Combining Model Learning and Model Checking to Analyze Java Libraries

Shahbaz Ali, Hailong Sun, Zhaowang
bazgikian@yahoo.com
Beihang University, China.

Specifications of LogIn Utility Module

Specification 1: User must not LogIn until he register first

Specification 2: abc

Specification 3: xyz

Specification 3: etc.

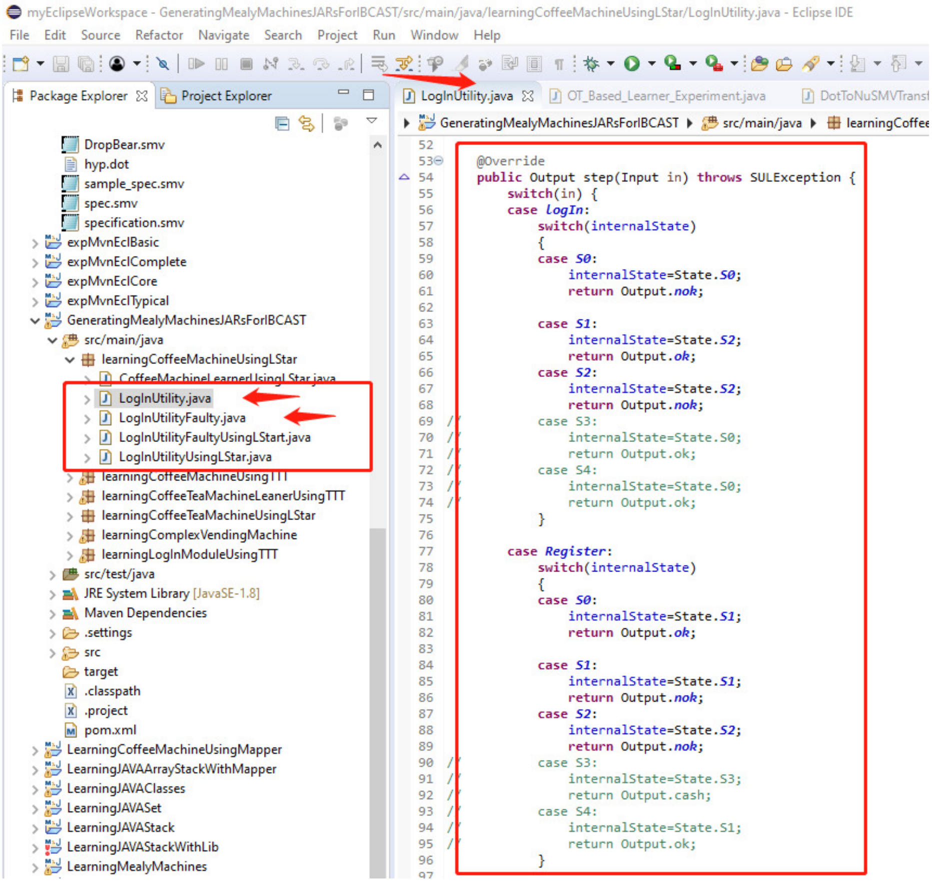
Step-1: We implement a simple LogIn utility scenario in java language, as shown in below source code:

```
//Source Code Example of LogInutility Class

//LogInUtility.java

public class LogInUtility {

//Define Input Alphabet

        public enum Input{
                LogIn,
                Register,
                LogOut
        }

//Define Output Alphabet


        public enum Output{
                nok,
                ok
        }

        public enum State{
                S0,
                S1,
                S2
        }

public State internalState = State.S0;
```

Combining Model Learning and Model Checking to Analyze Java Libraries

Shahbaz Ali, Hailong Sun, Zhaowang
bazgikian@yahoo.com
Beihang University, China.

```
@Override
public void pre() {

        internalState = State.S0;
}

@Override
public void post() {

}

@Override

public Output step(Input in {
        Switch(in) {

        case LogIn:
                switch(internalState)
                {
                case S0:
                        internalState=State.S0;
                        return Output.nok;

                case S1:
                        internalState=State.S2;
                        return Output.ok;
                case S2:
                        internalState=State.S2;
                        return Output.nok;
                }

        case Register:
                switch(internalState)
                {
                case S0:
                        internalState=State.S1;
                        return Output.ok;

                case S1:
                        internalState=State.S1;
                        return Output.nok;
                case S2:
                        internalState=State.S2;
                        return Output.nok;
                }

        case LogOut:
                switch(internalState)
                {
                case S0:
                        internalState=State.S0;
                        return Output.nok;
                        case S1:
                                internalState=State.S1;
                                return Output.nok;
                        case S2:
                                internalState=State.S1;
                                return Output.ok;
                default:
                throw new IllegalArgumentException("Unknown input " + in);
               }

                //return null;
        }
}
```

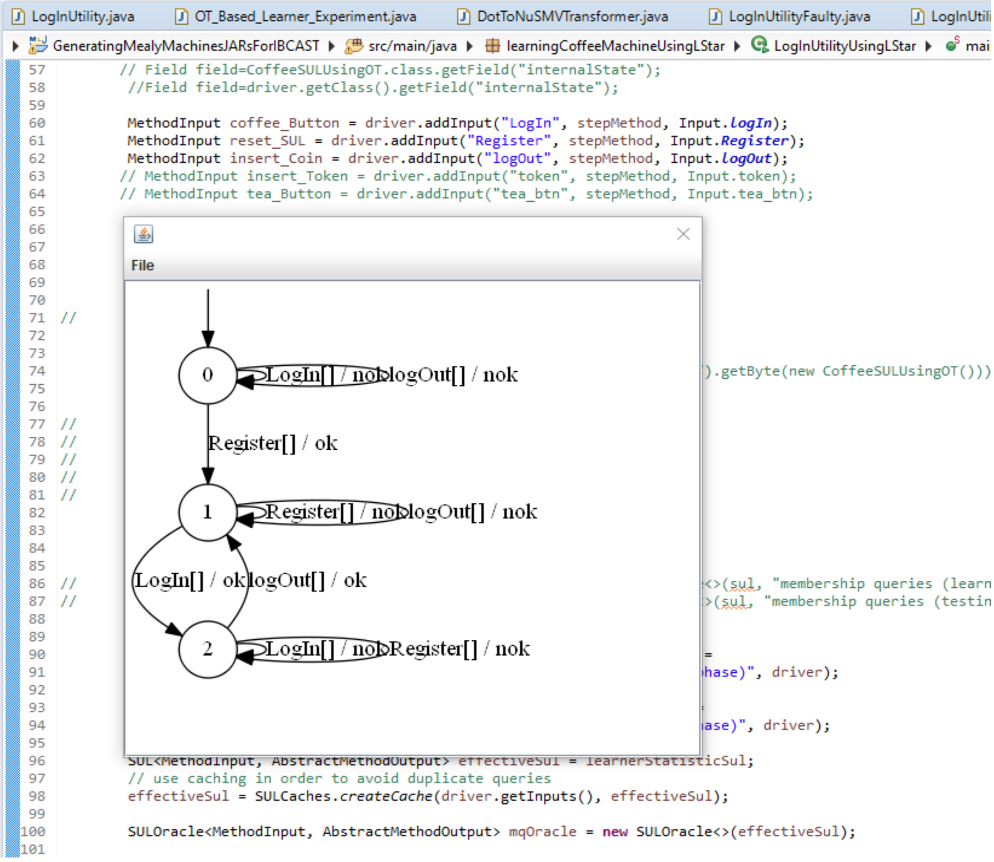

myEclipseWorkspace - GeneratingMealyMachinesJARsForIBCAST/src/main/java/learningCoffeeMachineUsingLStar/LogInUtility.java - Eclipse IDE
File
Edit
Source
Refactor
Navigate
Search
Project
Run
Window
Help
Package Explorer
Project Explorer
LogInUtility.java
OT_Based_Learner_Experiment.java
DotToNuSMVTransl
GeneratingMealyMachinesJARsForIBCAST
src/main/java
learningCoffee
DropBear.smv
hyp.dot
sample_spec.smv
spec.smv
specification.smv
expMvnEclBasic
expMvnEclComplete
expMvnEclCore
expMvnEclTypical
GeneratingMealyMachinesJARsForIBCAST
src/main/java
learningCoffeeMachineUsingLStar
CoffeeMachineLearnerUsingLStar.java
LogInUtility.java
LogInUtilityFaulty.java
LogInUtilityFaultyUsingLStart.java
LogInUtilityUsingLStar.java
learningCoffeeMachineUsingTTT
learningCoffeeTeaMachineLeanerUsingTTT
learningCoffeeTeaMachineUsingLStar
learningComplexVendingMachine
learningLogInModuleUsingTTT
src/test/java
JRE System Library [JavaSE-1.8]
Maven Dependencies
.settings
src
target
.classpath
.project
pom.xml
LearningCoffeeMachineUsingMapper
LearningJAVAArrayStackWithMapper
LearningJAVAClasses
LearningJAVASet
LearningJAVAStack
LearningJAVAStackWithLib
LearningMealyMachines

```

    @Override
    public Output step(Input in) throws SULException {
        switch(in) {
        case logIn:
            switch(internalState)
            {
            case S0:
                internalState=State.S0;
                return Output.nok;

            case S1:
                internalState=State.S2;
                return Output.ok;
            case S2:
                internalState=State.S2;
                return Output.nok;
//          case S3:
//              internalState=State.S0;
//              return Output.ok;
//          case S4:
//              internalState=State.S0;
//              return Output.ok;
            }

        case Register:
            switch(internalState)
            {
            case S0:
                internalState=State.S1;
                return Output.ok;

            case S1:
                internalState=State.S1;
                return Output.nok;
            case S2:
                internalState=State.S2;
                return Output.nok;
//          case S3:
//              internalState=State.S3;
//              return Output.cash;
//          case S4:
//              internalState=State.S1;
//              return Output.ok;
            }
```

Step-2: We execute the above source code using our suggested model learning setup, and get the following behavior model of login utility module.

Combining Model Learning and Model Checking to Analyze Java Libraries

Shahbaz Ali, Hailong Sun, Zhaowang
bazgikian@yahoo.com
Beihang University, China.

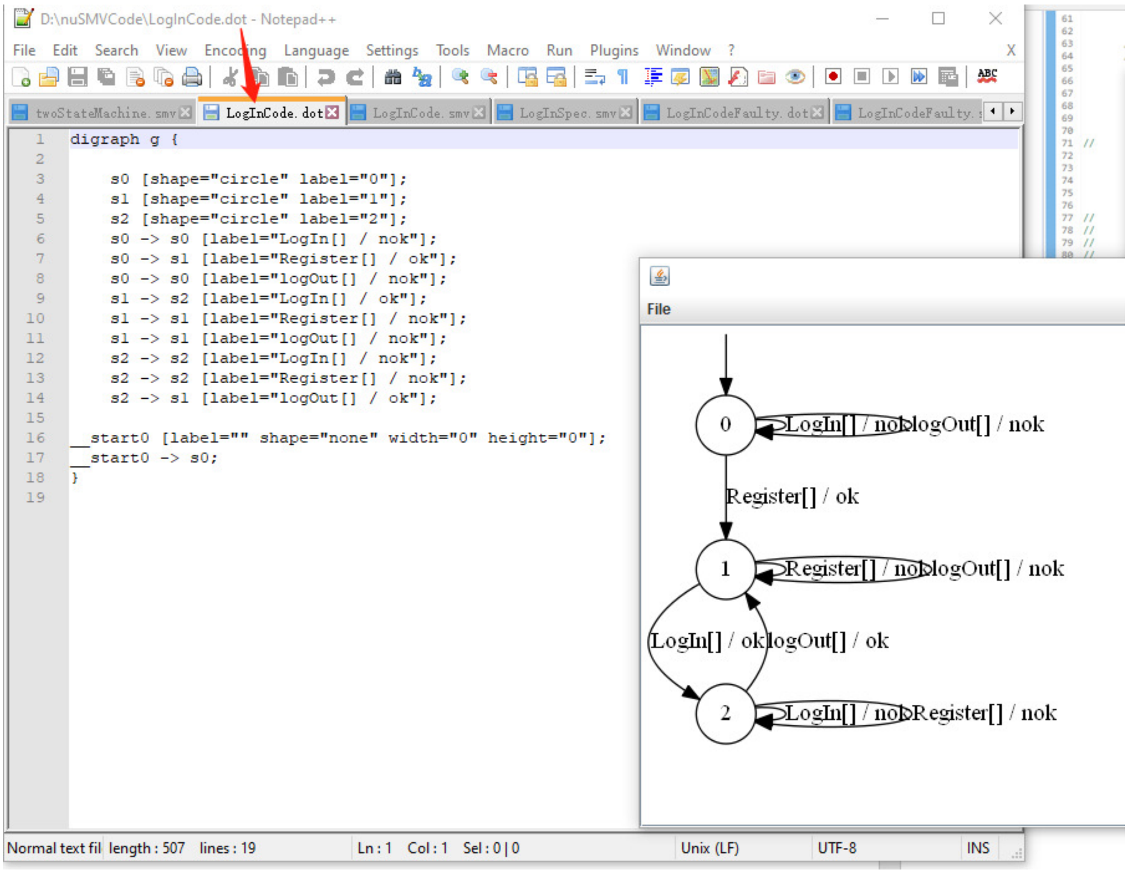


```
        // Field field=CoffeeSULUsingOT.class.getField("internalState");
         //Field field=driver.getClass().getField("internalState");

         MethodInput coffee_Button = driver.addInput("LogIn", stepMethod, Input.logIn);
         MethodInput reset_SUL = driver.addInput("Register", stepMethod, Input.Register);
         MethodInput insert_Coin = driver.addInput("logOut", stepMethod, Input.logOut);
        // MethodInput insert_Token = driver.addInput("token", stepMethod, Input.token);
        // MethodInput tea_Button = driver.addInput("tea_btn", stepMethod, Input.tea_btn);






 //


                                                                      ).getByte(new CoffeeSULUsingOT()))


 //
 //
 //
 //
 //




 //                                                                   <>(sul, "membership queries (learn
 //                                                                   >(sul, "membership queries (testin


                                                                      =
                                                                      hase)", driver);


                                                                      ase)", driver);

         SUL<MethodInput, AbstractMethodOutput> effectiveSul = learnerStatisticSul;
         // use caching in order to avoid duplicate queries
         effectiveSul = SULCaches.createCache(driver.getInputs(), effectiveSul);

         SULOracle<MethodInput, AbstractMethodOutput> mqOracle = new SULOracle<>(effectiveSul);

```

And below is the behavior model, output file of model learning phase, in the form of dot language. We save the model with name “LogInCode.dot”.

Combining Model Learning and Model Checking to Analyze Java Libraries

Shahbaz Ali, Hailong Sun, Zhaowang
bazgikian@yahoo.com
Beihang University, China.



```
digraph g {

    s0 [shape="circle" label="0"];
    s1 [shape="circle" label="1"];
    s2 [shape="circle" label="2"];
    s0 -> s0 [label="LogIn[] / nok"];
    s0 -> s1 [label="Register[] / ok"];
    s0 -> s0 [label="logOut[] / nok"];
    s1 -> s2 [label="LogIn[] / ok"];
    s1 -> s1 [label="Register[] / nok"];
    s1 -> s1 [label="logOut[] / nok"];
    s2 -> s2 [label="LogIn[] / nok"];
    s2 -> s2 [label="Register[] / nok"];
    s2 -> s1 [label="logOut[] / ok"];

__start0 [label="" shape="none" width="0" height="0"];
__start0 -> s0;
}
```

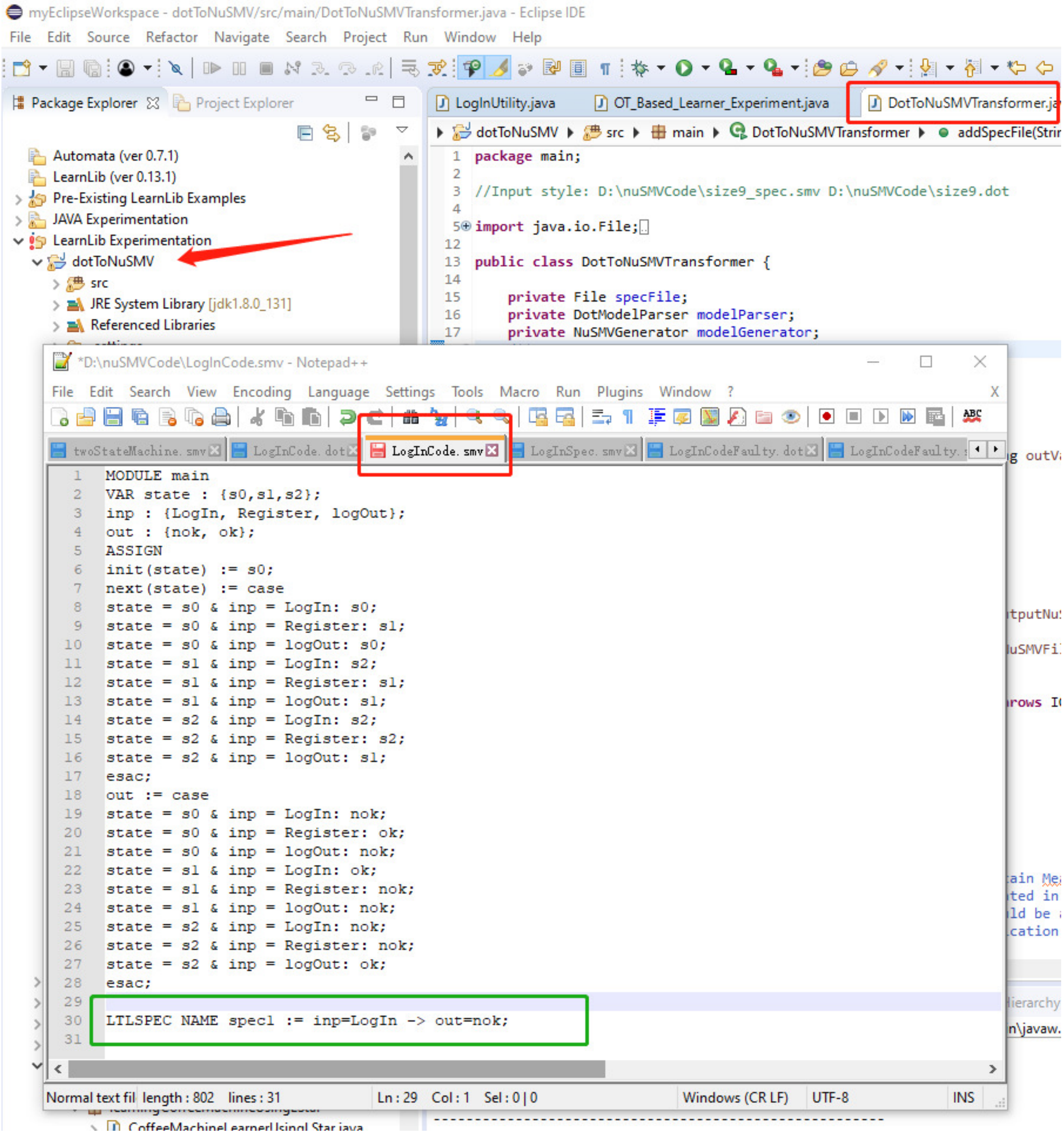
Step-3: Next, we transform the dot model into .smv format using “dotToNuSMVTransformer” utility. In the resultant LogInCode.smv format, we append the specification to be checked i..e, User must not LogIn until he register first. We specify the property using LTL specification language.

The following specification represents the case that at the start (when user has not created LogIn account), if the input is LogIn (i.e., user tries to Login) then output should be “Not OK” i.e., the system should deny log in.

LTLSPEC NAME spec1 := inp=LogIn -> out=nok

Combining Model Learning and Model Checking to Analyze Java Libraries

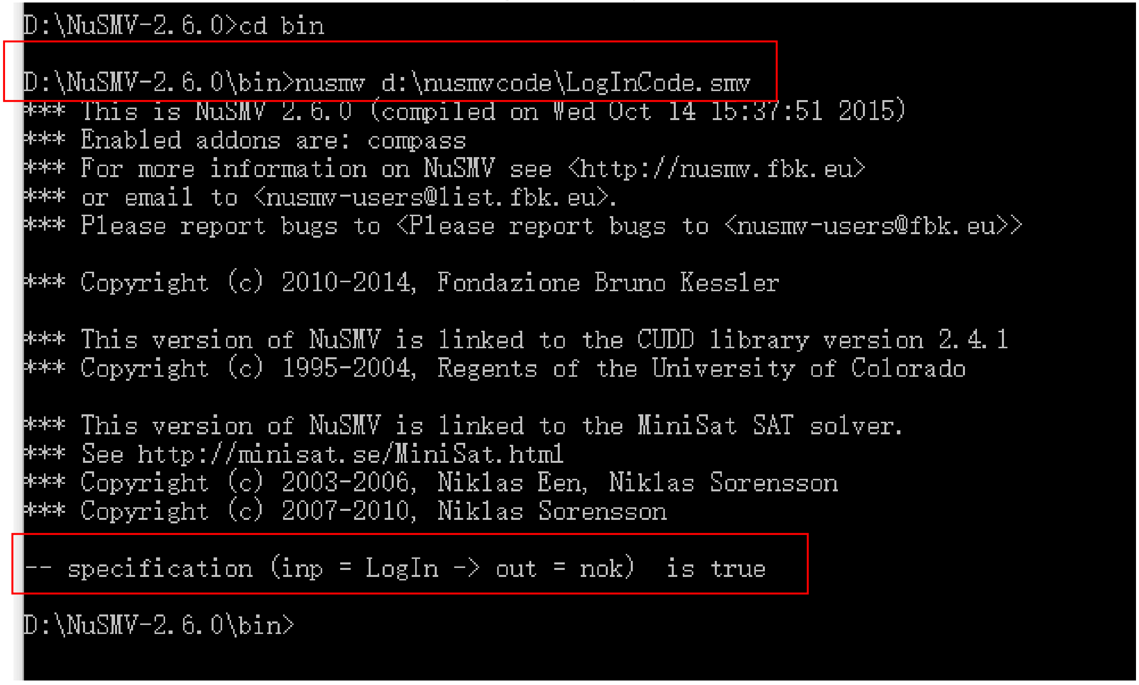
Shahbaz Ali, Hailong Sun, Zhaowang
bazgikian@yahoo.com
Beihang University, China.


myEclipseWorkspace - dotToNuSMV/src/main/DotToNuSMVTransformer.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer
Project Explorer
LogInUtility.java
OT_Based_Learner_Experiment.java
DotToNuSMVTransformer.ja
Automata (ver 0.7.1)
LearnLib (ver 0.13.1)
Pre-Existing LearnLib Examples
JAVA Experimentation
LearnLib Experimentation
dotToNuSMV
src
JRE System Library [jdk1.8.0_131]
Referenced Libraries
dotToNuSMV ▸ src ▸ main ▸ DotToNuSMVTransformer ▸ addSpecFile(Strir
*D:\nuSMVCode\LogInCode.smv - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
twoStateMachine.smv
LogInCode.dot
LogInCode.smv
LogInSpec.smv
LogInCodeFaulty.dot
LogInCodeFaulty.
Normal text fil length : 802 lines : 31 Ln : 29 Col : 1 Sel : 0 | 0 Windows (CR LF) UTF-8 INS
CoffeeMachineLearnerUsingLStar.java

```
package main;

//Input style: D:\nuSMVCode\size9_spec.smv D:\nuSMVCode\size9.dot

import java.io.File;

public class DotToNuSMVTransformer {

    private File specFile;
    private DotModelParser modelParser;
    private NuSMVGenerator modelGenerator;
```

```
MODULE main
VAR state : {s0,s1,s2};
inp : {LogIn, Register, logOut};
out : {nok, ok};
ASSIGN
init(state) := s0;
next(state) := case
state = s0 & inp = LogIn: s0;
state = s0 & inp = Register: s1;
state = s0 & inp = logOut: s0;
state = s1 & inp = LogIn: s2;
state = s1 & inp = Register: s1;
state = s1 & inp = logOut: s1;
state = s2 & inp = LogIn: s2;
state = s2 & inp = Register: s2;
state = s2 & inp = logOut: s1;
esac;
out := case
state = s0 & inp = LogIn: nok;
state = s0 & inp = Register: ok;
state = s0 & inp = logOut: nok;
state = s1 & inp = LogIn: ok;
state = s1 & inp = Register: nok;
state = s1 & inp = logOut: nok;
state = s2 & inp = LogIn: nok;
state = s2 & inp = Register: nok;
state = s2 & inp = logOut: ok;
esac;

LTLSPEC NAME spec1 := inp=LogIn -> out=nok;
```

Step-4: Next, we run the model checker i.e., NuSMV and observed that the system has validate/verify the property (as per our expectation that system should behave like this). The output of the model checking has been shown in the below figure.



```
D:\NuSMV-2.6.0>cd bin

D:\NuSMV-2.6.0\bin>nusmv d:\nusmvcode\LogInCode.smv
*** This is NuSMV 2.6.0 (compiled on Wed Oct 14 15:37:51 2015)
*** Enabled addons are: compass
*** For more information on NuSMV see <http://nusmv.fbk.eu>
*** or email to <nusmv-users@list.fbk.eu>.
*** Please report bugs to <Please report bugs to <nusmv-users@fbk.eu>>

*** Copyright (c) 2010-2014, Fondazione Bruno Kessler

*** This version of NuSMV is linked to the CUDD library version 2.4.1
*** Copyright (c) 1995-2004, Regents of the University of Colorado

*** This version of NuSMV is linked to the MiniSat SAT solver.
*** See http://minisat.se/MiniSat.html
*** Copyright (c) 2003-2006, Niklas Een, Niklas Sorensson
*** Copyright (c) 2007-2010, Niklas Sorensson

-- specification (inp = LogIn -> out = nok)  is true

D:\NuSMV-2.6.0\bin>
```

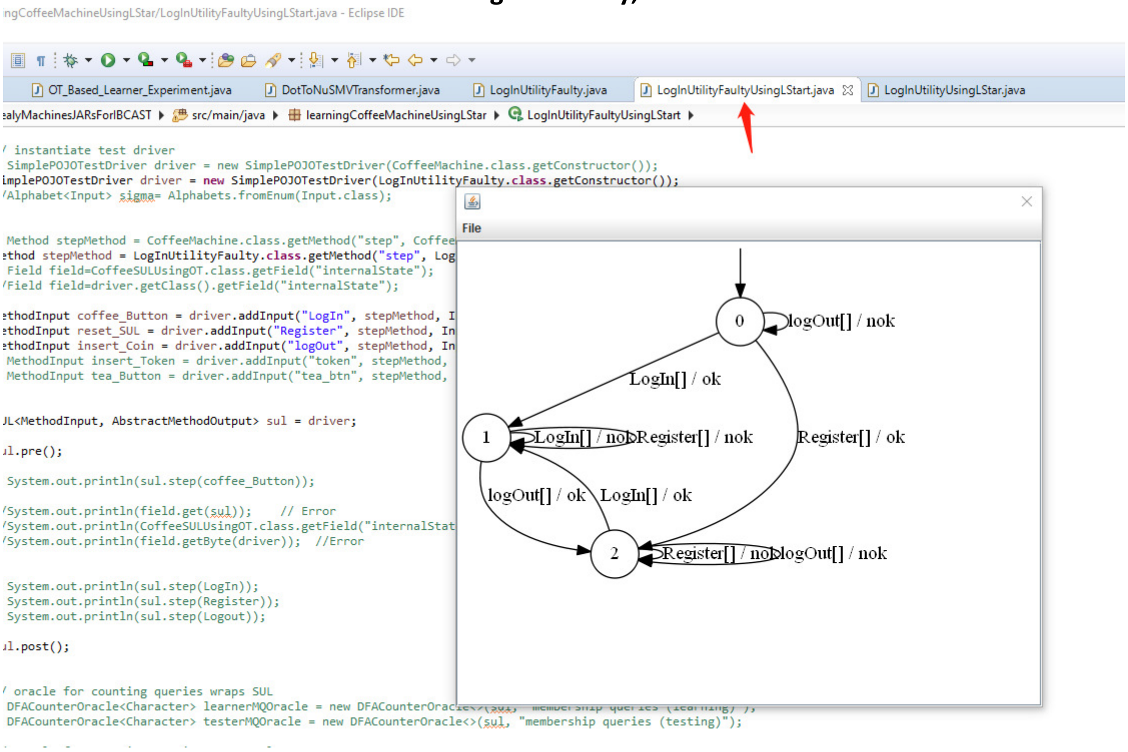
Step-5: Next, we introduce an error in the implementation of ”LogInUtiltiy.java” class. We suppose that the coder has given access to the user (by mistake) where he can LogIn without creating his logIn first (or any implementation error can be considered for demonstration purposes). The next slides highlight the fact that our proposed framework is capturing this erroneous behavior accurately and the same specification (**LTLSPEC NAME spec1 := inp=LogIn -> out=nok**) is now not validated this time (as model checker has not verified it and give us a counterexample for further analysis).

Combining Model Learning and Model Checking to Analyze Java Libraries

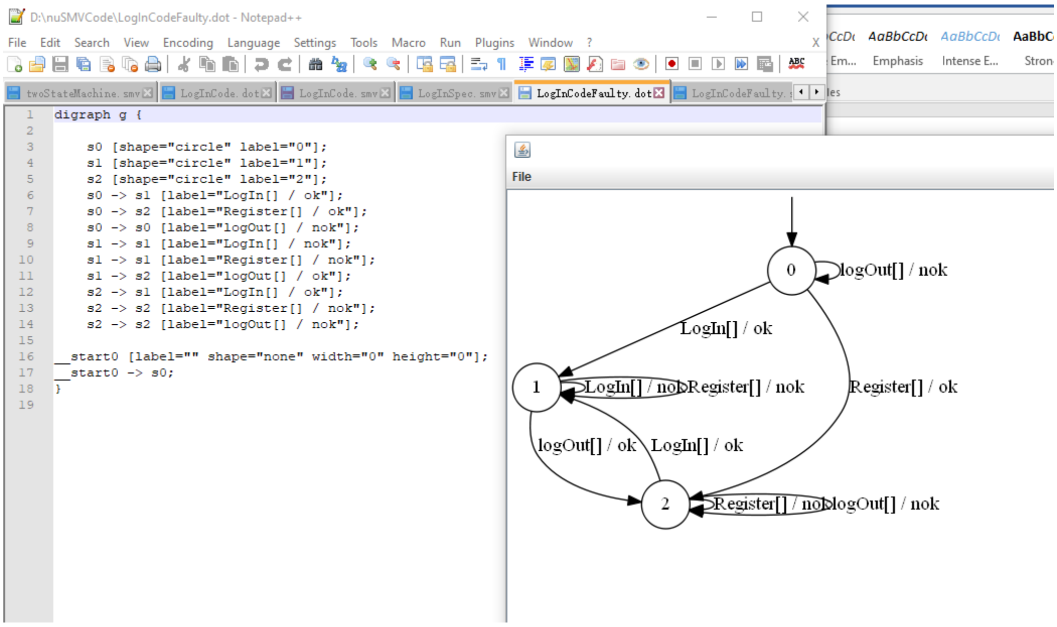
Shahbaz Ali, Hailong Sun, Zhaowang

bazgikian@yahoo.com

Beihang University, China.


ngCoffeeMachineUsingLStar/LogInUtilityFaultyUsingLStart.java - Eclipse IDE
OT_Based_Learner_Experiment.java
DotToNuSMVTransformer.java
LogInUtilityFaulty.java
LogInUtilityFaultyUsingLStart.java
LogInUtilityUsingLStar.java
:alyMachinesJARsForIBCAST
src/main/java
learningCoffeeMachineUsingLStar
LogInUtilityFaultyUsingLStart
File
0
1
2
logOut[] / nok
LogIn[] / ok
LogIn[] / nok
Register[] / nok
Register[] / ok
logOut[] / ok
LogIn[] / ok
Register[] / nok
logOut[] / nok

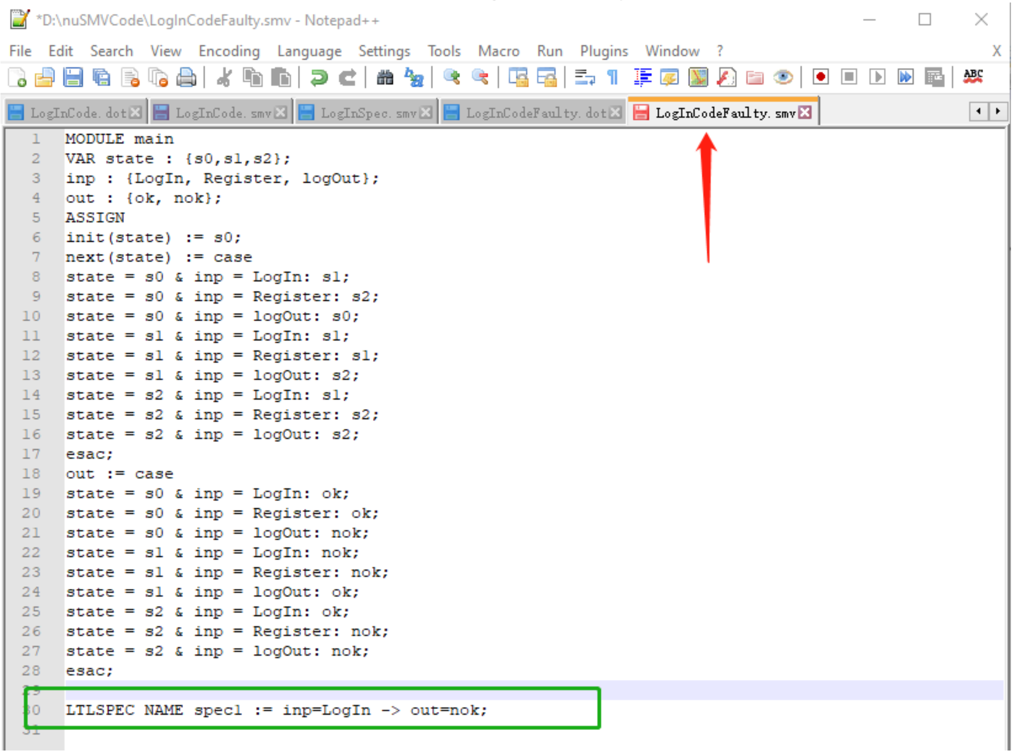
```
/ instantiate test driver
 SimplePOJOTestDriver driver = new SimplePOJOTestDriver(CoffeeMachine.class.getConstructor());
implePOJOTestDriver driver = new SimplePOJOTestDriver(LogInUtilityFaulty.class.getConstructor());
/Alphabet<Input> sigma= Alphabets.fromEnum(Input.class);


 Method stepMethod = CoffeeMachine.class.getMethod("step", Coffee
ethod stepMethod = LogInUtilityFaulty.class.getMethod("step", Log
 Field field=CoffeeSULUsingOT.class.getField("internalState");
/Field field=driver.getClass().getField("internalState");

ethodInput coffee_Button = driver.addInput("LogIn", stepMethod, I
ethodInput reset_SUL = driver.addInput("Register", stepMethod, In
ethodInput insert_Coin = driver.addInput("logOut", stepMethod, In
 MethodInput insert_Token = driver.addInput("token", stepMethod,
 MethodInput tea_Button = driver.addInput("tea_btn", stepMethod,


JL<MethodInput, AbstractMethodOutput> sul = driver;

ıl.pre();

 System.out.println(sul.step(coffee_Button));

/System.out.println(field.get(sul));    // Error
/System.out.println(CoffeeSULUsingOT.class.getField("internalStat
/System.out.println(field.getByte(driver));  //Error


 System.out.println(sul.step(LogIn));
 System.out.println(sul.step(Register));
 System.out.println(sul.step(Logout));

ıl.post();


/ oracle for counting queries wraps SUL
 DFACounterOracle<Character> learnerMQOracle = new DFACounterOracle<>(sul, "membership queries (learning)");
 DFACounterOracle<Character> testerMQOracle = new DFACounterOracle<>(sul, "membership queries (testing)");
```



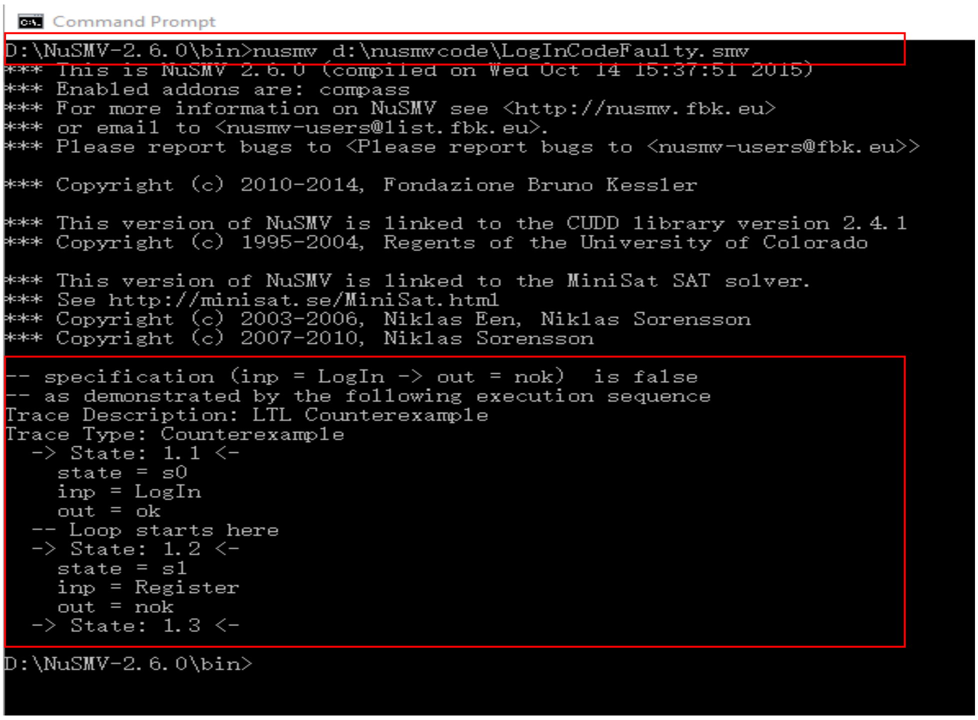
```
digraph g {

    s0 [shape="circle" label="0"];
    s1 [shape="circle" label="1"];
    s2 [shape="circle" label="2"];
    s0 -> s1 [label="LogIn[] / ok"];
    s0 -> s2 [label="Register[] / ok"];
    s0 -> s0 [label="logOut[] / nok"];
    s1 -> s1 [label="LogIn[] / nok"];
    s1 -> s1 [label="Register[] / nok"];
    s1 -> s2 [label="logOut[] / ok"];
    s2 -> s1 [label="LogIn[] / ok"];
    s2 -> s2 [label="Register[] / nok"];
    s2 -> s2 [label="logOut[] / nok"];

__start0 [label="" shape="none" width="0" height="0"];
__start0 -> s0;
}
```

Combining Model Learning and Model Checking to Analyze Java Libraries

Shahbaz Ali, Hailong Sun, Zhaowang
bazgikian@yahoo.com
Beihang University, China.


*D:\nuSMVCode\LogInCodeFaulty.smv - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
LogInCode.dot
LogInCode.smv
LogInSpec.smv
LogInCodeFaulty.dot
LogInCodeFaulty.smv

```
MODULE main
VAR state : {s0,s1,s2};
inp : {LogIn, Register, logOut};
out : {ok, nok};
ASSIGN
init(state) := s0;
next(state) := case
state = s0 & inp = LogIn: s1;
state = s0 & inp = Register: s2;
state = s0 & inp = logOut: s0;
state = s1 & inp = LogIn: s1;
state = s1 & inp = Register: s1;
state = s1 & inp = logOut: s2;
state = s2 & inp = LogIn: s1;
state = s2 & inp = Register: s2;
state = s2 & inp = logOut: s2;
esac;
out := case
state = s0 & inp = LogIn: ok;
state = s0 & inp = Register: ok;
state = s0 & inp = logOut: nok;
state = s1 & inp = LogIn: nok;
state = s1 & inp = Register: nok;
state = s1 & inp = logOut: ok;
state = s2 & inp = LogIn: ok;
state = s2 & inp = Register: nok;
state = s2 & inp = logOut: nok;
esac;

LTLSPEC NAME spec1 := inp=LogIn -> out=nok;
```



```
D:\NuSMV-2.6.0\bin>nusmv d:\nusmvcode\LogInCodeFaulty.smv
*** This is NuSMV 2.6.0 (compiled on Wed Oct 14 15:37:51 2015)
*** Enabled addons are: compass
*** For more information on NuSMV see <http://nusmv.fbk.eu>
*** or email to <nusmv-users@list.fbk.eu>.
*** Please report bugs to <Please report bugs to <nusmv-users@fbk.eu>>

*** Copyright (c) 2010-2014, Fondazione Bruno Kessler

*** This version of NuSMV is linked to the CUDD library version 2.4.1
*** Copyright (c) 1995-2004, Regents of the University of Colorado

*** This version of NuSMV is linked to the MiniSat SAT solver.
*** See http://minisat.se/MiniSat.html
*** Copyright (c) 2003-2006, Niklas Een, Niklas Sorensson
*** Copyright (c) 2007-2010, Niklas Sorensson

-- specification (inp = LogIn -> out = nok)  is false
-- as demonstrated by the following execution sequence
Trace Description: LTL Counterexample
Trace Type: Counterexample
  -> State: 1.1 <-
    state = s0
    inp = LogIn
    out = ok
  -- Loop starts here
  -> State: 1.2 <-
    state = s1
    inp = Register
    out = nok
  -> State: 1.3 <-

D:\NuSMV-2.6.0\bin>
```