A Model Checker for Synchronous Java

VO Duc-Duy
EPFL – Lausanne
Switzerland
sJava program

Bandera and BOGOR

Properties

Satisfied!

Counter-Example
Model a **Synchronous Java** program

Extend from a model checker
Model a **Synchronous Java** program
Communicating between threads ...

**Object A**

```java
active class A{
    run () {
        . . .
        select {
            case
            objB.b(msg) ;
            case
            accept a;
        }
        . . .
    }
}
```

**Object B**

```java
active class B{
    void b (String msg)
    {
        . . .
    }
    run () {
        accept b;
    }
}
```
Communicating between threads ...

Object A

```java
active class A{
  run () {
    ...
    select {
      case 
      objB.b(msg) ;
      case 
      accept a;
    }
    ...
  }
}
```

Object B

```java
active class B{
  void b (String msg)
  {
    ..........
  }
  run () {
    accept b;
  }
}
```
Communicating between threads …

Object A

```java
active class A{
    run () {
        ...
        select {
            case 
                objB.b(msg) ;
            case 
                accept a;
        }
        ...
    }
}
```

Object B

```java
active class B{
    void b (String msg)
    {
        .......... 
    }
    run () {
        accept b;
    }
}
```

rendezvous
Model an **active** object

**active** class A{
    A b = new A();
    ....
    public void run(){
        for(;;){
            select{
                case b.metha();
                case accept metha;
            }
            b.metha();
        }
    }
}
Interaction btw $a$, $b$ produces a deadlock.

Reachable states …
An Extension to Bandera and Bogor
(1): Computer Networking laboratory, EPFL, Switzerland.
(2), (3): SAnToS laboratory, Kansas State University, USA.
Bogor Architecture

Front-End
- Lexer
- Parser
- Type Checking
- Semantic Analyses

Model Checking Components
- IActionTaker
- IBacktrackIF
- IStateMgr
- IExpEvaluator
- ITransformer
- IStateFactory
- IValueFactory
- ISchedulingStg
- ISearcher

Extension Syntax
- .bir

Extension Implementation
- .config

Verified

Counter Example
Bogor Modelling Language (BIR)

• Guarded command language:

```
loc loc1:
  when guard_x do{}
  goto loc2;
loc loc2:
  when guard_y1 do{}
  goto loc4;
  when guard_y2 do{}
  goto loc3;
```

• Native support for variety object-oriented language features
  - Dynamically created objects and threads, exceptions, methods, inheritance, etc.
Synchronization in SBlir

A

loc loc0:
  when true do {
    addACall(clist, 1, call_B);
    addAnAcpt(clist, 2, acpt_A);
    pc = syncWait(clist);
  } goto loc1;

loc loc1:
  when pc == 1 do
    B.method();
    goto loc2;
  when pc == 2 do
    acceptA();
    goto loc3;

loc loc3:
  ...;

when pc == 2 do -> TRUE
  acceptB();
  goto loc2;

Search if there is any interaction btw threads.
There is NO: pc = 0

Call-accept btw A&B found -> Update A.pc=1, B.pc=2

B

loc loc0:
  when true do {
    addACall(clist, 1, call_C);
    addAnAcpt(clist, 2, acpt_B);
    pc = syncWait(clist);
  } goto loc1;

loc loc1:
  when pc == 1 do
    goto loc2;
  when pc == 2 do
    acceptB();
    goto loc2;

loc loc3:
  ...;

loc loc2:
  do {}
  goto loc4;

loc loc4:
  ...
Result of a simple deadlock.

```java
to thread MAIN {}{
   loc loc0: live {} |
   invoke {{|Deadlock.main(java.lang)|}}
   return;
}
```
Conclusion

• The Synchronous Java language allows us to perform the same kind of analyses as the one made upon formal languages.

• The extending part for verifying Synchronous Java programs has been tested. It can be used to detect deadlocks of simple models.
THANK YOU!