types

Extension Implementation

Implementing the set value for the set type

```
public class ReferenceElementSetValue
       implements ISetValue {

  protected HashSet set = new HashSet();

  public void add(IValue v) { set.add(v); }

  public boolean contains(IValue v) { return set.contains(v); }

  public boolean isEmpty() { return set.size() == 0; }

  public void remove(IValue v) { set.remove(v); }

  public IValue[] elements() {
    Object[] elements = set.toArray();
    orderValues(elements);

    IValue[] result = new IValue[elements.length];
    System.arraycopy(elements, 0, result, 0, elements.length);

    return result;
  }
  …
```

Reuse Java collection class

Extension Implementation

Implementing the set value for the set type

```
public class ReferenceElementSetValue
       implements ISetValue {

  protected HashSet set = new HashSet();

  public void add(IValue v) { set.add(v); }

  public boolean contains(IValue v) { return set.contains(v); }

  public boolean isEmpty() { return set.size() == 0; }

  public void remove(IValue v) { set.remove(v); }

  public IValue[] elements() {
    Object[] elements = set.toArray();
    orderValues(elements);

    IValue[] result = new IValue[elements.length];
    System.arraycopy(elements, 0, result, 0, elements.length);

    return result;
  }
  …
```

Most set operations are wrapper calls to HashSet methods

order elements for consistency

Extension Implementation

Implementing the set value for the set type

- operations on the value
  - *e.g.*, add, remove, isEmpty, *etc.*
- visitor pattern for the value
  - used for garbage collection, *etc.*
- linearization for state storage
  - fine-grained control over the value representation
- XML externalization for counter-example display

Extension Implementation

Implementing the visitor pattern for set values

```
public class ReferenceElementSetValue implements ISetValue {

  public IValue[] elements() { … }

  public void visit(…, boolean depthFirst, Set seen,
        LinkedList workList, …) {

    IValue[] elements = elements();

    if (depthFirst) {
      for (int i = 0; i < elements.length; i++) {
        workList.addFirst(elements[i]);
      }
    } else {
      for (int i = 0; i < elements.length; i++) {
        workList.add(elements[i]);
      }
    }
  }
  …
```

Visitor pattern for traversing element values in the set (used for GC, symmetry, *etc.*)

Extension Implementation

Implementing the set value for the set type

- operations on the value
  - *e.g.*, add, remove, isEmpty, *etc.*
- visitor pattern for the value
  - used for garbage collection, *etc.*
- linearization for state storage
  - fine-grained control over the value representation
- XML externalization for counter-example display

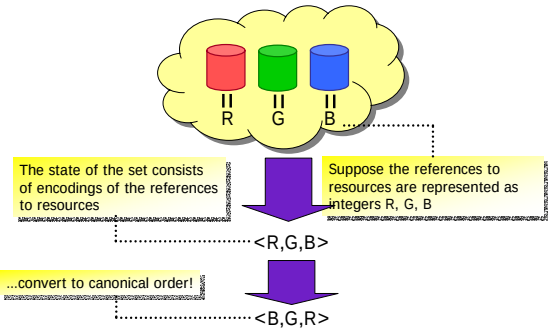
Extension Implementation

Implementing the linearization function

```
public class ReferenceElementSetValue implements ISetValue {

  public byte[][] linearize(int bitsPerNonPrimitiveValue,
                            ObjectIntTable nonPrimitiveValueIdMap, …) {
    IValue[] elements = elements();
    BitBuffer bb = new BitBuffer();

    if (elements.length > 0) {
      int[] elementIds = new int[elements.length];
      for (int i = 0; i < elements.length; i++) {
        elementIds[i] = nonPrimitiveValueIdMap.get(elements[i]);
      }

      Arrays.sort(elementIds);

      for (int i = 0; i < elements.length; i++) {
        bb.append(elementIds[i], bitsPerNonPrimitiveValue);
      }
    }

    return new byte[][] { bb.toByteArray() };
  }
  …
```

Convert the set value to a bit vector for state storage

Extensible Input Language – Set Extension Example (Semantics)



Extension Implementation

Implementing the set value for the set type

- operations on the value
  - *e.g.*, add, remove, isEmpty, *etc.*
- visitor pattern for the value
  - used for garbage collection, *etc.*
- linearization for state storage
  - fine-grained control over the value representation
- XML externalization for counter-example display

Extension Implementation

Implementing the XML externalization

```
public class ReferenceElementSetValue implements ISetValue {

  public void externalize(PrintWriter pw,
      INonPrimitiveValueIdTracker npvIdTracker) {
        IValue[] elements = elements();
        pw.println("<fields>");
        Type elementType = type.getTypeArg(0);
        for (int i = 0; i < elements.length; i++) {
            pw.println("<field>");
            pw.println("<id>element</id>");
            INonPrimitiveValue value =
                              (INonPrimitiveValue) elements[i];
            pw.print("<value type=\"");
            pw.print(Util.encodeXML(elementType.toString()));
            pw.print("\" val=\"");
            pw.print(npvIdTracker.getNonPrimitiveValueId(value));
            pw.print("\"");
            pw.println(" />");
            pw.println("</field>");
        }
        pw.println("</fields>");
        pw.flush();
    }
  …
```

Convert the set value to an XML representation for counter-example display

Extension Implementation

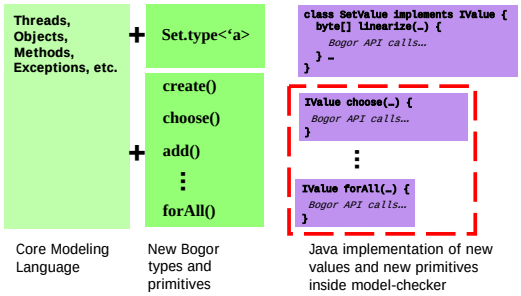
Implementing the XML externalization

```
<element id="Set.type&lt;Resource&gt;#1" isExt="true">
  <fields>
    <field>
      <id>element</id>
      <value type="Resource" val="Disk#1"/>
    </field>
    <field>
      <id>element</id>
      <value type="Resource" val="Disk#2"/>
    </field>
    <field>
      <id>element</id>
      <value type="Resource" val="Display#1"/>
    </field>
  </fields>
</element>
```

An example XML representation of a set value with three elements: Disk#1, Disk#2, Display#1

Domain-Specific Modeling

Bogor -- Extensible Modeling Language



Core Modeling Language

New Bogor types and primitives

Java implementation of new values and new primitives inside model-checker

Extension Implementation

Implementing the set operations for the set type

- set module extension
- set creation
- non-deterministically choose a set element
- adding an element

Extension Implementation

Implementing the set operations for the set type

```
public class SetModule implements IModule {

  protected SymbolTable symbolTable;
  protected TypeFactory tf;
  protected IExpEvaluator ee;
  protected IValueFactory vf;
  protected ISchedulingStrategist ss;

  public String getCopyrightNotice() { return null; }

  public void setOptions(String key, Properties options) {}

  public void connect(IBogorConfiguration bc) {
    symbolTable = bc.getSymbolTable();
    tf = symbolTable.getTypeFactory();
    ee = bc.getExpEvaluator();
    ss = bc.getSchedulingStrategist();
    vf = bc.getValueFactory();
  }

  public void dispose() {
    symbolTable = tf = ee = ss = vf = null;
  }
  …
```

The set module must implement IModule

Extension Implementation

Implementing the set operations for the set type

```
public class SetModule implements IModule {

  protected SymbolTable symbolTable;
  protected TypeFactory tf;
  protected IExpEvaluator ee;
  protected IValueFactory vf;
  protected ISchedulingStrategist ss;

  public String getCopyrightNotice() { return null; }

  public void setOptions(String key, Properties options) {}

  public void connect(IBogorConfiguration bc) {
    symbolTable = bc.getSymbolTable();
    tf = symbolTable.getTypeFactory();
    ee = bc.getExpEvaluator();
    ss = bc.getSchedulingStrategist();
    vf = bc.getValueFactory();
  }

  public void dispose() {
    symbolTable = tf = ee = ss = vf = null;
  }
  …
```

Store references to other Bogor modules as needed

Extension Implementation

Implementing the set operations for the set type

```
public class SetModule implements IModule {

  protected SymbolTable symbolTable;
  protected TypeFactory tf;
  protected IExpEvaluator ee;
  protected IValueFactory vf;
  protected ISchedulingStrategist ss;

  public String getCopyrightNotice() { return null; }

  public void setOptions(String key, Properties options) {}

  public void connect(IBogorConfiguration bc) {
    symbolTable = bc.getSymbolTable();
    tf = symbolTable.getTypeFactory();
    ee = bc.getExpEvaluator();
    ss = bc.getSchedulingStrategist();
    vf = bc.getValueFactory();
  }

  public void dispose() {
    symbolTable = tf = ee = ss = vf = null;
  }
  …
```

Used for displaying copyright notices

Extension Implementation

Implementing the set operations for the set type

```
public class SetModule implements IModule {

  protected SymbolTable symbolTable;
  protected TypeFactory tf;
  protected IExpEvaluator ee;
  protected IValueFactory vf;
  protected ISchedulingStrategist ss;

  public String getCopyrightNotice() { return null; }

  public void setOptions(String key, Properties options) {}

  public void connect(IBogorConfiguration bc) {
    symbolTable = bc.getSymbolTable();
    tf = symbolTable.getTypeFactory();
    ee = bc.getExpEvaluator();
    ss = bc.getSchedulingStrategist();
    vf = bc.getValueFactory();
  }

  public void dispose() {
    symbolTable = tf = ee = ss = vf = null;
  }
  …
```

Used for setting options of Bogor modules

Extension Implementation

Implementing the set operations for the set type

```
public class SetModule implements IModule {

  protected SymbolTable symbolTable;
  protected TypeFactory tf;
  protected IExpEvaluator ee;
  protected IValueFactory vf;
  protected ISchedulingStrategist ss;

  public String getCopyrightNotice() { return null; }

  public void setOptions(String key, Properties options) {}

  public void connect(IBogorConfiguration bc) {
    symbolTable = bc.getSymbolTable();
    tf = symbolTable.getTypeFactory();
    ee = bc.getExpEvaluator();
    ss = bc.getSchedulingStrategist();
    vf = bc.getValueFactory();
  }

  public void dispose() {
    symbolTable = tf = ee = ss = vf = null;
  }
  …
```

Connect to other Bogor modules

Extension Implementation

Implementing the set operations for the set type

```
public class SetModule implements IModule {

  protected SymbolTable symbolTable;
  protected TypeFactory tf;
  protected IExpEvaluator ee;
  protected IValueFactory vf;
  protected ISchedulingStrategist ss;

  public String getCopyrightNotice() { return null; }

  public void setOptions(String key, Properties options) {}

  public void connect(IBogorConfiguration bc) {
    symbolTable = bc.getSymbolTable();
    tf = symbolTable.getTypeFactory();
    ee = bc.getExpEvaluator();
    ss = bc.getSchedulingStrategist();
    vf = bc.getValueFactory();
  }

  public void dispose() {
    symbolTable = tf = ee = ss = vf = null;
  }
  …
```

Unlink references to other Bogor modules

Extension Implementation

Implementing the set operations for the set type

- set module extension
- set creation
- non-deterministically choose a set element
- adding an element

Extension Implementation

Implementing the set operations for the set type

```
public class SetModule implements IModule {

  // expdef Set.type<'a> create<'a>('a ...);
  public IValue create(IExtArguments arg) {

    if (arg.getTypeVariableArgument(0)
                  instanceof NonPrimitiveType) {

      ISetValue result = new ReferenceElementSetValue(
                        (NonPrimitiveExtType) arg.getExpType(),
                        vf.newReferenceId());

      int size = arg.getArgumentCount();

      for (int i = 0; i < size; i++) {
              result.add(arg.getArgument(i));
      }

      return result;
    }
    else if …
  }
  …
```

If the type argument is reference type then use the previously implemented set value

Extension Implementation

Implementing the set operations for the set type

```
public class SetModule implements IModule {

  // expdef Set.type<'a> create<'a>('a ...);
  public IValue create(IExtArguments arg) {

    if (arg.getTypeVariableArgument(0)
                  instanceof NonPrimitiveType) {

      ISetValue result = new ReferenceElementSetValue(
                        (NonPrimitiveExtType) arg.getExpType(),
                        vf.newReferenceId());

      int size = arg.getArgumentCount();

      for (int i = 0; i < size; i++) {
              result.add(arg.getArgument(i));
      }

      return result;
    }
    else if …
  }
  …
```

Add arguments to the set value

Extension Implementation

Implementing the set operations for the set type

```
public class SetModule implements IModule {

  // expdef Set.type<'a> create<'a>('a ...);
  public IValue create(IExtArguments arg) {

    if (arg.getTypeVariableArgument(0)
                  instanceof NonPrimitiveType) {

      ISetValue result = new ReferenceElementSetValue(
                        (NonPrimitiveExtType) arg.getExpType(),
                        vf.newReferenceId());

      int size = arg.getArgumentCount();

      for (int i = 0; i < size; i++) {
              result.add(arg.getArgument(i));
      }

      return result;
    }
    else if …
  }
  …
```

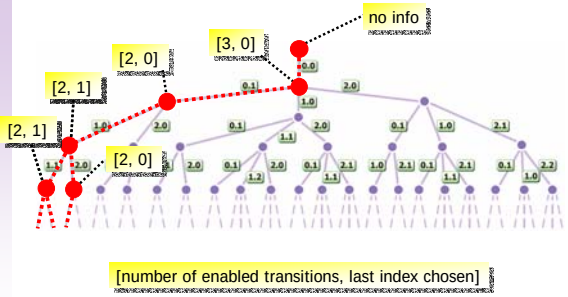
other element types are handled differently (more efficiently)

Extension Implementation

Implementing the set operations for the set type

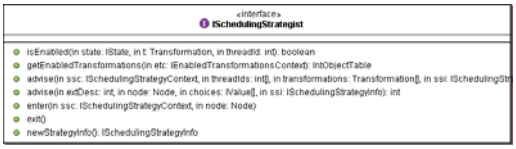
- set module extension
- set creation
- non-deterministically choose a set element
- adding an element

DefaultSchedulingStrategist & DefaultSchedulingStrategyInfo



[number of enabled transitions, last index chosen]

ISchedulingStrategist



- Used to determine
  - enabled transitions: isEnabled(), getEnabledTransformations()
  - which transition to take: advise()
- create strategy info

Extension Implementation

Implementing the set operations for the set type

```
public class SetModule implements IModule {

  // expdef 'a choose<'a>(Set.type<'a>);
  public IValue choose(IExtArguments arg) {

    ISetValue set = (ISetValue) arg.getArgument(0);

    IValue[] elements = set.elements();

    int size = elements.length;

    int index = 0;

    if (size > 1) {
      index = ss.advise(arg.getExtDesc(),
                        arg.getNode(),
                        elements,
                        arg.getSchedulingStrategyInfo());
    }

    return elements[index];
  }
  …
```

Get the set value

Extension Implementation

Implementing the set operations for the set type

```
public class SetModule implements IModule {

  // expdef 'a choose<'a>(Set.type<'a>);
  public IValue choose(IExtArguments arg) {

    ISetValue set = (ISetValue) arg.getArgument(0);

    IValue[] elements = set.elements();

    int size = elements.length;

    int index = 0;

    if (size > 1) {
      index = ss.advise(arg.getExtDesc(),
                        arg.getNode(),
                        elements,
                        arg.getSchedulingStrategyInfo());
    }

    return elements[index];
  }
  …
```

Get the set elements

Extension Implementation

Implementing the set operations for the set type

```
public class SetModule implements IModule {

  // expdef 'a choose<'a>(Set.type<'a>);
  public IValue choose(IExtArguments arg) {

    ISetValue set = (ISetValue) arg.getArgument(0);

    IValue[] elements = set.elements();

    int size = elements.length;

    int index = 0;

    if (size > 1) {
      index = ss.advise(arg.getExtDesc(),
                        arg.getNode(),
                        elements,
                        arg.getSchedulingStrategyInfo());
    }

    return elements[index];
  }
  …
```

Ask the scheduler which one to pick if there are two or more elements

Extension Implementation

Implementing the set operations for the set type

- set module extension
- set creation
- non-deterministically choose a set element
- adding an element

Extension Implementation

Implementing the set operations for the set type

```
public class SetModule implements IModule {

  // actiondef add<'a>(Set.type<'a>, 'a);
  public IBacktrackingInfo[] add(IExtArguments arg) {
    ISetValue set = (ISetValue) arg.getArgument(0);

    IValue element = (IValue) arg.getArgument(1);

    if (!set.contains(element)) {
      set.add(element);

      ISchedulingStrategyContext ssc =
           arg.getSchedulingStrategyContext();

      return new IBacktrackingInfo[] {
        createAddBacktrackingInfo(set, element, arg.getNode()
                          ssc.getStateId(), ssc.getThreadId(),
                              arg.getSchedulingStrategyInfo())

      };
    } else {
      return new IBacktrackingInfo[0];
    }
  }
…
```

Get the set and the element

Extension Implementation

Implementing the set operations for the set type

```
public class SetModule implements IModule {

  // actiondef add<'a>(Set.type<'a>, 'a);
  public IBacktrackingInfo[] add(IExtArguments arg) {
    ISetValue set = (ISetValue) arg.getArgument(0);

    IValue element = (IValue) arg.getArgument(1);

    if (!set.contains(element)) {
      set.add(element);

      ISchedulingStrategyContext ssc =
           arg.getSchedulingStrategyContext();

      return new IBacktrackingInfo[] {
        createAddBacktrackingInfo(set, element, arg.getNode()
                          ssc.getStateId(), ssc.getThreadId(),
                              arg.getSchedulingStrategyInfo())

      };
    } else {
      return new IBacktrackingInfo[0];
    }
  }
…
```

Add the element if it is not already in the set value

Extension Implementation

Implementing the set operations for the set type

```
public class SetModule implements IModule {

  // actiondef add<'a>(Set.type<'a>, 'a);
  public IBacktrackingInfo[] add(IExtArguments arg) {
    ISetValue set = (ISetValue) arg.getArgument(0);

    IValue element = (IValue) arg.getArgument(1);

    if (!set.contains(element)) {
      set.add(element);

      ISchedulingStrategyContext ssc =
           arg.getSchedulingStrategyContext();

      return new IBacktrackingInfo[] {
        createAddBacktrackingInfo(set, element, arg.getNode()
                          ssc.getStateId(), ssc.getThreadId(),
                              arg.getSchedulingStrategyInfo())

      };
    } else {
      return new IBacktrackingInfo[0];
    }
  }
…
```

Create the backtracking information

Extension Implementation

Implementing the set operations for the set type

```
public class SetModule implements IModule {

  // actiondef add<'a>(Set.type<'a>, 'a);
  public IBacktrackingInfo[] add(IExtArguments arg) {
    ISetValue set = (ISetValue) arg.getArgument(0);

    IValue element = (IValue) arg.getArgument(1);

    if (!set.contains(element)) {
      set.add(element);

      ISchedulingStrategyContext ssc =
           arg.getSchedulingStrategyContext();

      return new IBacktrackingInfo[] {
        createAddBacktrackingInfo(set, element, arg.getNode()
                          ssc.getStateId(), ssc.getThreadId(),
                              arg.getSchedulingStrategyInfo())

      };
    } else {
      return new IBacktrackingInfo[0];
    }
  }
…
```

If the element is already in the set then do nothing

Extension Implementation

Implementing the set operations for the set type

```
public class SetModule implements IModule {

  protected IBacktrackingInfo createAddBacktrackingInfo(
                final ISetValue set, final IValue element, … ) {

    return new IBacktrackingInfo() {

      public void backtrack(IState state) {
        set.remove(element);
      }

      public IBacktrackingInfo clone(Map cloneMap) {
        return createAddBacktrackingInfo(
          (ISetValue) cloneMap.get(set),
          (IValue) ((element instanceof INonPrimitiveValue) ?
                      cloneMap.get(element) : element),
          node, stateId, threadId, ssi.clone(cloneMap));
      }
…
    };
  }
…
```

Backtrack by removing the element from the set

Extension Implementation

Implementing the set operations for the set type

```
public class SetModule implements IModule {

  protected IBacktrackingInfo createAddBacktrackingInfo(
                final ISetValue set, final IValue element, … ) {

    return new IBacktrackingInfo() {

      public void backtrack(IState state) {
        set.remove(element);
      }

      public IBacktrackingInfo clone(Map cloneMap) {
        return createAddBacktrackingInfo(
          (ISetValue) cloneMap.get(set),
          (IValue) ((element instanceof INonPrimitiveValue) ?
                    cloneMap.get(element) : element),
           node, stateId, threadId, ssi.clone(cloneMap));
      }
      …
    };
  }
  …
```

(Deep) clone the backtracking information

A BIR Example – Resource Contention

Demo

- Model with the Set extension
- Incorporating the extension in the Bogor Eclipse plugin
- Run the example
  - invalid end state
  - invariant checking

Assessment

- Bogor provides a clean and well-designed framework for extending its modeling language
  - Allows introduction of new abstract types and abstract operations as first-class construct
    - Complete control over value representation (linearization)
    - No double interpretation (the operations are executed as atomic actions in the model checker instead of being interpreted by the model checker)
  - Analogous to adding new instructions to a Virtual Machine
    - essentially, we are building abstract machines for particular domains